

પ્રોગ્રામિંગ , ડેટા સ્ટ્રક્ચર એન્ડ અલ્ગોરિધમ યુઝીંગ પાયથોન

ગુજરાતી





PROGRAMMING, DATA STRUCTURES AND ALGORITHMS USING PYTHON

Gujarati

Translated by

Name	Institute
Mr. Nirav Bhatt	Charotar University of Science and Technology



swayam.gov.in/NPTEL



nptel.ac.in

Index

Sl. No	Topic	Page No
Week 1		
1.	એલ્ગોરિધમ્સ (Algorithms) અને પ્રોગ્રામિંગ(programming): સરળ GCD	4
2.	નાઇવ (naive) GCDની સુધારણા કરવી	12
3.	GCD માટે યુક્લિડનું એલ્ગોરિધમ	17
4.	પાયથોન (Python) ડાઉનલોડ (download) અને ઇન્સ્ટોલ (install) કરવું	24
Week 2		
5.	અસાઇનમેન્ટ સ્ટેટમેન્ટ(Assignment Statement) મૂળભૂત પ્રકારો- int, float, bool	30
6.	સ્ટ્રિંગ્સ (Strings)	37
7.	લિસ્ટ્સ (Lists)	44
8.	કન્ટ્રોલ ફ્લો (Control Flow)	53
9.	ફંક્શન્સ (Functions)	60
10.	ઉદાહરણો	65
Week 3		
11.	રેન્જ (range) વિશે વધુ ()	69
12.	મનિપ્યુલેટિંગ લિસ્ટ્સ (Manipulating lists)	72
13.	લૂપ (loop) માંથી બહાર નીકળવું	78
14.	એરેસ્ (Arrays) vs લિસ્ટ્સ (lists), બાયનરી સર્ચ (Binary Search)	82
15.	કાર્યક્ષમતા	90
16.	સોર્ટ સિલેક્શન (Selection Sort)	94

17	ઇન્સર્શન સોર્ટ (Insertion Sort)	99
18	રિકર્શન (Recursion)	103
Week 4		
19	મર્જસોર્ટ (Mergesort)	109
20	મર્જસોર્ટ (Mergesort), વિશ્લેષણ	117
21	ક્વિકસોર્ટ (Quicksort)	120
22	ક્વિકસોર્ટ (Quicksort)નું વિશ્લેષણ	124
23	ટ્યુપલ્સ (Tuples) અને ડિક્શનરી (dictionaries)	129
24	ફંક્શન(Function)ની વ્યાખ્યાઓ	133
25	લિસ્ટ (List)ની સમજણ	137
Week 5		
26	એક્સેપ્શન હેન્ડલિંગ (Exception Handling)	142
27	સ્ટાન્ડર્ડ ઇનપુટ અને આઉટપુટ (Standard input and output)	148
28	ફાઇલ્સ (files)નું હેન્ડલિંગ (Handling)	153
29	સ્ટ્રિંગ્સ ફંક્શન્સ (String functions)	160
30	ફોર્મેટિંગ પ્રિન્ટેડ આઉટપુટ (Formatting printed output)	166
31	પાસ (pass), ડેલ (del) () અને, નન (None)	169
Week 6		
32	બેકટ્રેકિંગ (Backtracking), N ક્વીન્સ (queens)	172
33	ગ્લોબલ સ્કોપ (Global scope), , નેસ્ટેડ ફંક્શન્સ (nested functions)	184
34	પર્મ્યુટેશન્સ (permutations) ઉત્પન્ન કરવું	188
35	ગણો, સ્ટેક્સ (stacks), ક્યુઝ (queues)	191
Week 7		
36	પ્રાયોરીટી (Priority) ક્યુઝ (queues) અને હિપ્સ (heaps)	197
37	એબ્સ્ટ્રેક્ટ ડેટાટાઇપ્સ (Abstract datatypes), ક્લાસિસ (classes) અને	204

	ઓબ્જેક્ટ્સ (objects)	
38	પાયથોન (Python)માં ક્લાસિસ (Classes) અને ઓબ્જેક્ટ્સ (objects)	210
39	યુઝર્સ ડિફાઇન્ડ લિસ્ટ્સ (User defined lists)	215
40	સર્ચ ટ્રીઝ (Search trees)	223
	Week 8	
41	મેમોઇઝેશન (Memoization) અને ડાઇનેમિક પ્રોગ્રામિંગ (dynamic programming)	231
42	ગ્રીડ પાથ્સ (Grid paths)	236
43	લાંબી સામાન્ય સબસિક્વન્સ (subsequence)	241
44	મેટ્રિક્સ (Matrix) ગુણકાર	247
45	રેપ-અપ (Wrap-up), પાયથોન (Python) vs અન્ય ભાષાઓ	252

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રોફેસર. માધવન મુકુંદ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 01 લેક્ચર - 01

એલ્ગોરિધમ્સ અને પ્રોગ્રામિંગ: સાદી જીસીડી પ્રોગ્રામિંગ ડેટા સ્ટ્રક્ચર્સ અને એલ્ગોરીધમ પરના અભ્યાસક્રમના પ્રથમ વ્યાખ્યાનમાં આપનું સ્વાગત છે. પાયથોનમાં

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 00:10)

ચાલો આપણે એલ્ગોરિધમનો અર્થ શું છે તે મૂળભૂત વ્યાખ્યા સાથે શરૂ કરીએ અને પ્રોગ્રામિંગ શું છે. તમારામાંના મોટાભાગના કદાચ જાણે છે, એક એલ્ગોરિધમનો વર્ણન છે કે કેવી રીતે વ્યવસ્થિત રીતે કેટલાક કાર્ય કરે છે. એક એલ્ગોરિધમનો પગલાઓનો ક્રમ છે, જે કંઈક મેળવવા માટે અમે રેસીપી તરીકે વિચારી શકીએ છીએ. તેથી, અલબત્ત શબ્દ રિકવરી, રસોઈમાંથી આવે છે જ્યાં આપણે ઘટકોની સૂચિ ધરાવે છે અને ત્યારબાદ વાનગી તૈયાર કરવા માટેનાં પગલાઓની શ્રેણી. તેથી, એ જ રીતે એક એલ્ગોરિધમ કંઈક તૈયાર કરવા અથવા આપેલ કાર્યને પ્રાપ્ત કરવા માટેનો એક માર્ગ છે. તેથી, આપણી વસ્તુના સંદર્ભમાં, આપણે એક પ્રોગ્રામ કોલ કરીએ છીએ તે રેસીપી છે. અને પ્રોગ્રામિંગ લેંગ્વેજનો ઉપયોગ કરીને આપણે એક પ્રોગ્રામ લખીશું. તેથી, પ્રોગ્રામિંગ લેંગ્વેજનો ધ્યેય એ જરૂરી છે કે પગલાંની શ્રેણીનું વર્ણન કરવું અને એ પણ વર્ણવવું કે કેવી રીતે જુદાં હોય તો અમે કેવી રીતે પગલાંઓની જુદી જુદી સિક્વન્સ કરી શકીએ. એક પગલાની કલ્પના એ એવી વસ્તુ છે જે એલ્ગોરિધમનો અમલ કરતી હોય તેવું કરી શકાય છે. હવે એક પ્રોગ્રામ મશીન દ્વારા અમલ કરવાની જરૂર નથી, તેમ છતાં કમ્પ્યુટર પ્રોગ્રામિંગના સામાન્ય સંદર્ભમાં આપણે અપેક્ષા રાખીએ છીએ કે કમ્પ્યુટર અમારા પગલાંઓ ચલાવશે. કોઈ વ્યક્તિ દ્વારા એક કાર્યક્રમ ચલાવવામાં પણ આવી શકે છે. હમણાં પૂરતું, હાથ પર કાર્ય ધારણા એક કાર્ય માટે એક હોલ તૈયાર છે. તેથી, તેમાં અલગ અલગ પગલાંઓનો સમાવેશ થાય છે જેમ કે ઓરડામાં સફાઈ, સ્ટેજની તૈયારી કરવી, સુશોભનની સુનિશ્ચિતતા કરવી, ખુરશીની વ્યવસ્થા કરવી વગેરે. આ લોકોની એક ટીમ દ્વારા ચલાવવામાં આવશે. હવે આ જૂથના કુશળતા અને અનુભવ પર આધાર રાખીને, તમે વિગતવાર અલગ અલગ સ્તરે આ એલ્ગોરિધમનો વર્ણન કરી શકો છો. દાખલા તરીકે, ખુરશીની ગોઠવણ જેવી સૂચના એવી રીતે સમજણ આપે છે જો લોકો સામેલ હોય તો તે ચોક્કસપણે અપેક્ષિત છે બીજી બાજુ, જો આ લોકોનો એક નવો ગ્રુપ છે જે આ પહેલાં ક્યારેય ન કર્યો હોય; તમને વધુ વિગતવાર પગલું આપવાનું વર્ણન કરવાની જરૂર પડી શકે છે. હમણાં પૂરતું, તમે કહી શકો છો કે 8 પંક્તિઓમાં ખુરશી ગોઠવો અને દરેક પંક્તિમાં 10 ખુરશી મૂકો. તેથી, એક પગલાની કલ્પના વ્યક્તિલક્ષી છે, તે વ્યક્તિ અથવા મશીન જે અમે એલ્ગોરિધમનો અમલ કરી રહી છે તેનાથી શું અપેક્ષા રાખીએ છીએ તે પર આધાર રાખે છે. અને તે ક્ષમતાની દ્રષ્ટિએ, અમે એલ્ગોરિધમનો પોતે વર્ણન કરીએ છીએ.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 02:44)

આ કોર્સમાં અમારું ધ્યાન કમ્પ્યુટર એલ્ગોરિધમ્સ પર હશે અને સામાન્ય રીતે, આ એલ્ગોરિધમ્સ માહિતીને ચાલાકી કરશે. સૌથી મૂળભૂત પ્રકારની એલ્ગોરિધમનો કે જે આપણે બધા શાળા થી પરિચિત છીએ તે એલ્ગોરિધમ છે જે સંખ્યાત્મક કાર્યોની ગણતરી કરે છે. દાખલા તરીકે, આપણી પાસે એક એલ્ગોરિધમ હોઈ શકે છે જે બે સંખ્યાઓ x અને y લે છે, અને x ને પાવર y ની ગણતરી કરે છે. તેથી, અમે શાળામાં આવા સંખ્યાબંધ કાર્યો જોયા છે. ઉદાહરણ તરીકે, x ના વર્ગમૂળને ગણતરી કરવા માટે, આપણે શાળામાં જે કરીએ છીએ તે x ની વર્ગમૂળને ગણતરી કરવા માટે અમારી પાસે એક જટિલ રીત છે અથવા આપણે x ને વાચથી વિભાજીત કરી શકીએ છીએ, જ્યાં આપણે લાંબી વિવિધન કરીએ છીએ

અને તેથી. આ બધા એલ્ગોરિધમ્સ છે, જે એક અથવા વધુ નંબરો આપવામાં આવેલ મૂલ્યોનું ગણતરી કરે છે, જે તેઓ આ ફંક્શનના આઉટપુટની ગણતરી કરે છે.

(સ્લાઈડ ટાઈમ નો સંદર્ભ લો: 03:35)

પરંતુ આપણા બધાએ કમ્પ્યુટર્સનો ઉપયોગ કર્યો છે, તે જાણે છે કે ઘણાં અન્ય વસ્તુઓ પણ ગણતરીના ક્ષેત્રની અંદર આવે છે. ઉદાહરણ તરીકે, જો આપણે માહિતી ગોઠવવા માટે સ્પ્રેડશીટનો ઉપયોગ કરીએ છીએ અને પછી આપણે સ્ટાંભની સોર્ટ કરવા માંગીએ છીએ. તેથી, આમાં ક્રમમાં આઈટમ્સને ચડતા ક્રમમાં અથવા ઉતરતા ક્રમમાં ગોઠવવામાં આવે છે. તેથી, માહિતીનું પુનર્ગઠન એ એક ગણતરી કાર્ય છે અને આ જાણવું જરૂરી છે કે કેવી રીતે આ એલ્ગોરિધમિક રીતે કરવું. આજની જિંદગીમાં આપણી આસપાસ ફરતા ગણતરી પણ જોવા મળે છે. ઉદાહરણ તરીકે, જ્યારે આપણે પ્રવાસ બુકિંગ સાઇટ પર જઈએ છીએ અને અમે એક શહેરથી બીજા શહેરમાં ફ્લાઇટ બુક કરવાનો પ્રયાસ કરીએ છીએ ત્યારે તે ઓછામાં ઓછા સમય અથવા લઘુત્તમ ખર્ચની શરતોમાં ફ્લાઇટની વ્યવસ્થા કરવાની ઓફર કરશે. તેથી, આ ઓપ્ટિમાઇઝેશન સમસ્યાઓ છે આમાં કોઈ ચોક્કસ રીતે માહિતીની ગોઠવણીનો સમાવેશ થાય છે અને તે પછી કેટલાક જથ્થાને ગણતરીમાં લેવાની જરૂર છે જે અમે ઇચ્છીએ છીએ. આ કિસ્સામાં, આપણે જાણીએ છીએ કે આપણે એકથી બી અને બી થી મેળવી શકીએ છીએ, જેમાંથી આપણે એકથી બી સુધી મેળવી શકીએ છીએ. અને અલબત્ત, ત્યાં ઘણા છે, ઘણી વધુ વસ્તુઓ છે કે જે આજે આપણે જુઓ, જે કમ્પ્યુટર પ્રોગ્રામ દ્વારા ચલાવવામાં આવે છે. અમે રમતો રમી શકે છે હમણાં પૂરતું, અમે સુડોકુને હલ કરી શકીએ છીએ અથવા આપણે પ્રોગ્રામ સામે ચેસ રમી શકીએ છીએ. જ્યારે આપણે દસ્તાવેજ લખવા માટે વર્ડ પ્રોસેસરનો ઉપયોગ કરીએ છીએ અથવા જ્યારે અમે અમારા સેલ ફોનનો ઉપયોગ એસએમએસ સંદેશાઓ માટે કરીએ છીએ, ત્યારે કમ્પ્યુટર અમારા જોડણીમાં ભુલો સૂચવે છે અમે આમાંની કેટલીક બાબતોને આ અભ્યાસક્રમમાં જોશું, પરંતુ બિંદુએ એ નોંધવું છે કે અમારા સંદર્ભમાં કોઈ પ્રોગ્રામ માહિતીને જુએ છે અને તેને આપેલ જરૂરિયાતમાં પરિવર્તિત કરે છે. તેથી, તે માત્ર એક નંબર લેવાનો અને નંબરને બહાર મૂકવાનો પ્રશ્ન નથી. તે વસ્તુઓ ફરીથી ગોઠવવા સમાવેશ કરી શકે છે તે વધુ જટિલ વસ્તુઓને ગણતરીમાં લઈ શકે છે તે ચોક્કસ રીતે માહિતીનું આયોજન કરી શકે છે, તેથી આ ગણતરીઓ વધુ ધ્યાનપાત્ર બને છે અને તે જ અમે ડેટા માળખું કહીએ છીએ.

(સ્લાઈડ ટાઈમ નો સંદર્ભ લો: 05:36)

આને સમજાવવા માટે ચાલો આપણે એ કાર્યને જોવું જોઈએ કે જેમાંથી મોટાભાગના લોકોએ જોયું છે અને અલ્ગોરિધમનીક રીતે સમજવા પ્રયત્ન કરો. તેથી, જે ગુણધર્મ કે જે હું ગણતરી કરવા ઇચ્છું છું તે બે મહાન હકારાત્મક પૂર્ણાંકો એમ અને n નો ભાગાકાર છે. તેથી, જેમ આપણે જાણીએ છીએ કે વિભાજક એ સંખ્યા છે જે વિભાજન કરે છે. તેથી k એ m ના ભાજક છે; જો હું કે દ્વારા મીટર વિભાજીત કરી શકું અને કોઈ રીમાઇન્ડર નહીં મળે તેથી, m અને n નો સૌથી સામાન્ય ભાગાકાર કંઈક હોવો જોઈએ જે સામાન્ય ભાજક છે. તેથી, સામાન્ય અર્થ એ છે કે તે બંનેને વિભાજિત કરવો પડે છે અને તે આમાંથી સૌથી મોટો હોવો જોઈએ. તેથી, જો સૌથી મોટી K જેમ કે k વિભાજન મીટર અને k પણ વિભાજન મીટર દાખલા તરીકે, જો આપણે 8 અને 12 ની સંખ્યા જોઈએ, તો આપણે જોઈ શકીએ છીએ કે 4 એ 8 નો પરિબળ છે, 4 એ 8 નું ભાગાકાર છે, 4 પણ 12 ની ભાજક છે, 12 નો બીજો ભાગાકાર 6 છે, પરંતુ 6 જો આપણે 8 અને 12 ના વિભાજકથી પસાર થઈએ તો તે તપાસવું સહેલું છે કે 8 અને 12 બંનેમાં ભાગ લેનાર સૌથી મોટી સંખ્યા 4 છે. તેથી, 8 અને 12 ની જીસીડી 4 છે. શું 18 અને 25 25 ની 5 ઘાત 5 છે. તેથી, તે 1 અને 25 સિવાયના એક જ ભાગાકાર છે, જે 5 છે. અને 5 એ 18 ના ભાજક નથી, પરંતુ સદભાગ્યે 1 ની ભાજક છે. તેથી, આપણે કહી શકીએ કે જીસીડી 18 અને 25 1 છે; ત્યાં 1 થી મોટો કોઈ સંખ્યા નથી કે જે 18 અને 25 બંનેને વિભાજન કરે છે. કારણ કે 1 દરેક સંખ્યાને વિભાજિત કરે છે, કારણ કે જેમ આપણે 18 અને 25 ના કિસ્સામાં જોયું તેમ, હંમેશા બે નંબરોમાં ઓછામાં ઓછો એક સામાન્ય ભાજક હોવો જોઈએ. જીસીડી હંમેશાં સારી રીતે વ્યાખ્યાયિત થશે; તે ક્યારેય નહીં બનશે કે આપણે સામાન્ય વિભાજક શોધી શકતા

નથી અને કારણ કે તમામ સામાન્ય વિભાજક સંખ્યાઓ હશે, અમે તેને નાનાથી મોટામાં ગોઠવી શકીએ છીએ અને સૌથી મોટું સામાન્ય ભાજક તરીકે પસંદ કરી શકીએ છીએ. તેથી, સકારાત્મક સંખ્યા m અને n ના કોઈ પણ જોડીને, અમે ચોક્કસપણે આ બે નંબરોના જુસીડીની ગણતરી કરી શકીએ છીએ.

(સ્લાઈડ ટાઈમ નો સંદર્ભ લો: 07:39)

તો, એમ કેવી રીતે એમ, એન ની જુસીડી ગણાય છે? તેથી, આ એ છે કે જ્યાં આપણે એલ્ગોરિધમિક રીતે આવીએ છીએ, અમે કોઈપણ મીટર અથવા કોઈપણ n એ માટે mn ના વ્યવસ્થિત રીતે કમ્પ્યુટિંગ જુસીડીની એકસમાન રીતનું વર્ણન કરવા માગીએ છીએ. તેથી, અહીં એક ખૂબ સરળ પ્રક્રિયા છે. તે સૌથી કાર્યક્ષમ નથી; અમે આગળ વધીએ છીએ ત્યારે એક વાર આપણે વધુ સારી રીતે જોશું. પરંતુ જો આપણે માત્ર જુસીડીની વ્યાખ્યા જોઈએ તો તે કહે છે કે હું એમના તમામ પરિબલોને જોઉં છું, n ના તમામ પરિબલોને જોઉં છું અને સૌથી મોટો એક છે જે બંનેનું પરિબલ છે. તેથી, આ કરવાના નિષ્કપટ માર્ગ એમના પરિબલોની પ્રથમ યાદી હશે, પછી બીજા ક્રમાંકના તમામ પરિબલોની સૂચિબદ્ધ થશે અને પછી આ બે યાદીઓમાં બન્ને યાદીઓમાં સૌથી વધુ સંખ્યામાં દેખાશે. આ લગભગ શાબ્દિક જુસીડીની વ્યાખ્યા છે. હવે પ્રશ્ન એ છે કે આ એક અલ્ગોરિધમનો છે. સાચું, જો આપણે એક પગલું તરીકે પરિબલોને યાદીમાં મૂકવાનો વિચાર કરીએ, તો અલ્ગોરિધમથી આપણે જે જોઈએ છીએ તે બે બાબતો છે. એક તે છે કે શું કરવું તેનું વર્ણન મર્યાદિત રીતે લખવું જોઈએ. આ અર્થમાં કે હું મૂલ્ય m અને n ને અનુલક્ષીને સૂચનાને લખી શકવા સક્ષમ હોવી જોઈએ જેથી તે તેને વાંચી શકે અને તેને એકવાર અને બધા માટે સમજી શકે. અહીં ખૂબ જ સ્પષ્ટ છે, અમારી પાસે ત્રણ પગલાંઓ બરાબર છે. તેથી, અલ્ગોરિધમનો રચના કરવા માટે અમારી પાસે ત્રણ પગલાંઓ છે જેથી તે ચોક્કસ રીતે મર્યાદિત રીતે રજૂ કરે. એલ્ગોરિધમની અન્ય આવશ્યકતા એ છે કે આપણે સચોટ સંખ્યાના પગલાં પછી જવાબ મેળવવો જોઈએ. આ મર્યાદિત સંખ્યામાં સંખ્યાઓ m અને n ના જુદા જુદા મૂલ્યો માટે અલગ હોઈ શકે છે, તમે કલ્પના કરી શકો છો કે જો તમારી પાસે n માટે બહુ નાની સંખ્યા છે તો ઘણા પરિબલો ન હોય તો n માટે તમારી પાસે ઘણાં પરિબલો હશે. તેથી, એમ અને એન ના પરિબલોને બહાર કાઢવાની પ્રક્રિયાને લાંબુ સમય લાગી શકે છે; જો કે, અમે ખાતરી આપવી જઈએ છીએ કે આ પ્રક્રિયા હંમેશાં સમાપ્ત થશે અને તે પછી તે પૂર્ણ થશે તો અમે હંમેશા બંને યાદીઓમાં દેખાતી સૌથી મોટી સંખ્યા શોધી શકશે.

(સ્લાઈડ ટાઈમ નો સંદર્ભ લો: 09:45)

એવી દલીલ કરવા માટે કે આ પ્રક્રિયા સારી રીતે વ્યાખ્યાયિત થયેલ છે જે આપણે ખ્યાલ રાખવી જોઈએ કે મીટરના પરિબલો 1 અને મીટર વચ્ચે હોવા જોઈએ બીજા શબ્દોમાં કહીએ તો, અલબત્ત, અલબત્ત, અસીમિત ઘણાં વિવિધ સંભાવના છે કારણ કે પરિબલોને અમે મીટર કરતાં મોટી સંખ્યામાં જોવાની જરૂર નથી, કારણ કે તે મીટરમાં સમાન રીતે ન જઈ શકે. તેથી, મીટરના પરિબલોની ગણતરી કરવા માટે અમારે બધા જ જરૂર છે તે ચકાસવા એ છે કે દરેક સંખ્યા એકથી મીટર છે અને જો તે સ્મૃતિપત્ર વિના મીટર વહેંચે, તો આપણે તેને પરિબલોની સૂચિમાં ઉમેરીએ છીએ. તેથી, અમે પરિબલોની ખાલી સૂચિ સાથે શરૂઆત કરીએ છીએ અને અમે તેને 1, 2, 3, 4 અને મીટર સુધી ધ્યાનમાં લઈએ છીએ અને દરેક નંબર માટે અમે તપાસ કરીએ છીએ, ભલે આપણે આ સંખ્યા દ્વારા મીટર વિભાજીત કરીએ તો આપણે 0 ની સ્મૃતિપત્ર મેળવીએ ની 0 અમે તેને યાદીમાં ઉમેરીએ છીએ. ચાલો આપણે કોંક્રિટનું ઉદાહરણ જોઈએ, ચાલો 14 અને 63 ના જુસીડીની ગણતરી કરવાનો પ્રયાસ કરીએ. તેથી, અમારા અલ્ગોરિધમનો પ્રથમ પગલું 14 ના પરિબલોની ગણતરી કરવા કહે છે. તેથી, 14 ની પરિબલો ઉપરના આપણા નિરીક્ષણ દ્વારા એક વચ્ચે રહેવું જોઈએ. અને 14 કરતા મોટો નહીં એક પરિબલ હોઈ શકે છે. તેથી, અમે એક અને 14 વચ્ચેનાં અમારા બધા સંભવિત પરિબલોની યાદી શરૂ કરીએ છીએ અને તેમને પરીક્ષણ કરીએ છીએ. તેથી, આપણે અલબત્ત જાણીએ છીએ, કે 1 હંમેશાં ભાગલા પાડશે; આ કિસ્સામાં 2 14 ને વિભાજીત કરે છે, કારણ કે 14 વિભાજન 2 એ 7 છે બાકીની સાથે. હવે 3 ભાગલા નથી, 4 ભાગલા નથી, 5 ભાગલા નથી, 6 ભાગલા નથી; પરંતુ 7 કરે છે, કારણ કે જો

આપણે 14 થી 7 ને વિભાજીત કરીએ છીએ તો આપણે 0 ની બાકીની રકમ મેળવી શકીએ છીએ. પછી 8 ભાગલા નથી, નવ ભાગલા નથી અને તેથી પર. અને આખરે, આપણે શોધી કાઢીએ છીએ કે માત્ર એક જ અન્ય પરિબળ બાકી 14 છે. તેથી દરેક નંબર માટે $m - 1$ અને m એ પરિબળો હશે અને ત્યારબાદ વચ્ચેના પરિબળો હોઈ શકે છે.

(સ્લાઇડ સમય જણાવો: 11:35)

તેથી, આ કરવાથી અમે ઓળખી કાઢ્યાં છે કે 14 ના પરિબળો અને આ પરિબળો ચોક્કસપણે 1, 2, 7 અને 14 છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 11:40)

14 અને 63 ની પરિપક્વતા એ 63 ના પરિબળોની ગણતરી કરવા માટે છે. તેથી, તે જ રીતે આપણે બધી સંખ્યાઓ એકથી 63 ની નીચે લખીશું અને આપણે તપાસ કરીશું કે કયા ભાગો વહેંચે છે. તેથી, ફરીથી આપણે શોધીશું કે 1 ભાગો, અહીં 2 છે વિભાજન નહીં; કારણ કે 63 પણ નથી, 3 વિભાજન કરે છે, પછી આપણે અહીં સંખ્યાઓનો સમૂહ શોધીએ છીએ, જે ભાગલા નથી. પછી અમે શોધી કાઢીએ છીએ કે 7 વિભાજન થાય છે, કારણ કે 7 9 ના 63 છે. પછી ફરી 8 વિભાજન નથી, પરંતુ 9 કરે છે. પછી ફરીથી સંખ્યાઓનો મોટો તફાવત છે, જે ભાગલા નથી. અને પછી 21 વિભાજીત થાય છે, કારણ કે 21 3 ની 63 છે. અને પછી છેવટે, આપણે શોધી કાઢીએ છીએ કે અમારી પાસે છેલ્લા પરિબળ છે 63. તેથી, જો આપણે આ પદ્ધતિથી એક ક્રમાંકથી પસાર કરીએ 63 દરેક નંબર પાર કરીએ જે કોઈ પરિબળ નથી સૂચિ 1, 3, 7, 9, 21 અને 63 ની સાથે અંત થાય છે. અમારા નંબરો 14 અને 63 ના પરિબળોને ગણતરીમાં લીધા પછી, અમારા અલ્ગોરિધમનો આગળનું પગલું કહે છે કે આપણે બન્ને સૂચિમાં સૌથી મોટું પરિબળ શોધવું જોઈએ. તો, આપણે આ કેવી રીતે કરી શકીએ, આપણે સામાન્ય પરિબળોની યાદી કેવી રીતે બનાવી શકીએ? હવે આ કરવા માટે વધુ હોશિયાર રીત છે, પરંતુ અહીં એક બહુ સરળ રીત છે. અમે ફક્ત એક સૂચિમાં જઈએ છીએ કે 14 ની પરિબળોની સૂચિ અને દરેક વસ્તુની યાદીમાં આપણે તપાસીએ છીએ તે 63 નો પરિબળ છે. તેથી, આપણે 1 થી શરૂ કરીએ છીએ અને આપણે કહીએ છીએ કે તે 63 ના પરિબળ તરીકે દેખાય છે. તેથી અમે સામાન્ય પરિબળો યાદીમાં ઉમેરો પછી આપણે 2 ને જોશું તો આપણે પૂછવું જોઈએ; તે દેખાતું નથી તેથી અમે તેને છોડી દો. પછી આપણે 3 પર નજર કરીએ છીએ અને 7 ને નજર કરીએ છીએ અને અમે શોધી કાઢીએ છીએ કે અમે 7 ઉમેરીએ છીએ. પછી છેવટે, આપણે 14 પર નજર કરીએ છીએ અને શોધી કાઢો કે 14 દેખાતું નથી તેથી અમે તેને છોડી દઈએ છીએ. આ રીતે અમે વ્યવસ્થિત રીતે 1, 2, 7 અને 14 ની વટાવી ગયા છીએ અને આમાંના માત્ર 1 અને 7 બન્ને સૂચિમાં દેખાય છે. અને હવે આ કાર્ય હોવાથી અમારી પાસે તમામ સામાન્ય પરિબળોની સૂચિ છે જે અમે તેમને નાનાથી સૌથી મોટી ગણ્યા, કારણ કે અમે 14 ના પરિબળોમાં ચડતા ક્રમમાં તેથી, આ સૂચિ પણ ચડતા ક્રમમાં હશે. તેથી, સૌથી મોટો પરિબળો પાછો ફરે તો આ સૂચિમાં સૌથી વધુ પરિબળ આપે છે. 7. આ અમારા ફંક્શનનું આઉટપુટ છે. અમે 14 ના પરિબળોની ગણતરી કરી છે, 63 ના પરિબળોની ગણતરી કરી છે, 14 ની દરેક પરિબળ માટે વ્યવસ્થિત રીતે ચકાસાયેલ છે, પછી ભલે તે 63 ના પરિબળ છે અને અહીં સામાન્ય પરિબળોની સૂચિની ગણતરી કરવામાં આવી છે અને ત્યારબાદ આ સૂચિમાંથી આપણે સૌથી મોટું એક કાઢ્યું છે અને આ હકીકત છે, અમારા જીસીડી છે. આ એલ્ગોરિધમ કેવી રીતે ચલાવશે તેનું ઉદાહરણ છે.

(સ્લાઇડ સમયનો: 14:20)

જો તમને તેને વધુ વિગતમાં લખવાનું હોય તો, આપણે કહી શકીએ કે આપણે નોંધ રાખવાની જરૂર છે કે આપણે આ યાદીઓને યાદ રાખવાની જરૂર છે, અને પછી તેમને પાછા આવો. તેથી, આપણે 14 નાં પરિબળોની ગણતરી કરવાની જરૂર છે, બાજુને રાખીએ છીએ કે આપણે તેને લખવાની જરૂર છે ક્યાંક

આપણે ગણતરી કરી શકીએ કે 63 નું અવયવ ક્યાંક લખ્યું છે અને પછી આ બે યાદીઓની તુલના કરો. તેથી, અન્ય શબ્દોમાં આપણે આને સંગ્રહિત કરવા માટે કેટલાક નામો આપવાની જરૂર છે. ચાલો આપણે આ યાદીના નામો તરીકે એક અને એમના પરિબલો માટે એફએમ કહીએ. તેથી, આપણે શું કરીએ તે છે કે આપણે નંબરો એક થી મીટર સુધી ચાલે છે. અને દરેક j માટે, આ સૂચિમાં 1 મીટર માં અમે તપાસો કે શું j વિભાજન મીટર, શું j રીમાઇન્ડર તરીકે 0 દ્વારા વિભાજિત છે અને જો આપણે એમ અથવા એફએમ યાદી પરિબલો તેને ઉમેરો એ જ રીતે, દરેક j થી 1 થી n માટે આપણે તપાસીએ છીએ, કે શું j એ વિભાજન કરે છે અને જો આપણે તેને યાદીમાં ઉમેરીએ. હવે આપણી પાસે બે યાદીઓ એફએમ અને એફએન છે જે મીટરના પરિબલો અને n નો પરિબલો છે. હવે અમે સામાન્ય પરિબલોની સૂચિ ગણતરી કરવા માંગીએ છીએ, જેને આપણે cf ને બોલાવીશું. તેથી, આપણે જે કરીએ છીએ તે દરેક એક માટે છે જે પ્રથમ ક્રમાંકનો પરિબલ છે, યાદ રાખો કે અમારા કેસ 14 હતા, જ્યાં પ્રત્યેક એક હતી તેથી અમે અમારા કેસમાં 1, 2, 7 અને 14 થી ચાલી હતી. તેથી, દરેક એક સૂચિ માટે આપણે સામાન્ય પરિબલોની યાદીમાં એક ઉમેરીએ છીએ જો તે અન્ય સૂચિમાં પણ દેખાય છે. તેથી, બીજી સૂચિમાં જો તમે નંબર 1, 3, 7, 9, 21 અને 63 છે. તેથી, આપણે આ સૂચિની સરખામણી કરીએ છીએ અને જો આપણે તેને શોધીશું તો અમે તેને cf માં ઉમેરીશું.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 15:58)

અને આ કાર્યા પછી હવે આપણે સામાન્ય પરિબલોની સૂચિનું સૌથી મોટું મૂલ્ય પરત કરવા માંગીએ છીએ. યાદ રાખો કે એક હંમેશા સામાન્ય પરિબલ હશે. તેથી, સૂચિ cf ખાલી રહેશે નહીં. ત્યાં ઓછામાં ઓછો એક મૂલ્ય હશે, પરંતુ ત્યારથી આપણે તેમને એફએમ અને એફએનથી ચઢતા ક્રમાંક ઉમેરીએ છીએ, જ્યાં 1 થી મીટર અને 1 થી n નું સૌથી મોટું મૂલ્ય નિર્માણ કરવામાં આવે છે તે સૌથી વધુ મૂલ્ય છે. આ અમને જુસીડી માટે થોડો વધુ વિગતવાર અલ્ગોરિધમ આપે છે. મીટરનાં પરિબલોની સૂચિની ગણતરી કેવી રીતે કરવી અને કેવી રીતે n ના પરિબલોની સૂચિની ગણતરી કરવી અને આ બે યાદીઓ વચ્ચેના સૌથી સામાન્ય પરિબલને કેવી રીતે ગણતરી કરવી તે વધુ વિગતવાર કરતાં વધુ વિગતવાર છે. તેથી, પહેલાં આપણે ત્રણ અમૂર્ત નિબંધો આપ્યા હતા, હવે અમે 6 માં વિસ્તૃત થઈ ગયા છીએ, સહેજ વધુ વિગતવાર સ્ટેટમેંટો.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 16:47)

આ પહેલેથી જ આપણો પ્રથમ પાયથન પ્રોગ્રામ લખવા માટે અમને પૂરતી માહિતી આપે છે. અલબત્ત, આપણે થોડી વધુ શીખવાની જરૂર છે, તે પહેલાં આપણે તેને કેવી રીતે લખવા તે જાણીએ છીએ, પરંતુ અમે તે ચોક્કસપણે કેવી રીતે વાંચવું તે સમજી શકીએ છીએ. તો, આ પિથન પ્રોગ્રામિંગ શું કરી રહ્યું છે તે આપણે પહેલાના પગલામાં અનૌપચારિક રીતે વર્ણવેલ છે. પિથન પ્રોગ્રામમાં પ્રથમ વસ્તુ એક રેખા છે જે કાર્યને વ્યાખ્યાયિત કરે છે. તેથી, આપણે એમ કોમિયા n નો ફંક્શન જુસીડી વ્યાખ્યાયિત કરી રહ્યા છીએ. તેથી, m અને n એ બે આર્ગ્યુમેંટ્સ છે જે કોઈપણ કાર્ય જેવા કોઈપણ સંખ્યા હોઈ શકે છે. જ્યારે આપણે ગણિતમાં XY ના F ને વાંચીએ છીએ ત્યારે તેનો અર્થ એ કે એક્સ અને વાય આર્બિટ્રેટર મૂલ્યો છે અને પ્રત્યેક x અને y માટે મૂલ્યો પર આધાર રાખીને કંઈક કરીએ જે આપણે સાથે કાર્યને કોલ કરીએ છીએ. તેથી, આ કહે છે કે આ એક વ્યાખ્યા છે, તેથી કાર્ય $gcd\ m, n$ ની વ્યાખ્યા માટે def . હવે પ્રથમ પગલું એમના પરિબલોની સૂચિની ગણતરી કરવા માટે છે. પાયથનમાં ચોરસ કૌસનો ઉપયોગ કરીને આપણે એક યાદી લખીએ છીએ. તેથી, યાદી xyz અને તેથી પર લખાયેલ છે. તેથી, ખાલી સૂચિ માત્ર એક ખુલ્લું કૌસ અને એક ચોરસ બંધ કૌસ છે. તેથી, આપણે પરિબલોની ખાલી સૂચિ સાથે શરૂ કરીએ છીએ. તેથી, આ સમાનતા એ મૂલ્ય નિર્ધારિત કરવું. તેથી, અમે ખાલી યાદી માટે મીટરના પરિબલોની યાદી એફએમને અસાઇન કરીએ છીએ. હવે આપણે રેન્જ 1 થી n માં દરેક વેલ્યુ ચકાસવાની જરૂર છે. હવે પાયથન શ્રેણીમાં કાર્ય કરે છે તે આંતરિક બનેલ છે, પરંતુ આપણે જોશું કારણ કે પાયથનની અજોડતાને લીધે તમને સિવાય રેન્જ નહીં, પરંતુ એક ઓછી. તેથી, જો હું કહું કે સંખ્યા 1 થી n વત્તા 1 માં

આપીશ, તો તે મને એક શ્રેણીમાં મીટર પર આપે છે, એક અપ ઉચ્ચ મર્યાદા સુધી, પરંતુ ઉપરની મર્યાદા શામેલ નથી. તેથી, આ કહેશે કે હું એક બે ત્રણ ઉપર મીટર સુધી લઈ જઈશ. 1 ની દરેક કિંમતો માટે, અમે તપાસીએ છીએ કે આ સાચું છે. હવે ટકાવારી બાકી કામગીરી છે. તે તપાસ કરે છે કે શું બાકીના ભાગો 1 દ્વારા 0 વડે વહેંચે છે. જો બાકીની ભાગ 1 દ્વારા વહેંચાયેલ 0 હશે તો આપણે તે સૂચિ F_n માં ઉમેરવું પડશે, આપણે તે જમણા એપેન્ડમાં ઉમેરવું એ અંગ્રેજી શબ્દ છે જેનો અર્થ એ છે કે સૂચિનો અંત તેથી, અમે n ને જોડીએ છીએ. તેથી, આ પગલામાં, અમે એક્સએમ ગણતરી કરી છે. આ બરાબર છે કે આપણે પહેલાંના ઉદાહરણમાં અનૌપચારિક રીતે લખ્યું હતું કે આપણે ફક્ત એટલું જ કહ્યું હતું કે હું 1 થી મીટર એક માટે એક ઉમેરું છું જો હું મીટર વિભાજિત કરું અને હવે અમે તેને પાયથન વાક્યરચના કર્યું છે. તેથી, અમે પરિબલોની ખાલી સૂચિ વ્યાખ્યાયિત કરી છે અને તે રેન્જમાં દરેક સંખ્યા માટે અમે તપાસ કરી છે કે તે ભાજક છે અને તે પછી તેને ઉમેરો. અને હવે અહીં આપણે બરાબર એ જ વસ્તુ n માટે કરીએ છીએ. તેથી, અમે આ શ્રેણીમાં દરેક જે માટે n નાં પરિબલોની ખાલી સૂચિ સાથે શરૂ કરીએ છીએ જો તે વિભાજિત કરે તો આપણે તેને જોડીએ છીએ હવે, આ બિંદુએ અમારી પાસે બે યાદી એક્સએમ અને એક્સએન છે. હવે, આપણે સામાન્ય પરિબલોની સૂચિ ગણતરી કરવા માગીએ છીએ. તેથી, સામાન્ય પરિબલોની સૂચિ દર્શાવવા માટે આપણે સીએક્સનો ઉપયોગ કરીએ છીએ. શરૂઆતમાં કોઈ સામાન્ય પરિબલો નથી. હવે, પ્રથમ યાદીમાં દરેક પરિબલ માટે જો પરિબલ બીજી સૂચિમાં દેખાય છે તો આપણે તેને cf માં જોડીએ છીએ. તેથી, એ જ ફક્શન એડેન્ડનો સમગ્ર ઉપયોગ થઈ રહ્યો છે. ફક્ત એક સૂચિ લો અને મૂલ્ય ઉમેરો કયા મૂલ્ય? અમે તે મૂલ્ય ઉમેરીએ છીએ જે હવે અમે જોઈ રહ્યા છીએ તે શરતોને સંતોષે છે તેથી, પહેલાં આપણે ઉમેરતા હતા કે ભાગાકાર 0 હતું, બાકીની 0 હતી, હવે અમે તેને ઉમેરી રહ્યા છીએ તે બન્ને સૂચિમાં દેખાય છે. પ્રથમ યાદીમાં દરેક એક માટે જો તે બીજી સૂચિમાં દેખાય છે તો તે ઉમેરો. આ પછી આપણે એક્સએમ, સીએક્સ. અને હવે અમે જમણી સૌથી તત્વ માંગો છો તેથી, આ ફક્ત કેટલાક પાયથન વાક્યરચના છે જો તમે જુઓ છો કે તેના બદલે, જો આપણે ડાબી બાજુથી ગણતરી કરવાનું શરૂ કરીએ તો પછી સૂચિમાં સ્થાનો 0, 1, 2, 3 અને 4 ની સંખ્યામાં હોય છે. પરંતુ પાયથોન પાસે શોર્ટકટ છે જે કહે છે કે તમે જમણાથી ગણતરી કરવા માંગો છો, તો આપણે નંબરોને બાદબાકી 1, ઓછા 2 અને તેથી આગળ વધીએ છીએ. તેથી, તે કહે છે કે સીએક્સના બાદબાકી 1 ની એવરેજ પરત કરો કે જે પાયથોન જાગ્રનમાં છે જેનો અર્થ એ છે કે સૌથી વધુ તત્વ તેથી, આ સૌથી વધુ તત્વ છે આ બિંદુએ તે સમજવા માટે પૂરતું છે કે આપણે વાસ્તવમાં આ પ્રોગ્રામને આ પ્રોગ્રામનો પ્રયાસ અને ડીકોડ કરી શકીએ છીએ, ભલે આપણે કેટલાક સ્થળોએ કોલોન શા માટે ઉપયોગમાં લઈ રહ્યા છીએ અને શા માટે આપણે કંઈક દબાણ કરી રહ્યા છીએ તે સમજી ન શકતા. અહીં અન્ય વાક્યરચનાની સૂચનાઓ છે, તેથી ઉદાહરણ તરીકે, તમારી પાસે આ વિરામચિહ્ન છે, જે આ કોલોન જેવા થોડી વિચિત્ર છે. પછી તમારી પાસે એ હકીકત છે કે આ રેખા આ રેખાના સંદર્ભમાં ઇન્ડેન્ટેડ છે, આ રેખા આ રેખા પર ઇન્ડેન્ટેડ છે આ બધા લક્ષણો છે કે જે અન્ય ભાષાઓમાં પ્રોગ્રામ્સ વાંચવા અને વાંચવા માટે પાયથન પ્રોગ્રામ્સને થોડું સરળ બનાવે છે. તેથી, જ્યારે આપણે પાયથોન સિન્ટેક્સ વધુ ઔપચારિક રીતે શીખીએ ત્યારે આ પર આવીશું. પરંતુ આ બિંદુએ તમે પોતાને સમજાવી શકશો કે પાયથન પગથિયાં આ સમૂહ અનૌપચારિક અલ્ગોરિધમનો એક ખૂબ વફાદાર રેન્ડરીંગ છે જે આપણે પહેલાની સ્લાઇડમાં લખ્યું હતું.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 22:08)

ચાલો આપણે એવા કેટલાક મુદ્દાઓની નોંધ લઈએ કે જે આપણે આ ચોક્કસ ઉદાહરણથી જાણી શકીએ છીએ. તેથી, પ્રથમ મહત્વપૂર્ણ મુદ્દો એ છે કે અમને મધ્યવર્તી મૂલ્યોનો સાચો માર્ગ જાળવવાની જરૂર છે. તેથી, આપણી દલીલોનાં નામ અને નામોથી શરૂ થવાના બે નામ છે. પછી આપણે આ ત્રણ નામોનો ઉપયોગ આ પરિબલો અને સામાન્ય પરિબલોની ગણતરી માટે કરીએ છીએ અને અમે આઈ, જે અને એક્સ જેવા અન્ય નામોનો ઉપયોગ કરીએ છીએ. આ દ્વારા ચલાવવા માટે. અમને 1 થી n સુધી ચાલવાની જરૂર છે આપણને 1 થી n સુધી ચાલવા માટે j ની જરૂર છે. અલબત્ત, અમે હું ફરી ઉપયોગ કરી શકે છે પરંતુ તે ઠીક છે. અમે cf માં તમામ પરિબલો દ્વારા ચલાવવા માટે f નો ઉપયોગ કરીએ છીએ. તેથી, મધ્યવર્તી મૂલ્યોનું ધ્યાન રાખવાની આ બધી રીતો છે નોંધ કરવા માટે બીજો મુદ્દો એ છે કે મૂલ્ય એક

આઈટમ હોઈ શકે છે. ઉદાહરણ તરીકે, એમએન સંખ્યાઓ છે, એ જ રીતે હું, j અને f દરેક પગલે નંબરો છે. તેથી, આ એકલ મૂલ્યો હશે અથવા તે સંગ્રહો હોઈ શકે છે તેથી, ત્યાં સૂચિ છે તેથી એફએમ એક યાદી છે, fn એ એક સૂચિ છે. તેથી, તે એક જ નામ છે જે આ કિસ્સામાં મૂલ્યોનો સંગ્રહ સૂચવે છે જેમાં સૂચિની પ્રથમ ક્રમ અને પછીની સ્થિતિ અને છેલ્લી સ્થિતિ છે. આ સંખ્યાઓની સૂચિ છે કોઈ અન્ય સંગ્રહોની કલ્પના કરી શકે છે અને અમે તે સાથે જઈએ છીએ. તેથી, સંગ્રહો મહત્વનું છે, કારણ કે જો કોઈ કાર્યક્રમ લખવા માટે ખૂબ જ મુશ્કેલ હોય તો અમે દરેક પરિબલને અલગથી નામ આપવાનું રાખવું પડે. મીટરના બધા પરિબલો માટે અમને એકસાથે નામની જરૂર છે. આ નામો સૂચિત કરી શકે છે એક મૂલ્યો અથવા મૂલ્યોનો સંગ્રહ દર્શાવવા. અને ચોક્કસ માળખા સાથે મૂલ્યોનો સંગ્રહ ચોક્કસપણે આપણે ડેટા માળખું કહીએ છીએ. તેથી, આ વધુ સામાન્ય રીતે ડેટા સ્ટ્રક્ચર્સ કહેવાય છે. તેથી, આ કિસ્સામાં અમારી પાસે ડેટા માળખું એક યાદી છે.

(સ્લાઈડ સમયનો ઉલ્લેખ કરો: 23:56)

આ નામો અને મૂલ્યો સાથે અમે શું કરી શકીએ છીએ એક વસ્તુ એ છે કે આપણે કોઈ નામનું મૂલ્ય આપી શકીએ. તેથી, ઉદાહરણ તરીકે, જ્યારે આપણે એફએન લખીએ છીએ ખાલી ખાલી યાદીની બરાબર છે, તો આપણે ખાલી યાદી ખાલી કરવા માટે એફએન ની કિંમત સુયોજિત કરી રહ્યા છીએ. આ એમ કહીને બે વસ્તુઓ કહે છે કે મૂલ્ય empty સૂચિ છે, તેથી તે પાયથનને પણ કહે છે fn સૂચિને સૂચવે છે કે આ બે પગલા અહીં ચાલતા હોય છે. અને બીજો ભાગ એ છે કે જ્યારે આપણે cf માં દરેક એફ માટે કંઈક લખીએ છીએ, જે પ્રત્યેક રીતે કહે છે કે દરેક મૂલ્યને CF માં લે છે અને તેને એક પછી એક નામ એફ એફ પર ફાળવો. તેમ છતાં, તેમની પાસે આ સમાનતા ચિહ્ન સ્પષ્ટ રૂપે નથી તે સ્પષ્ટપણે આ માટે નવા મૂલ્યો સોંપણી કરે છે કારણ કે જેમ આપણે યાદી સીએફ જમણા કરીએ છીએ. તેથી, મુખ્ય વસ્તુ જે આપણે પાયથન પ્રોગ્રામમાં કરીએ છીએ તે નામોને વેલ્યુ આપવાનું છે.

(સ્લાઈડ ટાઈમ નો સંદર્ભ લો: 24:37)

અને મૂલ્ય અસાધન કર્યા પછી આપણે મૂલ્યને બદલી શકીએ છીએ. દાખલા તરીકે, દર વખતે આપણે એક નવો પરિબલ શોધીએ છીએ, આપણે કોઈ પણ જૂના પરિબલ દ્વારા ન માંગીએ છીએ કે આપણે હાલની યાદી એફએમ લેવા માંગીએ છીએ અને અમે તે ઉમેરવા માંગીએ છીએ. તેથી, ઉદાહરણ તરીકે આ ફંક્શન એપ્લન્સ નામ FN ની વેલ્યુને નવા નામથી સુધારે છે જે જૂના નામ લે છે અને તે ઓવરને અંતે હું એક લાકડી. વધુ સામાન્ય રીતે તમારી પાસે સંખ્યા હોઈ શકે છે અને આપણે સંખ્યાના બે ગણીને બદલી શકીએ છીએ. તેથી, આપણી પાસે કંઈક હોઈ શકે છે, જેમ કે હું બે વખત બરાબર છું. તેથી, ગુણ ગુણાકાર માટે વપરાય છે તેનો અર્થ એ નથી કે હું બે વાર અંકિત રૂપે બરાબર છું કારણ કે; દેખીતી રીતે, જ્યાં સુધી હું 0 નથી ત્યાં સુધી હું બે વખત પોતે જ ન હોઈ શકું. તેનો અર્થ એ છે કે હું વર્તમાન મૂલ્ય લેતો, તે બેથી વધવું અને તેને i ને સોંપવું. તેથી, આપણે આ પ્રમાણે જ જોઈશું, પરંતુ અસાધનમેન્ટ ક્યાં તો એક સંપૂર્ણ નવો મૂલ્ય આપી શકે છે અથવા તમે જૂના મૂલ્યનો ઉપયોગ કરીને મૂલ્ય અપડેટ કરી શકો છો. તેથી, અહીં આપણે સૂચિ fn ના કાર્યનું જૂનું મૂલ્ય લઈએ છીએ અને અમે મૂલ્યને જોડીએ છીએ જે તેને એફએનની નવી કિંમત મેળવશે.

(સ્લાઈડ ટાઈમ નો સંદર્ભ લો: 25:49)

અન્ય ભાગમાં આપણે એ નોંધવાની જરૂર છે કે આપણે કેવી રીતે પગલાંઓ ચલાવો. તેથી, અમે આજે પ્રવચનની શરૂઆતમાં જણાવ્યું હતું કે પ્રોગ્રામ એ પગલાંનો ક્રમ છે. પરંતુ આપણે ફક્ત શરૂઆતથી અંત સુધી અંધકારથી આ ક્રમમાં ક્રમ ચલાવતા નથી. ક્યારેક આપણે ફરી અને ફરી એક જ વસ્તુ કરવી પડશે. હમણાં પૂરતું આપણે દરેક શક્ય પરિબલને 1 થી મીટર સુધી તપાસવું જોઈએ જો તે મીટરને વિભાજિત કરે અને પછી તેને યાદીમાં મુકવું. તેથી, કેટલાક પગલાઓ પુનરાવર્તન કરવામાં આવે છે, ઉદાહરણ

તરીકે, યાદીમાં પ્રત્યેક આઇટમ માટે અહીં આપણે કંઈક કરીએ છીએ. અને કેટલાક પગલાંઓ એક્ઝિક્યુટ થાય છે, જો તે કિંમત જે અમે જોઈ રહ્યા છીએ તે ચોક્કસ પરિસ્થિતિઓને પૂર્ણ કરે છે. જ્યારે આપણે એમ કહીએ છીએ કે જો એમ ટકા i ની સંખ્યા 0 છે, જો બાકીનો ભાગ એ 0 દ્વારા વહેંચાય તો પછી ઉમેરો. તેથી, આ પગલું એ એક્ઝેમ સાથે જોડાયેલો છે કારણ કે હું એમની પરિબળો સાથે મેળ ખાય તો તે માત્ર મીટરનું પરિબળ છે. તેથી, અમે વારંવાર પગલાં લીધાં છે જ્યાં એક જ વસ્તુ ફરીથી અને ફરીથી થાય છે. અને તેઓ પાસે શરતો હોય છે જે કોઈ ચોક્કસ સ્થિતિ ધરાવતી હોય તો જ કરવામાં આવે છે. તો, અમે અહીં બંધ કરીશું. આ ઉદાહરણોએ તમને બતાવવું જોઈએ કે પ્રોગ્રામ્સ આપણે તર્કથી શું કરતા અલગ નથી, તે ફક્ત તેમને યોગ્ય રીતે લખવાની એક સવાલ છે, અને ખાતરી કરો કે અમે તમામ મધ્યવર્તી મૂલ્યો અને પગલાં કે જે અમે લાંબા સમય સુધી જરૂર છે તેનો સાચવી રાખીએ છીએ, તેથી કે આપણે વસ્તુઓ ગુમાવી નથી. અમે આ ઉદાહરણને વધુ વિગતમાં જોશું કારણ કે આપણે લાંબા સમય સુધી જઈએ છીએ, અને તેને લખવાની અન્ય રીતો શોધવાનો પ્રયાસ કરીએ છીએ, અને અન્ય લક્ષણોની તપાસ કરીએ છીએ, પરંતુ આવશ્યકપણે પ્રોગ્રામિંગને સમજાવવાનો આ એક સારો માર્ગ છે.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રો. માધવન મુકુંદ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 01 લેક્ચર - 02

નિષ્કપટ જીસીડી પ્રથમ વ્યાખ્યાનમાં આપણે પ્રોગ્રામિંગમાં કેટલાક મૂળભૂત ખ્યાલો રજૂ કરવાના ઉદાહરણ તરીકે જીસીડીનો ઉપયોગ કર્યો છે. અમે એ જ ઉદાહરણ જોવાનું ચાલુ રાખીશું અને જુઓ કે કેવી રીતે અમારા પ્રોગ્રામને રિફાઇન કરવું અને નવા વિચારોની શોધ કરવી.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 00:14)

અહીં જીસીડી માટે અમારું મૂળભૂત એલ્ગોરિધમ હતું, જે આપણે જણાવ્યું હતું કે કાર્યની વ્યાખ્યા નીચે વધુ કે ઓછું છે. અમે ઇનપુટ્સ મીટર અને એન માટે પરિબલોની બે સૂચિ બનાવી છે. તેથી, આપણે એફએમનું પરિબલ મીટરના પરિબલો બનાવીએ છીએ, અને n નાં પરિબલો, અને ત્યારબાદ આપણે cf ને બંને લિસ્ટ અથવા સામાન્ય પરિબલોમાં પરિબલોની સૂચિ ગણતરી કરીએ છીએ. અમારું ધ્યેય આ સામાન્ય સૂચિમાં સૌથી વધુ સામાન્ય સંખ્યામાં સૌથી વધુ સામાન્ય ભાગાકાર પરત કરવાની છે, જે આ સૂચિમાં છેલ્લું છે, કારણ કે અમે આ પરિબલો ચઢતા ક્રમમાં ઉમેરીએ છીએ.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 00:51)

તો શું આપણે આના કરતાં વધુ સારી રીતે કરી શકીએ? જે રીતે આપણે આગળ વધ્યું છે, આપણે સૌ પ્રથમ સંખ્યામાં 1 થી મીટર સુધી એમના પરિબલોની યાદી એફએમના ગણતરી માટે સ્કેન કરીએ છીએ, અને ત્યારબાદ આપણે ફરીથી એફએમની ગણતરી કરવા માટે 1 થી n ની શરૂઆત કરીએ છીએ. તેથી, એક સ્પષ્ટ સુધારો માત્ર 1 થી મોટી સંખ્યામાં m અને n સુધીની સ્કેન કરે છે અને એક સ્કેન ગણતરી યાદી એફએમ અને એફએન આ યાદીમાં દરેક માટે હું 1 ની મહત્તમ મીટર અને n ની મહત્તમ સંખ્યામાં પ્રથમ તપાસ કરું છું જો હું મીટર વિભાજિત કરું તો, જો આપણે તેને m ના પરિબલોની યાદીમાં ઉમેરીએ, અને પછી અમે તપાસીએ છીએ કે હું n ને વિભાજિત કરું છું અને જો આપણે આમ કરીએ તો તે fin યાદીમાં ઉમેરો. 1 થી મીટર પર બે અલગ સ્કેન કરવા અને પછી 1 થી n અને ભૂતકાળને પુનરાવર્તન કરવાને બદલે અમે તેને એક સ્કેનમાં કરીએ છીએ.

(સ્લાઇડ સમય જણાવો: 01:46)

પણ આમાં પણ સુધારો થઈ શકે છે જો આપણે તેને એક પાસમાં કરી રહ્યા છીએ અને અમે તપાસ કરી રહ્યા છીએ કે નંબરો બંને - M અને n ને ભાગાકાર કરે છે, તો પછી આપણે સીધી જ સામાન્ય પરિબલો કેમ નથી તપાસીએ? બીજી સૂચિમાં બે યાદીઓની ગણતરી કરવાને બદલે અને પછી સંયોજન કરીને આપણે સીધી રીતે નીચે પ્રમાણે કરી શકીએ છીએ: દરેક માટે હું 1 થી મહત્તમ મહત્તમ m અને n , જો હું એમ અને n બંનેને વિભાજિત કરું તો આપણે સીધું જ સામાન્ય યાદીમાં ઉમેરીએ છીએ. પરિબલો જો તે n તો વિભાજન કરે છે અથવા જો તે માત્ર એક જ વિભાજન કરે છે તો તે એક સામાન્ય પરિબલ નથી અને અમે કાઢી નાખી શકો છો. વાસ્તવમાં, નોંધવું જોઈએ કે મહત્તમ મીટર અને એન જવા કરતાં આપણે લઘુત્તમ મીટર અને એન પર જવું જોઈએ, કારણ કે એક વખત અમે નાની સંખ્યાને પાર કરીશું તો અમને નાની સંખ્યા માટે પરિબલ મળશે નહીં. યાદ રાખો કે m અને 1 અને n વચ્ચેના m ની પરિબલો

અને 1 અને n વચ્ચે આવેલા છે. જો મીટર એ n કરતા નાના હોય છે, ઉદાહરણ તરીકે, જો આપણે ઇનપુટ મીટર વત્તા 1 જો તે n નો પરિબળ બનાવતો હોય તો તે ચોક્કસપણે મીટરનો પરિબળ ન હોઈ શકે. તેથી અમારી વધુ સારી રણનીતિ શ્રેણી 1 માં ઓછામાં ઓછા મીટર, અને n માં દરેક 1 માટે છે, જો હું એમ અને n બંનેને વિભાજીત કરું તો અમે સામાન્ય પરિબળોની સૂચિમાં ઉમેરો.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 03:03)

આ નવી વ્યૂહરચનાના અમલીકરણમાં ખૂબ ટૂંકા પાયથોન પ્રોગ્રામ છે. તેથી યાદીઓ fin અને fin ની ગણતરી કરવાને બદલે આપણે સામાન્ય પરિબળોની સૂચિની ગણતરી કરીએ છીએ. અમે હું 1 થી લઇને લઘુત્તમ મીટર મીટર અને n સુધીની શ્રેણીને યાદ કરીએ છીએ અને યાદ રાખવું છે કે પાયથોનની શ્રેણીની મર્યાદા આપવાની જરૂર છે, જે મર્યાદાથી ઉપર છે, તેથી અમે 1 થી લઘુત્તમ એમએન સુધી જઈએ છીએ. વત્તા 1. અને હવે અમારી પાસે એક વધારાની જોડાયેલી છે જેને લોજિકલ કનેક્ટીવ કહેવાય છે અને જે કહે છે કે આપણે બે શરતો સાબિત કરવા માંગીએ છીએ, અમે બાકીની માંગો છો જ્યારે મીટર 0 વડે ભાગવામાં આવે છે, તો બીજી શબ્દોમાં, હું મીટર વિભાજીત કરું છું અને અમે પણ બાકીની માંગો છો જ્યારે n એ 0 દ્વારા વહેંચાયેલો છે, તેથી મને બંને એમ અને n ને વહેંચવું જોઈએ અને જો આપણે સામાન્ય પરિબળોની સૂચિમાં ઉમેરો તો. અને ફરી એકવાર આમ કર્યા પછી આપણે ચડતા ક્રમમાં તે કરી રહ્યા છીએ, તેથી સામાન્ય પરિબળો ઉમેરવામાં આવે છે કારણ કે આપણે મોટા લોકો સાથે આગળ વધીએ છીએ. તેથી, આપણે છેવટે છેલ્લો એલિમેન્ટ જોઈએ છે કે જે પાયથોનને સીએફની બાદબાકી 1 થી ઘટક તરીકે આપવામાં આવે છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 04:15)

તેથી આ કરવાથી, કદાચ આપણે વસ્તુઓને વધુ સરળ બનાવી શકીએ. શું આપણે સામાન્ય પરિબળોની સૂચિની જરૂર છે? યાદ રાખો કે અમને ફક્ત સૌથી સામાન્ય સામાન્ય પરિબળની જરૂર છે. અમે જોયું કે હંમેશા ઓછામાં ઓછી એક સામાન્ય પરિબળ હશે 1. તેથી, સૌથી સામાન્ય પરિબળની કલ્પના હંમેશાં કોઈપણ જોડી મીટર અને એન માટે સારી રીતે વ્યાખ્યાયિત રહેશે. દરેક વખતે અમે 1 થી શરૂ કરી શકીએ અને દરેક વખતે આપણે મોટા સામાન્ય પરિબળને શોધી શકીએ છીએ જે આપણે પહેલાના એકને કાઢી નાખી શકીએ છીએ, અમને બધા સામાન્ય પરિબળોને યાદ રાખવાની જરૂર નથી કે અમને ફક્ત સૌથી મોટી જરૂર છે. તેથી આ અમારી વ્યૂહરચનાને મોટા પ્રમાણમાં સરળ બનાવી શકે છે કારણ કે આ સૂચિમાં દરેક સંભવિત પરિબળની યાદી રાખવાની જરૂર નથી; આપણે જે સૌથી મોટું એક જોયું છે તેને જાળવી રાખવું જોઈએ. અમે સૌથી તાજેતરનાં સામાન્ય પરિબળ માટે એમઆરસીએફ નામના નામનો ઉપયોગ કરી શકીએ છીએ, અને આ નામને સામાન્ય પરિબળના મૂલ્ય સાથે અપડેટ કરીએ છીએ જે આપણે છેલ્લે જોયું.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 05:17)

તો અહીં આ વિચારનું પાયથોન અમલીકરણ છે જ્યાં આપણી પાસે કોઈ પણ સૂચિ નથી. અમે સીધા જ તમામ સંભવિત પરિબળોને 1 થી લઘુત્તમ m અને n પર સ્કેન કરીએ છીએ. જ્યારે પણ આપણે એક સામાન્ય પરિબળ શોધીએ છીએ ત્યારે અમે અમારા નામ એમઆરસીએફના મૂલ્યને વર્તમાન સામાન્ય પરિબળ તરીકે અપડેટ કરીએ છીએ જે અમને મળ્યું છે. હવે યાદ રાખો કે 1 હંમેશાં એક સામાન્ય પરિબળ હશે, તેથી શરૂઆતમાં એમઆરસીએફને મૂલ્ય 1 આપવામાં આવશે, તે નહીં થાય કે આપણે આ વારંવાર અમલ કરીને પસાર થવું જોઈએ અને ક્યારેય એમઆરસીએફ ન આપશો કારણ કે શરત ક્યારેય સાચી નથી. એક સામાન્ય પરિબળ હોવાથી આપણે ઓછામાં ઓછા 1 ની બરાબર એમઆરસીએફ ધરાવું જોઈએ, પરંતુ જો આપણે મોટા સામાન્ય પરિબળ શોધીએ તો એક પછીના સામાન્ય પરિબળ દ્વારા બદલવામાં આવશે. પુનરાવર્તન અથવા લૂપ માટે આ સમાપ્ત થાય તે પછી

આપણે 1 અને લઘુત્તમ એમ અને n વચ્ચે સૌથી મોટો સામાન્ય પરિબળ છે જે ખરેખર એમ અને એન ના જીસીડી છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 06:18)

આપણે હજી પણ કેટલીક વધુ ઓપ્ટિમાઇઝેશન કરી શકીએ છીએ. ત્યારથી, અમે સૌથી મોટું સામાન્ય પરિબળ શોધી રહ્યા છીએ, શા માટે આપણે શરૂઆતમાં શરૂ કરીએ છીએ જે અમને સૌથી નાનું સામાન્ય પરિબળ આપશે. તેથી, અમે સૂચિના અંતમાં શરૂ કરી શકીએ છીએ અને પાછળની તરફ કામ કરી શકીએ છીએ. 1 થી લઘુત્તમ મીટર અને n નીચલા ભાગને બદલે અમે લઘુત્તમ મીટર અને n થી શરૂ કરી શકીએ છીએ અને પાછળથી 1 તરફ કાર્ય કરી શકીએ છીએ. ફરીથી બાંધકામ એ છે કે 1 હંમેશા એક સામાન્ય પરિબળ તરીકે બતાવવામાં આવશે, તેથી જો કોઈ અન્ય ન હોય ખૂબ જ અંતમાં સામાન્ય પરિબળો આપણે 1 સૌથી મહાન સામાન્ય પરિબળ તરીકે મેળવશો. અને આ પ્રક્રિયામાં આપણે કોઈ પણ સામાન્ય પરિબળની પાછળ જવાની જરૂર નથી, જો આપણે પાછળથી મોટાથી સૌથી નાનો સૌથી નાનો પ્રથમ સામાન્ય પરિબળ શોધીએ છીએ તો તે હકીકતમાં એમ અને એન ના જીસીડી હોવો જોઈએ.

(સ્લાઇડ સમય જણાવો: 07:09)

આપણે આને પાચથોનમાં કેવી રીતે લખીશું? ઠીક છે, તમે રેન્જમાં હું તે માટે સંશોધિત કરી શકો છો, તેથી નોંધ લો કે સામાન્ય રીતે આ ફંક્શન નાની વેલ્યુથી વધુ મોટું મૂલ્ય સુધી જાય છે, તમે તેના બદલે પાછળની જગ્યાએ તેને સંશોધિત કરી શકો છો. પરંતુ આમ કરવાને બદલે આપણે તે જોઈશું કે જ્યારે આપણે વાસ્તવમાં ઔપચારિક પાચથોનમાં પ્રવેશીશું ત્યારે ચાલો આપણે મૂલ્યોની સૂચિમાંથી પસાર થવા માટે એક નવી રીત શોધીએ. અમે ઇન્ડેક્સ i ને સોપણી દ્વારા શરૂ કરીએ છીએ, તે મૂલ્ય જે આપણે લઘુત્તમ m અને n સાથે શરૂ કરવા માગીએ છીએ, યાદ રાખો કે હું મારા માટે સૌથી વધુ શક્ય મૂલ્યથી પ્રારંભ કરવા માંગું છું અને પાછળની તરફ જઈશ. તેથી આપણી પાસે શું છે, જ્યારે કહેવામાં આવે છે તેના બદલે એક નવો શબ્દ છે, તેથી જ્યારે અંગ્રેજી શબ્દ સૂચવે છે કે શરત સાચી છે ત્યારે થાય છે. તેથી, જ્યારે હું જે શોધી રહ્યો છું તે હકારાત્મક છે. તેથી જ્યારે હું 0 કરતા મોટો છે, આપણે શું કરીએ? અમે તપાસ કરીએ છીએ કે હું એક સામાન્ય પરિબળ છે. આ પહેલાની જેમ જ છે, આપણે તપાસીએ કે હું મીટર વિભાજીત કરું છું અને હું પણ n ને વિભાજીત કરું છું. જો આપણે સામાન્ય પરિબળ શોધીએ છીએ, તો અમે ફક્ત તે જ મૂલ્ય પાછીએ છીએ જે આપણે શોધી કાઢ્યું છે અને અમે આ વિધેયમાંથી પૂર્ણ રૂપે બહાર નીકળીએ છીએ. દરેક વખતે જ્યારે તમે ફંક્શનમાં રીટર્ન સ્ટેટમેન્ટ જુઓ છો, ત્યારે ફંક્શન બંધ થાય છે અને ફંક્શનમાં વેલ્યુ એ ફંક્શન આપણને પરત આપે છે. તેથી અમે લઘુત્તમ મીટર અને n સાથે બરાબર શરૂ કરીએ છીએ અને હું તપાસ કરું છું કે શું હું એક સામાન્ય પરિબળ છે કે કેમ તે છે, તેથી અમે બહાર નીકળીએ છીએ અને છેલ્લે અમે મળેલું મૂલ્ય પરત કરીએ છીએ. અને આ જ એકમાત્ર કિંમત છે જે આપણને જરૂર છે, અમને કોઈ અન્ય સામાન્ય પરિબળોની જરૂર નથી. તેથી, આપણે પહેલી વાર એક સામાન્ય પરિબળ જોઈને પાછા ફરો. બીજો બાજુ, જો હું એક સામાન્ય પરિબળ ન હોઉં તો આગળ વધવા માટે આગળના એકને ચેક કરીને આગળ વધવાની જરૂર છે અને આ સુધારા દ્વારા હાંસલ કરવામાં આવે છે. તેથી, યાદ રાખો કે અમે આ સમાનતા ઓપરેશનનો ઉપયોગ કરીને મૂલ્યો અથવા અપડેટ મૂલ્યો સોંપી શકીએ છીએ. આ સમાનતા ઓપરેશન ગાણિતિક સમાનતા નથી કારણ કે તે દેખાય છે, પરંતુ તે મૂલ્યનું સોંપણી છે. તે કહે છે, મારી જૂની કિંમત લઈ અને તેને નવું મૂલ્ય બનાવવું. તેથી તે કહે છે કે, હું માઈનસ 1 માં સુધારો કરું છું, હું વર્તમાન મૂલ્ય લઈશ, બાદબાકી કરું છું અને તેને બદલું છું. ગાણિતિક સમાનતાને ડબલ સમાન તરીકે લખવામાં આવે છે. આ અમારી પરિસ્થિતિઓમાં આપણે જે કરીએ છીએ તે જ છે. તેથી, આ યાદ રાખવું અગત્યનું છે કે ડાબેરી સમાન સમાનતા, ડાબી બાજુની બાજુ જમણી બાજુની બરાબર છે, જ્યારે પાચથોનમાં સિંગલ સમાનતા અને અન્ય ઘણા પ્રોગ્રામિંગ લેંગ્વેજનો અર્થ વેરિએબલને વેલ્યુ આપવાનું છે. તેથી, આ અંતિમ ઓપ્ટિમાઇઝેશન છે કે જે આ નિષ્કપરી

અલ્ગોરિધમનો છે જે મૂળભૂત પરિબલોને શરૂઆતથી અંત સુધી સ્કેન કરવા માટે છે. તેથી હવે આપણે અંતથી તે શરૂઆતમાં જ કરી રહ્યા છીએ અને માત્ર પ્રથમ પરિબલ જે અમે શોધીએ છીએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 10:03)

આપણે આ ઉદાહરણમાં જે જોયું તે એક નવી પ્રકારનું લૂપ છે. તેથી, આ નવી પ્રકારનું લૂપ એક ખાસ શબ્દ ધરાવે છે જ્યારે અને જ્યારે સ્થિતિ સાથે સાથે છે, જ્યાં સુધી સ્થિતિ સાચી છે ત્યાં સુધી, અમે તે સમયના શરીરની અંદર જે કરીએ છીએ તે સાચું છે. હવે નોંધ લો કે પાયથોન indentation નો ઉપયોગ કરે છે, તેથી આ નિવેદનો આ સમયના સંદર્ભમાં સરભર થાય છે. આ કેવી રીતે પાયથોન જાણે છે કે પગલાં 1 થી k આ સંબંધમાં છે. તેથી, આ પગલાઓ એ છે કે આ વસ્તુના અંતે પુનરાવર્તિત થવું આવશ્યક છે, તમે પાછા આવો છો અને તમે તપાસ કરો કે આ સ્થિતિ હજુ પણ સાચી છે, જો તે સાચું હોય તો તમે તે એક વધુ સમય અને તેથી વધુ કરો છો. તેથી, જ્યારે ઉપયોગી છે જ્યારે અમને અગાઉથી ખબર નથી કે કેટલા પગલાંઓ આપણે પુનરાવર્તન કરીશું. જ્યારે આપણે પરિબલોની સૂચિ માટે સ્કેન કરી રહ્યા હતા ત્યારે અમે જાણતા હતા કે અમે એક સાથે શરૂ કરીશું અને લઘુત્તમ મીટર અને એન સુધી જઈશું. અમે અગાઉથી આગાહી કરી શકીએ છીએ કે અમે ઘણા પગલાંઓ બરાબર કરવા જોઈએ અને તેથી અમે લૂપ માટે આનો ઉપયોગ કરી શકીએ છીએ, તેથી લૂપની ચોક્કસ સંખ્યા પુનરાવર્તિત હોય છે. બીજી બાજુ, જ્યારે લૂપ સામાન્ય રીતે ઉપયોગમાં લેવાતો હોય ત્યારે જ્યારે તમે સ્ટોપ થતા હોય ત્યારે અગાઉથી જાણતા નથી. તેથી આ કિસ્સામાં આપણે લઘુત્તમ મીટર અને એન સાથે શરૂ કરવા જઈએ છીએ, પાછળની બાજુએ કામ કરીએ છીએ અને જલદી અમે આ પરિબલ શોધી શકીએ છીએ, પરંતુ અગાઉથી કોઈ ખ્યાલ નથી કે આ પ્રારંભિક રીતે આવશે અથવા પાછા બધી રીતે જવું પડશે. જે આપણે જાણીએ છીએ તે માન્ય પરિબલ હોવાની ખાતરી આપી છે. એક જ્યારે સમસ્યાઓનો સામનો કરી શકે છે તે એક છે કે આપણે પાછા આવવાનું ચાલુ રાખવું જોઈએ અને શોધવું જોઈએ કે આ સ્થિતિ સાચું છે. તેથી અમે આ સમયમાંથી આગળ વધ્યા નહીં. તેથી, જ્યાં સુધી સ્થિતિ સાચી છે ત્યાં સુધી આ પગલાંઓ ચલાવવામાં આવશે અને પછી તમે પાછા જશો અને તે ફરીથી કરશો. જો તમે કંઈક બદલાયેલ નથી જે કંઈક ખોટી બનાવે છે તો તમે કદી બહાર નહીં જાઓ.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 11:51)

અમારા અગાઉના ઉદાહરણમાં, શરત ખોટી બનાવવા માટે, અમારે 0 બનવાની જરૂર છે. તેથી અમે લઘુત્તમ મીટર અને n થી શરૂ કરીએ છીએ. તેથી, આપણે શું બાંધધરી આપીએ છીએ કે દર વખતે જ્યારે આપણે આ પસાર કરીએ છીએ અને અમે સમાપ્ત નથી કરતા, આપણે શું કરવા માગીએ છીએ અમે 1 ને ઘટાડીએ છીએ, અને તેથી આપણે અમુક ચોક્કસ મૂલ્યથી શરૂ કરીએ છીએ અને અમે 1 દ્વારા હું ઘટાડીએ છીએ પછી આપણે 0. સામાન્ય રીતે જ્યારે તમે જ્યારે લૂપનો ઉપયોગ કરો છો ત્યારે તમારે ખાતરી કરવી જ જોઈએ કે તમે લૂપને સમાપ્ત કરવા તરફ પ્રગતિ કરી રહ્યા છો, નહીં તો તમારી પાસે એક અનંત લૂપ કહેવાય ખતરનાક પ્રકારનું વર્તન છે જ્યાં ગણતરી માત્ર મૂલ્ય પરત કર્યા વગર ચાલુ રહે છે અને તમને એનો કોઈ ખ્યાલ નથી કે તે જવાબની ગણતરી કરવા માટે ઘણો સમય લાવ્યો છે કે પછી તે સમાપ્ત થવાનો નથી.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 12:35)

તો આ પ્રવચનમાં આપણે શું જોયું છે તે આપણે ખૂબ નિષ્કપટ ખ્યાલથી શરૂ કરી શકીએ છીએ જે કાર્યને અમલમાં મૂકવા જેટલું ઓછું કામ કરે છે અને તે અલ્ગોરિધમનો નાટ્યાત્મક રીતે સરળ બનાવવાના અમારા રસ્તાઓનું કાર્ય કરે છે. હવે કોમ્પ્યુટેશનલ બિંદુ પરથી એક વસ્તુ એ છે કે નવી આવૃત્તિઓ પ્રોગ્રામ માટે સરળ છે અને તેથી તે સમજવા માટે, તેઓ જે સમય લે છે તે ખૂબ અલગ નથી. અમે હજી પણ મૂળભૂત રીતે તમામ મૂલ્યો દ્વારા સિક્કાંતમાં 1 થી લઘુત્તમ m અને n સુધી ચાલી રહ્યા છીએ. જો

આપણે શરૂઆતથી શરૂ કરીએ તો આપણે આ બધી કિંમતો દ્વારા ચલાવીશું કારણ કે આપણે સામાન્ય પરિબળો શોધવા માટે આ બધા નંબરોને સ્કેન કરીએ છીએ. છેલ્લા સંસ્કરણમાં, જ્યાં અમે પાછળની બાજુએ કામ કરવાનો પ્રયાસ કરી રહ્યા હતા અને પ્રથમ સામાન્ય પરિબળને રોકવા પ્રયાસ કરી રહ્યા હતા તે હજુ પણ બની શકે છે કે બે નંબરો 1 કરતાં અન્ય કોઈ સામાન્ય પરિબળ નથી. તેથી ફરી, અમે બધી રીતે ઓછામાં ઓછા મીટર જવાબ આપતાં પહેલાં 1 થી પાછા આવો. તેમ છતાં, પ્રોગ્રામ્સ સરળ કોમ્પ્યુટેશનલ રીતે દેખાય છે, તે બધા લગભગ સમાન છે અને તે મૂલ્યો m અને n ને પ્રમાણસર સમય લે છે. આગળના વ્યાખ્યાનમાં આપણે શું જોશું, તે છે કે જીસીડીની ગણતરી કરવા માટે આપણે ખરેખર એક નાટ્યાત્મક રીતે અલગ રીતે આવી શકીએ છીએ, જે વધુ કાર્યક્ષમ હશે.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રોફેસર. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટીટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 01 લેક્ચર - 03

ઇક્વિલિડના એલ્ગોરિધમ જીસીડી માટે ચાલો પ્રોગ્રામ સાથે સંકળાયેલાં વધુ મુદ્દાઓની શોધ માટે જી.સી.ડી. .

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 00:09)

આપણે જીસીડીની મૂળભૂત વ્યાખ્યાથી શરૂઆત કરી, જેમાં જણાવ્યું હતું કે આપણે સૌ પ્રથમ મીટરના તમામ પરિબલોની ગણતરી કરીશું, તેને યાદીમાં સંગ્રહિત કરીશું, n ના બધા પરિબલોની ગણતરી કરીશું, તેને બીજી સૂચિમાં સંગ્રહિત કરીશું, આ બે યાદીઓમાંથી, સામાન્ય પરિબલોની સૂચિ બહાર કાઢો અને આ સામાન્ય પરિબલ યાદીમાં સૌથી વધુ એક અહેવાલ આપો. અમારી પ્રથમ સરળીકરણ એ છે કે આપણે વાસ્તવમાં એક પાસને 1 થી લઘુત્તમ m અને n સુધી કરી શકીએ છીએ અને સામાન્ય પરિબલોની સૂચિની સીધી ગણતરી કરી શકીએ છીએ. અમે પછી અવલોકન કરીએ છીએ કે અમને સામાન્ય પરિબલોની આ સૂચિની જરૂર નથી કારણ કે આપણી રૂચિ માત્ર મહાન સામાન્ય પરિબલ અથવા શ્રેષ્ઠ સામાન્ય ભાજક છે. આથી, આપણે એક જ નામમાં અત્યાર સુધીમાં જે સૌથી સામાન્ય પરિબલ જોયું છે તેનું ધ્યાન રાખીશું અને અંતે તેને જાણ કરીશું. અમારું અંતિમ સુચન એ જોવાનું હતું કે જો આપણે સૌથી સામાન્ય બાબતમાં રસ ધરાવીએ છીએ, તો આપણે શરૂઆતથી શરૂ કરીશું અને પ્રારંભની શરૂઆતથી કરીશું નહીં તેથી, 1 થી શરૂ કરીને અને ઓછામાં ઓછા મીટર અને n ની ઉપર લઘુત્તમ અને ઉપરની તરફ કામ કરવાને બદલે, એક પછી એક અને પાછળની બાજુએ કામ કરો અને જલદી અમે એક સામાન્ય પરિબલ શોધીએ છીએ, તો અમે તેની જાણ કરીએ છીએ અને બહાર નીકળીએ છીએ. યાદ રાખો કે 1 સામાન્ય પરિબલ બનવાની ખાતરી આપી છે. તેથી જ્યારે આપણે લઘુત્તમ મીટર અને n થી શરૂ કરીએ અને પાછળની તરફ જઈએ, જો આપણે કોઈ અન્ય સામાન્ય પરિબલ ન જોશું, તો અમે હજી પણ ખાતરી આપીએ છીએ કે જ્યારે આપણે એકને હિટ કરીશું ત્યારે યોગ્ય રીતે બહાર નીકળીશું. તેથી આપણે શું નોંધ્યું છે કે, આ વિવિધ આવૃત્તિઓ અગાઉની આવૃત્તિઓ કરતાં સરળ છે, છતાં બધાની ગણતરીની દ્રષ્ટિએ સમાન કાર્યક્ષમતા છે, એટલે કે તે આપણને સૌથી ખરાબ કિસ્સામાં 1 અને વચ્ચેની બધી સંખ્યાઓ દ્વારા ચલાવવા માટે દબાણ કરશે. લઘુત્તમ મીટર અને એન, અમે આગળ અથવા પાછળની બાજુએ કામ કરીએ છીએ કે કેમ તે મહાન સામાન્ય પરિબલ મળે તે પહેલાં.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 02:08)

તેથી પ્રાચીન ગ્રીક સમયે, યુક્લિડ દ્વારા સંભવતઃ મોડેમ પરિભાષામાં પ્રથમ એલ્ગોરિધમનો શોધ કરવામાં આવી હતી, અને તે આ સમસ્યા માટે હતી - જીસીડી તેથી યુક્લિડ શું કહે છે તે નીચે મુજબ છે. ધારો કે આપણી પાસે એક વિભાજક છે જે m અને n બંનેને વિભાજીત કરે છે, તેથી આ એક સામાન્ય ભાજક છે અને અમે સૌથી મોટું ડી. ચાલો દલીલના ખાતર ધારીએ કે મીટર n કરતા વધારે છે. તેથી જો ડી બંને એમ અને એન ને વિભાજીત કરે છે, તો આપણે m એ એક વખત અને t તરીકે લખી શકીએ છીએ જેમ કે b અને t કેટલાક મૂલ્યો a અને b માટે છે, તેથી m એ t ની બહુવિધ છે અને તેથી n એ છે. તેથી જો આપણે સમીકરણોને બાદ કરીએ તો ડાબી બાજુ M એ મીટર n એ છે. તેથી, અમે મીટર લઈએ છીએ અને m માંથી n બાદ કરીએ છીએ, તેથી સંલગ્ન રીતે આપણે t માંથી બીડીને બાદ કરીએ છીએ. તેથી, મીટર બાદ n એ એડ માઇનસ બી.ડી.ની બરાબર છે, પરંતુ ત્યારથી ડી સામાન્ય શબ્દ છે, એટલે કે m

નોન n એ માઈનસ b વખત d છે. આ એવી જગ્યા છે જ્યાં અમે એ ધારણાનો ઉપયોગ કરી રહ્યા છીએ કે m એ n કરતા વધારે છે, તેથી ઓછા બો એ સકારાત્મક સંખ્યા હશે. પરંતુ મહત્વની બાબત એ છે કે m minus n એ d ની બહુવિધ છે. બીજા શબ્દોમાં કહીએ તો, જો d એમ અને એન એમ બંનેને વિભાજિત કરે છે, તો તે પણ m ઓછા એન ને વહેંચે છે. અને ત્યારથી d એ m અને n ના સૌથી મોટા ભાજક છે, તે ચાલુ થશે કે d એ સૌથી મોટો ભાજક છે જે m, n અને m બાદબાકી માટે સામાન્ય છે. બીજા શબ્દોમાં કહીએ તો, m અને n ના જીસીડી એ બે ના નાનકડા જીસીડી જેવા જ છે, એટલે કે n અને બે મીટર અને n નો તફાવત, m minus n . તેથી, આપણે તેનો ઉપયોગ જીસીડી શોધવાની પ્રક્રિયાને સરળ બનાવવા માટે કરી શકીએ છીએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 04:00)

તો અહીં યુક્લિડના અલ્ગોરિધમનો પ્રથમ વર્ઝન છે. તેથી, મૂલ્યનો વિચાર કરો: mc ના જીસીડી એમ ધારી રહ્યા છીએ કે મીટર n કરતા વધારે છે. તેથી જો n એ પહેલેથી જ એમનું ભાગાકાર છે, તો આપણે પૂર્ણ કર્યું છે અને આપણે n પરત કરીશું. નહિતર, અમે સમસ્યાને નવામાં રૂપાંતરિત કરીએ છીએ અને m અને n ના gcd ને ગણતરીમાં લેવાની બદલે, જે આપણે પ્રારંભ કર્યું છે, અમે n અને m બાદના જીસીડીની ગણતરી કરીએ છીએ અને તેના બદલે તે મૂલ્ય પરત કરીએ છીએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 04:32)

તો, અહીં આ વિચારનો પાયથોન અમલ છે. અહીં રજૂ કરાયેલી કેટલીક નવી સુવિધાઓ છે, તો ચાલો આપણે તેમને જુઓ. પ્રથમ આ વિશિષ્ટ નિવેદન છે જે પ્રતીક હેશથી પ્રારંભ થાય છે. તેથી પાયથોન માં, આ પ્રકારની નિવેદનને એક ટિપ્પણી કહેવામાં આવે છે. તેથી ટિપ્પણી એ એક નિવેદન છે કે તમે પ્રોગ્રામમાં વાંચતા કોઈ વ્યક્તિને શું કરવું તે સમજાવવા માટે પ્રોગ્રામમાં મૂકવામાં આવે છે, પરંતુ પ્રોગ્રામ ચલાવતા કમ્પ્યુટર દ્વારા તેને અવગણવામાં આવે છે. તેથી, આ વિધાન કહે છે કે અમે એમ ધારી રહ્યા છીએ કે મીટર n કરતા વધારે અથવા તેના બરાબર છે. તેથી, આ આપણને કહે છે કે જ્યારે કાર્યક્રમ ચાલુ રહે છે ત્યારે તે ધારણા છે. અલબત્ત, તે શક્ય છે કે જે વ્યક્તિ જીસીડીને આમંત્રણ આપે છે તે આનો ખ્યાલ નથી કરતો, તેથી તે n કરતાં નાના મીટર સાથે તેનો ઉપયોગ કરી શકે છે અને તેથી અમે તેને ઠીક કરીએ છીએ. આ અસાઇનમેન્ટની વિશિષ્ટ પ્રકારની છે જે પાયથોન ને વિશિષ્ટ છે; તે મોટા ભાગની અન્ય પ્રોગ્રામિંગ ભાષાઓમાં હાજર નથી. તેથી આપણે શું કરવા માંગીએ છીએ, મૂળભૂત રીતે આપણે મૂલ્યો m અને n લેવા માંગીએ છીએ અને અમે તેમને બદલાવવા માંગીએ છીએ, જમણે. અમે મીટરનું નવું મૂલ્ય, n નું જૂનું મૂલ્ય અને n નું નવું મૂલ્ય, મીટરનું જૂનું મૂલ્ય બનાવવું છે, જેથી મીટર અને n એ ખોટા ક્રમમાં હોય તો અમે તેમને ઉલટાવીએ છીએ. તેથી, આ પાયથોન સ્ટેટમેન્ટ શું કરે છે તે મૂલ્યોની જોડી લે છે અને તે એકસાથે સોંપણી કરે છે તેથી તે કહે છે કે n ની વેલ્યુ મીટરની કિંમતમાં જાય છે અને m ની વેલ્યુ n ની વેલ્યુ જાય છે. હવે તે અગત્યનું છે કે તે એક સાથે છે, કારણ કે જો તમે તેને કોઈ ક્રમમાં કરો છો, જો તમે પ્રથમ n માં મૂલ્યની નકલ કરો છો, તો પછી n નું જૂનું મૂલ્ય ખોવાયું છે. તેથી, તમે મીટરના જૂના મૂલ્યને n ના નવા મૂલ્યમાં નકલ કરી શકતા નથી કારણ કે તમે તેને ગુમાવ્યું છે. તેથી કલ્પના કરો કે તમારી પાસે બે મગ પાણી છે, અને હવે તમે તેમની સામગ્રીઓનું વિનિમય માગો છો. હવે તમારે જગ્યા બનાવવી પડશે, તેમાંથી છુટકારો મેળવ્યા વગર તમે તેમાં તે રેડી શકતા નથી અને એકવાર તમે તેમાંથી છુટકારો મેળવી શકો છો જેથી તમે તેમાં રેડી શકતા નથી, તેથી તમારે સામાન્ય રીતે ત્રીજા મોઢુંની જરૂર છે. તમારે પ્રથમ અહીં સ્થાનાંતરિત કરવું અને તેને સુરક્ષિત રાખવું જોઈએ, અને પછી તમારે તેને અહીં સ્થાનાંતરિત કરવાની જરૂર છે અને પછી તમારે તેને કોપી કરવાની જરૂર છે. તેથી આ સામાન્ય રીત છે કે મોટાભાગની પ્રોગ્રામિંગ લેંગ્વેજ તમને બે મૂલ્યોનું વિનિમય કરવા માટે કહેશે, પરંતુ પાયથોન ને આ નિફ્ટી સુવિધા છે, જેના દ્વારા તમે મૂલ્યોની એક જોડી લઈ શકો છો અને સાથે સાથે તેને અપડેટ કરી શકો છો અને ખાસ કરીને આ એક સાથે અપડેટથી આપણને મૂલ્યોનું વિનિમય કરવાની પરવાનગી મળે છે. એક કિંમત પાર્ક કરવા માટે આ વધારાની અસ્થાયી જગ્યા હોવા

અંગે ચિંતા. કોઈપણ રીતે, આ પ્રથમ ભાગ જે કરી રહ્યો છે તે બધું એ સુનિશ્ચિત કરવાનું છે કે આ શરત જે અમે ધારી છે તે સંતોષ છે. તેથી હવે અમે અલ્ગોરિધમનો જડ પર આવીએ છીએ. જો એમ એન ને વિભાજીત કરે છે, જે એન નું ભાગ્યા n ની બરાબર છે તો પછી આપણે n એ gcd હોવું જોઈએ અને આપણે n પરત કરીશું. જો આ કિસ્સો ન હોય, તો આપણે છેલ્લી સ્લાઇડમાં જે શોધ્યું તે પછી આપણે પાછા જઈએ છીએ અને હવે આપણે જુસીડીના n અને તફાવત M બાદબાજ n ની ગણતરી કરીશું. અમે n અને m બાદ n ના gcd ને આદર્શ ગણવા માંગીએ છીએ. તેથી, આપણે તફાવત M બાદબાજી n ની ગણતરી કરીએ છીએ અને આપણે આનો ઉપયોગ કરી શકીએ છીએ. પરંતુ, તે શક્ય છે, ઉદાહરણ તરીકે - જો એમ કહે છે 97 અને n એ 2 છે તો તફાવત 9 95 હશે. તફાવત એ ખૂબ સરસ રીતે n કરતા વધારે હોઈ શકે છે, અને અમે આ ફંક્શનને આ પ્રથમ નંબર કરતાં મોટી સંખ્યા સાથે કોલ કરવા માંગીએ છીએ. મોટી સંખ્યા. તો આપણે આ ખાતરી કરીશું પણ ભલે અમારી ફંક્શન આની કાળજી લે. આપણે શું કરવા માંગીએ છીએ, આપણે જુસીડીને n અને m બાદના n ને કોલ કરવા માંગીએ છીએ, આપણે જુસીડીને મહત્તમ મૂલ્ય અને પ્રથમ દલીલ અને ન્યૂનતમ મૂલ્ય n અને તફાવત તરીકેના તફાવત સાથે કોલ કરીશું. તેથી તે ખાતરી કરશે કે બે મૂલ્યોનું મોટું ધોરણ પ્રથમ જાય છે અને બે મૂલ્યોનું નાનું સ્થાન જાય છે. અને આ જુસીડી, જે નવું જુસીડી આપે છે તે આ ફંક્શન પરત આપશે. આ પછી આપણે શું જોશું તે એક ઉદાહરણ છે, જે તદ્દન સ્વાભાવિક છે, જેને રિકર્ઝન કહેવામાં આવે છે. પુનરાવર્તનનો અર્થ એ છે કે, આપણે આ સમસ્યાનું નિરાકરણ કરીને નાની સમસ્યાનું નિરાકરણ કરીને અને આ પ્રશ્નનો ઉપયોગ કરીને અમારી વર્તમાન સમસ્યાના જવાબની સીધી નોંધણી કરવા જઈ રહ્યા છીએ. તેથી અમે m અને n ના જુસીડીને હટાવવા માંગીએ છીએ, પરંતુ m અને n ના જુસીડી તેના બદલે અમે gcd n અને m minus n હટાવીએ છીએ અને જે જવાબ અમને આપે છે અમે તે માટે સીધો જ જુસીડી તરીકે રિપોર્ટ કરીએ છીએ, તેથી અમે ફક્ત નાના મૂલ્યો સાથે કાર્ય કરે છે અને પછી આપણે તેને પરત કરીએ છીએ. હવે જ્યારે તમે આવું ફરી યાદ આવવું કોલ કરો છો, ત્યારે તે એક લૂપ જેવું છે; તે ફંક્શનને ફરીથી બોલાવશે, તે બદલામાં નાના ફંક્શનનો ઉપયોગ કરશે અને તેથી વધુ. અને તમારે સુનિશ્ચિત કરવું પડશે કે આ અનુક્રમમાં જે જુસીડી જુસીડીને જુદા જુદા મૂલ્યો સાથે બોલાવવાનું ચાલુ રાખે છે, તે અટકાવ્યા બિંદુ વગર અનંત પ્રગતિ થતું નથી. તેથી, ઔપચારિક રીતે આપણે તેની ખાતરી કરવી તે છે કે જે n ને વિભાજન કરે છે તે શોધવાની આ બાંધધરી છે, તેથી આ તે જગ્યા છે જ્યાં જુસીડી વાસ્તવમાં પોતાને ફોન કર્યા વિના બહાર નીકળે છે. આપણે ખાતરી કરવી પડશે કે આખરે અમે આ બિંદુ સુધી પહોંચીશું. જો તમે અહીં જુઓ તો શું થઈ રહ્યું છે તે છે કે જે જુસીડી (GCD) ને પસાર કરાવેલી કિંમતો નાની અને નાની થઈ રહી છે હવે આપણે માઇનસ n એ માટે શું કરી શકીએ? મૂલ્ય શું હોઈ શકે? તે 0 હોઈ શકે? સારું, જો મીટર બાદ n એ 0 હોય તો તેનો અર્થ એમ બરાબર છે, જો એમ બરાબર n થાય, તો નિશ્ચિતપણે મીટર n દ્વારા વિભાજીત થાય છે. જો એમ બાદબાજી n એ 0 હોય તો તે બહાર નીકળ્યું હોત, તેથી તે 0 ન હોઈ શકે. તે ઓછામાં ઓછા 1 હોવું જોઈએ, તેથી જ્યારે પણ આપણે આ ફંક્શનને કહીશું, માઇનસ n તે ઓછામાં ઓછી એક છે. બીજી બાજુ દરેક વખતે આપણે નાના મૂલ્યો સુધી પહોંચીએ છીએ. તેથી, આપણે કેટલાક મૂલ્યથી શરૂ કરીએ છીએ અને m બાદમાં ઘટાડો ઘટતો રહે છે. જ્યારે તે વાસ્તવમાં 1 સુધી પહોંચે છે ત્યારે શું થાય છે? ઠીક છે, જ્યારે તે 1 સુધી પહોંચે છે અને 1 દરેક અન્ય નંબરને વિભાજિત કરે છે, તેથી m ટકા n અથવા m ભાગ્યા n , બાકીની 0 હશે, તેથી આપણે 0 ના જુસીડી પરત કરીશું. બીજા શબ્દોમાં કહીએ તો, અમે આ ફંક્શનને ખાતરી આપી હતી કારણ કે તે સંખ્યાને ઘટાડીએ છીએ જેને આપણે કાર્યને બોલાવીએ છીએ, અંતે આખરે કોલ કરશે જ્યાં જુસીડી સમાપ્ત થાય છે. આ અગત્યનું છે અને અમે આ પછી પાછા આવીશું પણ જ્યારે પણ તમે આના જેવી કાર્ય લખશો, ત્યારે તમારે ખાતરી કરવી પડશે કે એક મૂળભૂત કેસ છે જે મર્યાદિત સંખ્યામાં પગલાઓ પર પહોંચી જશે, ભલેને તમે શરૂ કરો તે કોઈ બાબત નથી. આ યુક્લિડનું અલ્ગોરિધમ છે, પ્રથમ સંસ્કરણ જ્યાં આપણે જોયું કે m અને n ના gcd ને gcd n અને m માઇનસ n દ્વારા બદલી શકાય છે. અને આ ચોક્કસ અમલીકરણમાં આપણે જે કંઈ જોયું છે તે ત્રણ બાબતો છે, આપણે જોયું કે આપણા કોડમાં ટિપ્પણી કેવી રીતે કરવી, આપણે જોયું છે કે પાચથીન એ એક જ સમયે બે ચલોનો એક સાથે સુધારાની મંજૂરી આપે છે, જેથી m અલ્પવિરામ n એ બરાબર એન અલ્પવિરામ મી. અમે જોયું છે કે આપણે વર્તમાન ફંક્શનની ગણતરી કરવા માટે સમાન કાર્યનો ઉપયોગ નવા દલીલો સાથે કરી શકીએ છીએ. તેથી એમ કહીને કોઈ સમસ્યા નથી કે m અને n ના

જીસીડીની ગણતરી કરવા માટે, હું તેના બદલે અન્ય કોઈ મૂલ્ય પર જીસીડીની ગણતરી કરું છું અને મારા જવાબને પરત આપવા માટે તેનો જવાબ આપું છું.

(સ્લાઇડ સમયનો સંદર્ભ લો: 11:53)

ચાલો આ અલ્ગોરિધમનો એક અલગ સંસ્કરણ જોઈએ, જ્યાં આપણે રિકર્સિવ કોલને જ્યારે લૂપ દ્વારા બદલીએ છીએ. અમે આ પ્રમાણભૂત અલ્ગોરિધમનો અમારા છેલ્લા સંસ્કરણમાં જોયું જ્યારે અમે લઘુત્તમ લઘુત્તમ અલ્પવિરામથી n થી 1 ની ગણતરી કરી રહ્યા હતા, તેથી અમે તપાસ કરી હતી કે 0 કરતા વધારે હતો કે નહીં અને અમે ડિબ્રેટીંગ રાખ્યું છે. ઠીક છે, અહીં આપણે એક નો ઉપયોગ કરીને રિકર્ઝન કરી રહ્યા છીએ જ્યારે, તેથી અહીં નોંધવું પ્રથમ વસ્તુ એ છે કે મેં આ ટિપ્પણીને ખસેડી દીધી છે જે લીટીના અંતે એક અલગ લીટીમાં ઉપયોગમાં લેવાતી હતી. શું પાચથોન કહે છે, જો હેશ હોય તો બાકીની રેખા અવગણવામાં આવે છે. તેથી, તે આ વાક્યને વાંચે છે તે એક માન્ય શરત જુએ છે અને તે પછી હેશ જુએ છે, તેથી તે એવું છે કે જ્યારે આ એક્ઝિક્યુટ થાય ત્યારે આ નિવેદન પાચથોન પ્રોગ્રામનો ભાગ ન હતો. ટિપ્પણી ક્યાં એક અલગ રેખામાં હોઈ શકે છે અથવા તે એક રેખાના અંતમાં હોઈ શકે છે. અલબત્ત, યાદ રાખો કે તમે તેના પછી કંઈપણ ન મૂકી શકો જે તમે પાચથોન ને એક્ઝિક્યુટ કરવા માગો છો, કારણ કે તે હેશ જુએ છે પછી બાકીની રેખા અવગણવામાં આવે છે, તેથી તે મધ્યમાં ન હોઈ શકે, તમે કોઈ મધ્યમાં ટિપ્પણી કરી શકતા નથી. એક લીટીના, પરંતુ તમે તેને અલગ રેખા પર મૂકી શકો છો અથવા તમે તેને લીટીના અંતે મૂકી શકો છો. તેથી કોઈપણ રીતે આ પહેલાં અમારી ટિપ્પણી છે. તેથી અહીં સુધી કોઈ ફેરફાર નથી સિવાય કે મેં ટિપ્પણીની સ્થિતિને ખસેડી દીધી છે. હવે અમે આ બિંદુએ પહોંચીએ છીએ જ્યાં અમને ખરેખર કેટલાક ગણતરી કરવા પડે છે. આ બિંદુએ જો અમને n મળ્યા છે જે n ને વિભાજીત કરે છે તો અમે પૂર્ણ કરી શકીએ છીએ અને અમે સીધા જ n પરત કરી શકીએ છીએ. તેથી, આ આપણું રિકર્સિવ કોડ શું કરશે. જો અમને આટલું ન મળ્યું હોય તો આપણે કેટલાક કામ કરવું પડશે. શરત એ છે કે એન દ્વારા વિભાજીત થયેલ છે તે વાસ્તવમાં બાકીનું ઉત્પાદન કરે છે કે નહીં તેથી, આ ચિન્હ બરાબર નથી, આ ઉદ્ધાર ચિન્હ સાથે પરત આવે છે, તેથી આ ગાણિતિક સમાન નથી. યાદ રાખો કે સમાનતાના ગાણિતિક પ્રતીક માટે આપણે આ ડબલનો ઉપયોગ કરીએ છીએ. આ અમારા પ્રતીકનું બરાબર નથી. તેથી, જ્યાં સુધી બાકી રહેલો છે ત્યાં સુધી, તે બાકીનો ભાગ n ભાગ્યા n નથી 0 આપણે તફાવતની ગણતરી કરીએ તે પહેલા આપણે શું કર્યું અને આપણે બે કિંમતોના મહત્તમ અને મહત્તમ મૂલ્ય દ્વારા મીટરને બદલીએ બે મૂલ્યોના નાના દ્વારા. અમારી પાસે એક જોડી એમએન હોય છે જેની જીસીડી અમે અધિકાર શોધવાનો પ્રયાસ કરી રહ્યા છીએ, ધારણા સાથે એમ દરેક પગલે n કરતાં મોટી છે, અમે n ના મોટા અને તફાવત અને n ના નાના દ્વારા અને n ના તફાવત અને તફાવતને બદલીએ છીએ. આ બરાબર આપણે ફરી યાદ આવતી કોલમાં શું કરી રહ્યા છીએ, અમે કહીએ છીએ કે અમે તે જીસીડીની ગણતરી કરી રહ્યા છીએ. અહીં આમાં જ્યારે લૂપ અસરકારક રીતે અમે કહીએ છીએ કે મહત્તમ n diff અને ન્યુનત્તમ n diff ની ગણતરી દ્વારા mn ના જીસીડીને બદલવું. અમે જ્યાં સુધી કોઈ શરતને હટાવવામાં ન આવે ત્યાં સુધી અમે આ કરવાનું ચાલુ રાખીએ છીએ, જ્યાં ખરેખર એન મીટર વિભાજિત થાય છે, અને બરાબર જેમ આપણે ફરી યાદ આવતાં કિસ્સામાં જણાવ્યું હતું કે ત્યાં એક સીમા કેસ હશે જ્યાં સૌથી ખરાબ કિસ્સામાં n બનશે અને 1 બધું જ વિભાજિત થશે. તે જ રીતે અહીં તફાવત ઓછો રાખશે, તફાવત 0 ન હોઈ શકે, કારણ કે જો તફાવત 0 છે તો તે વિભાજિત થઈ શકે છે, તેથી મોટાભાગના તફાવત નીચે 1 પર જઈ શકે છે અને જ્યારે તે એકને હિટ કરે છે ત્યારે આપણે પૂર્ણ થાય છે. તે જ યાદ આવવું કાર્યનું આ થોડું વર્ઝન જે આપણે લખ્યું હતું, તેથી જો તે તમને મદદ કરે તો તમે આ બાજુની બાજુએ જોઈ શકો છો અને સમજવા પ્રયત્ન કરો કે આ ફરી યાદ આવતી વસ્તુઓ શું કરી રહી છે અને જ્યારે શું કરી રહ્યું છે અને જુઓ કે તેઓ મૂળભૂત રીતે તે જ કરી રહ્યા છે વસ્તુ. અને રિકર્ઝનને સમાપ્ત કરવું જોઈએ તે વિચાર બરાબર એ દાવાના સમાન છે કે જે આપણે અગાઉ કહ્યું હતું કે જ્યારે તમે લખો છો ત્યારે તમારે ખાતરી કરવી જ પડશે કે જ્યારે તમે શરત ખોટી બનાવવા તરફ પ્રગતિ કરી છે, જેથી જ્યારે બહાર નીકળે. તેથી, જેમ રિકર્ઝન હંમેશાં પર જઈ શકે છે, જો તમે સાવચેત ન હોવ અને તમે તે

દલીલો સાથે પરિપૂર્ણ નહીં કરો જે સમાપ્તિની બાંધધરી આપે છે, જ્યારે તમે હંમેશાં જ ચાલુ રાખી શકો છો જો તમે ખાતરી કરો કે તમે તે સમયે પ્રગતિ ન કરો જ્યારે શરત આખરે ખોટી બની જાય છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 15:42)

આપણે વાસ્તવમાં આ કરતા થોડું સારું કરી શકીએ છીએ. ચાલો હાથની અમલ કરીને આમાં એક સમસ્યા જુઓ. તેથી અમે 101 અને 2 ના જુસીડી જેવી સંખ્યા સાથે શરૂઆત કરીએ તેવું ધારવું, પછી અમારા એલ્ગોરિધમ કહેશે કે આ હવે તફાવતના જુસીડી અને n બનશે, તફાવત 99 છે તેથી 99 અને 2 હશે, અને પછી તે 97 ના જુસીડી બનશે અને 2 અને તેથી વધુ. તેથી, અમે આ લગભગ 50 પગલાં રાખીશું અને છેવટે અમે 5 અને 2 ના જુસીડી નીચે આવીશું અને પછી 3 અને 2 ના જુસીડી આવશે. હવે જ્યારે આપણે તફાવતની ગણતરી કરીએ છીએ ત્યારે આપણને 2 અને 1 ના જુસીડી મળે છે, તેથી હવે તફાવત નાની બનો પછી આ બિંદુએ અમે જાણ કરીશું કે જવાબ 1 છે. તેથી, વાસ્તવમાં અમને 101 ના 2 જુસીડી કરવા માટે આશરે 50 પગલાંઓ લે છે. નિષ્કપટ અભિગમની અમારી એક ટીકા એ છે કે તે નંબરો પોતાને પ્રમાણમાં લે છે. જો તમારી પાસે સંખ્યાઓ m અને n હશે તો આપણે લઘુત્તમ મીટર અને n ની બરાબર ની સમાન સંખ્યામાં લેશે. હવે અહીં, વાસ્તવમાં આપણે લઘુત્તમ કરતા મોટા પગલાં લઈ રહ્યા છીએ કારણ કે લઘુત્તમ 2 છે, જો તમે માત્ર પરિબલો પરિબલો હતા તો આપણે જોશું કે 2 નું માત્ર પરિબલ 2 છે અને તે એક પરિબલ નથી 101, આપણે શરૂઆતમાં જ અટકાવીશું. આ વાસ્તવમાં સૌથી ખરાબ લાગે છે, પછી કેટલાક કિસ્સાઓમાં અમારા અગાઉનાં અલ્ગોરિધમનો.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 17:05)

અહીં એક સારી નિરીક્ષણ ધારવું એ છે કે મીટર ન વહેંચે છે. બીજા શબ્દોમાં કહીએ તો જો હું એમ એન ની વહેંચાય તો ભાગ્ય અને બાકીનો ભાગ મળશે. તેથી, હું n તરીકે q વખત n વત્તા r લખી શકું છું જ્યાં q ભાગ્ય છે અને r બાકી છે તેથી, તમે આ શબ્દો હાઈ સ્ક્રલ અંકગણિતથી યાદ રાખી શકો. એન એનક વખત જાય છે અને બાકીના આર છોડી દે છે અને અમને ખાતરી છે કે આર એ n કરતા ઓછી છે, અન્યથા તે વધુ એક વખત જઈ શકે છે તે q વત્તા 1 બની જશે. અમારી પાસે બાકીની આર છે જે n કરતા ઓછી છે. તેથી ઉદાહરણ તરીકે જો હું કહું 7 અને હું તેને 3 વડે વિભાજિત કરવા માંગું છું, તો આ 2 ગુણ્યા 3 વત્તા 1 થશે, તો આ મારી ભાગ હશે અને આ મારું બાકી રહેલું હશે. અને મહત્વની વસ્તુ બાકી છે તે હંમેશા હું શું કરી રહ્યો છું તેના કરતા નાના હોય છે. હવે, ચાલો આપણે પહેલાં એમ ધારીએ કે આપણી પાસે m અને n બન્ને માટે સામાન્ય ભાજક છે. બીજા શબ્દોમાં કહીએ તો આપણે પહેલા ડી અને n તરીકે એક વખત d અને n તરીકે લખી શકીએ છીએ કારણ કે કેટલાક નંબરો a અને b માટે d છે, કારણ કે m એ d નું બહુવિધ છે અને તે n છે. જો તમે તેને અહીંથી સમીકરણમાં પ્લગ કરો તો, આપણે જોઈએ છીએ કે m એ ઘણી વખત છે જે q વખત n બરાબર છે જે b વખત d plus r છે. તેથી, ડી ડાબે વિભાજિત કરે છે અને ડી જમણી એક ભાગ વિભાજિત. તમે સરળતાથી તમારી જાતને મનાવી શકો છો કે ડી એ પણ વહેંચવું જોઈએ. જો તમે ચિત્રાત્મક રીતે કરવા માંગો છો તો તેના વિશે વિચારવાનો રસ્તો એ છે કે મારી પાસે આ નંબર મીટર છે અને હું તેને n ના એકમોમાં તોડી શકું છું અને પછી અહીં એક નાનું બીટ છે. બીજી બાજુ જો હું d ને જોઉં છું, તો સમાન રીતે બધું જ વિભાજન થાય છે. તેથી તે આ બ્લોકમાં વિભાજિત કરે છે અને તે પણ સમગ્ર વસ્તુને વિભાજિત કરે છે. જો હું ડી સાથે ચાલુ રાખું છું, તો તે આ સરહદ પર બરાબર બંધ થવાનું કારણ છે કારણ કે ડી એ પણ n ને વિભાજિત કરે છે, તેથી ડી બરાબર બરાબર છે જે આ છેલ્લા બીટને વિભાજિત કરવું જ જોઈએ. બીજા શબ્દોમાં કહીએ તો, આપણે ખૂબ સરળતાથી એવી દલીલ કરી શકીએ છીએ કે આર એ ડીની એક મલ્ટીપલ પણ હોવી જોઈએ. તેથી ડી સાથે સાથે આર વિભાજિત જ જોઈએ.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 19:21)

જો ડી વિભાજન મીટર અને બી ને વિભાજન કરે તો ડી દ્વારા બાકીની ભાગો n ને વિભાજીત કરવી પડશે. અને જેમ આપણે પહેલા તફાવત સાથે જોયું તેમ, છેલ્લી વખત મેં કહ્યું હતું કે આપણે એન દ્વારા વિભાજીત તફાવત એમ જોવું જોઈએ. હવે અમે કહીએ છીએ કે આપણે બાકીના ભાગને એન અને ડી દ્વારા વિભાજીત કરવું જોઈએ અને ડી ખરેખર જીસીડી નો અને આ બાકી હશે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 19:44)

આ સુધારેલ છે અને આ એલ્ગોરિધમનું વાસ્તવિક વર્ઝન છે કે યુક્લિડે દરખાસ્ત કરી છે, નહીં કે તફાવત પરંતુ બાકીનો એક. એમ કહે છે કે એમના જીસીડી અને એન એમ ધારી રહ્યા છીએ કે મીટર એ મોટી સંખ્યામાં n છે. હવે જો n એ વિભાજન કરે છે તો આપણે પૂર્ણ થાય છે, આપણે ફક્ત n પરત કરીએ છીએ, તે પહેલાંની જેમ જ છે. નહિંતર, દોરેલા ભાગની સંખ્યાને બાકીના ભાગમાં મળશે અને બાકીનાને મળશે અને n અને r ના જીસીડી પરત કરશો, અને આ બિંદુએ યાદ રાખવું એક અગત્યની બાબત એ છે કે આર ચોક્કસપણે n કરતા ઓછી છે. તેથી આપણે અહીં આ સ્થિતિની ચિંતા કરવાની જરૂર નથી, આપણે વધુમાં વધુ અને મિનિટે લેવાની જરૂર નથી કારણ કે બાકીનાએ એન કરતા ઓછો હોવાની ખાતરી આપી છે અને ન તો એક વધુ સમય જશે.

(સ્લાઈડ ટાઇમ નો ઉલ્લેખ કરો: 20:31)

જેમ પહેલાં આપણે આનું અમલીકરણ સરળ બનાવવું તે પહેલાં, અને આ વધુ સરળ છે કારણ કે અમારે આ મહત્તમ મિનિ બિઝનેસ કરવાની જરૂર નથી. તેથી, અગાઉના સમયની જેમ આપણે પ્રથમ એમ અને એન ફ્લિપ કરીએ છીએ, જો તેઓ યોગ્ય ક્રમમાં નથી. તો જો n એ મીટર વહેંચાય તો એન નું વિભાજન થયેલ ભાગ બાકીની સંખ્યા 0 છે, અને આપણે પૂર્ણ થાય છે, નહિ તો આપણે જીસીડી અને બાકીની રકમ પરત કરીશું, તેથી આ બાકીની ભાગ છે. અને યાદ રાખો કે બાકીનો ભાગ હંમેશા n કરતા ઓછો છે તેથી અમે તેને ફ્લેપિંગ અને આ બિંદુએ મહત્તમ અને મિનિટ લેવા વિશે ચિંતા કરવાની જરૂર નથી. અને સમાન TP અગાઉના કિસ્સામાં અમે તેને ફરીથી યાદ આવવું વસ્તુ સાથે કરવાનું બદલે આ સમયે સમગ્ર વસ્તુ કરી શકો છો. અમે પ્રથમ એમ અને એન વિનિમય કરીએ છીએ જો તેઓ ખોટા ક્રમમાં હોય, તો પછી જ્યાં સુધી બાકીનો નથી 0 આપણે બે નંબરો ના નાના દ્વારા મીટરને બદલીએ છીએ અને આપણે બાકીની સંખ્યામાં n ને બદલીએ છીએ અને અમે આગળ વધીએ છીએ. હવે અમે ખાતરી આપીએ છીએ કે આ શેષ કાં તો 0 પર જશે, પરંતુ જો તે 0 માં જશે તો તેનો અર્થ એ થાય કે તે વિભાજન કરે છે અથવા જો તે સૌથી ખરાબ કિસ્સામાં 0 ન હોય તો શેષ ઘટી જાય છે કારણ કે તે હંમેશા આપણે જે નંબર સાથે પ્રારંભ કર્યો તેના કરતા નાની છે. તેથી તે ઘટતું રહે છે અને તે 1 સુધી પહોંચે છે પછી આગળના પગલામાં તે ભાગ લેશે. તેથી આખરે, અમે ઓછામાં ઓછી એક પાછી આપીશું.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 21:48)

જો આપણે તે ઉદાહરણ પર પાછા જઈએ જે આપણે જોઈ રહ્યા હતા, તેથી જો આપણે જોયું કે gcd 101, 2, અને અમે તે તફાવતનો ઉપયોગ કરીને કર્યું છે જેણે કહ્યું કે અમે 50 પગલા લીધા છે. હવે અહીંયા આપણે બાકીના કરીએ તો હું સીધી જ શોધ કરું છું કે, r બરાબર 1 બરાબર છે જો હું 101 બાય 2 ને વિભાજીત કરું તો તે 50 ગણી બાકી રહે છે. એક પગલુંમાં હું જીસીડી 2 અલ્પવિરામ 1 માં જઈશ અને મને 1 મળશે. વાસ્તવમાં, તમે શું બતાવી શકો છો કે બાકીનો સાથે આ સંસ્કરણ વાસ્તવમાં આંકડાઓની સંખ્યાને પ્રમાણસર લે છે, તેથી જો હું સો આંકડો નંબર કહ્યો હોય તો તે લગભગ સો પગલાઓ લેશે. તેથી દાખલા તરીકે જો આપણી સંખ્યા એક અબજ જેટલી છે, તેથી આશરે 10 થી 9 માં આશરે દસ અંકો હશે. પછી જો હું નિષ્કપરીક અલ્ગોરિધમ કરું તો તે બિલિયન નંબરોના કેટલાક સતત સમય લાગી શકે છે કહે છે કે એક અબજ પગલાં. પરંતુ આ અલ્ગોરિધમનો કારણ કે તે 10 થી 9 ના અંકોની સંખ્યાના પ્રમાણમાં અંદાજે 10 અંકો ધરાવે છે તે ફક્ત 10 પગલાં લે છે, તેથી આ સંસ્કરણમાં કાર્યક્ષમતામાં નાટ્યાત્મક

સુધારો થયો છે. આ અમે કંઈક કરી રહ્યા છીએ જ્યારે અમે આ અભ્યાસક્રમ કરી રહ્યા છીએ. આ કોર્સ પ્રોગ્રામિંગ, ડેટા માળખાં અને ગાણિતીક નિયમો વિશે છે. તેથી પ્રોગ્રામિંગ ભાગ એવી રીતે વાત કરે છે કે પ્રોગ્રામમાં આપેલ વિચારને વ્યક્ત કરવાની શ્રેષ્ઠ રીત કઈ છે કે તે સરળ છે તેની ખાતરી કરવા માટે સરળ છે કે તે વાંચવા અને જાળવવા માટે સરળ અને સરળ છે, તેથી તે પ્રોગ્રામિંગ ભાગ છે. તમે કેવી રીતે લખો છો, તમે કેવી રીતે તમારા વિચારોને સૌથી સ્પષ્ટ ફેશનમાં વ્યક્ત કરી શકો છો. પરંતુ આ વિચાર પોતે સ્પષ્ટ હોવો જોઈએ અને તે જ છે જ્યાં ડેટા માળખાં અને અલ્ગોરિધમનો પ્રારંભ થાય છે. તેથી તમે સુંદર ગદ્ય લખી શકો છો, પરંતુ તમારી પાસે કોઈ વિચારો નથી અથવા તમારી પાસે ખૂબ તેજસ્વી વિચારો હોઈ શકે છે પરંતુ તમે કઢંગી રીતે પોતાને વ્યક્ત કરી શકો છો, તેમાંના બંને શ્રેષ્ઠ નથી આ અન્ય કોઈ ભાષામાં લખવા જેવું છે. તમે સ્પષ્ટ કરવા માટે તેજસ્વી વિચારો ધરાવી શકો છો, પરંતુ જો તમે વિચારો કે જેની સાથે વાત કરી રહ્યા હોવ તે વ્યક્તિને તમે તેમને અભિવ્યક્ત કરી શકતા નથી, તો તેની અસર ગુમાવે છે તેથી, તમારે વિચારોની જરૂર છે અને તમને તેમને અભિવ્યક્ત કરવા માટે એક ભાષાની જરૂર છે. પ્રોગ્રામિંગ આ વિચારોને વ્યક્ત કરવા વિશે છે, પરંતુ વિચારો પોતે એલ્ગોરિધમ્સ અને ડેટા સ્ટ્રક્ચર્સથી આવે છે.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ અને એલ્ગરિધમ્સ ઇન પાયથોન
પ્રો. માધવન મુકુન્દ ડિપાર્ટમેન્ટ
ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 01 લેક્ચર - 04

પાયથોન ડાઉનલોડ અને ઇન્સ્ટોલ કરવું આ પ્રથમ અઠવાડિયાના અમારા અંતિમ વ્યાખ્યાન માટે, આપણે જોશું કે કેવી રીતે વાસ્તવમાં ઉપયોગ કરવો અમારી સિસ્ટમ પર પાયથોન.

(સ્લાઇડ સમયનો સંદર્ભ લો: 00:10)

પાયથોન પ્રોગ્રામિંગ ભાષા છે, જે તમામ પ્લેટફોર્મ્સ પર ઉપલબ્ધ છે. તેથી, તમે Linux પર અથવા Mac પર અથવા Windows પર કાર્ય કરી રહ્યા છો, તો તમે તમારી સિસ્ટમ પર કામ કરતી પાયથોન ના સંસ્કરણને શોધી શકશો. પાયથોન સાથેના નાના જટીલતાઓમાંની એક એવી છે કે બે સ્વપ્નો અથવા પાયથોનનાં બે સંસ્કરણો છે, જે સામાન્ય રીતે મળી આવે છે. તેથી, પાયથોન 2.7 નામની જૂની સંસ્કરણ છે, અને તેમાં પાયથોન 3 તરીકે ઓળખાતી નવી આવૃત્તિ છે. પાયથોન 3 એ સક્રિય રીતે વિકસિત કરવામાં આવે છે, પાયથોન 2.7 વધુ અથવા ઓછા એક સ્થિર સંસ્કરણ છે અને હાલમાં પાયથોન 3 ની આવૃત્તિ 3.5 છે. 2 અથવા તે કંઈક તેથી, તમે 3.5 અથવા 3.4 નો ઉપયોગ કરી રહ્યા છો તે ખૂબ જ તફાવત નથી, પરંતુ 2.7 અને 3 વચ્ચે તફાવત છે. અને આ અભ્યાસક્રમના હેતુ માટે, અમે પાયથોન 3 સાથે કામ કરીશું.

(Slide Time નો સંદર્ભ લો: 01:06)

શું છે? આ બે આવૃત્તિઓ વચ્ચેનો તફાવત? ઠીક છે, પાયથોન થોડા લક્ષણો સાથે શરૂઆત કરી હતી અને તે વધુ સર્વતોમુખી પ્રોગ્રામિંગ ભાષામાં વિકાસશીલ રાખવામાં આવ્યું હતું. તેથી, પાયથોન ખૂબ જ પુનરાવર્તન અને પાયથોન 2.7 ની મદદથી પસાર થયું હતું, જ્યારે પાયથોનના ડેવલપર્સ નક્કી કર્યું હતું કે તમારે શુદ્ધ શરૂઆત કરવી જોઈએ. અને કેટલીક નવી સુવિધાઓ જે ભાષામાં ઉમેરાઈ હતી અને એડ હોંક રીત સારી રીતે સંકલિત કરવી જોઈએ જે તેને વધુ મજબૂત પ્રોગ્રામિંગ ભાષા બનાવે છે. પાયથોન 3 એ આવશ્યકપણે પાયથોન નો આધુનિક સંસ્કરણ છે, જેમાં તે લક્ષણોનો સમાવેશ થાય છે કે જ્યાં પાયથોન પર ઉમેરાય છે, કારણ કે તે જૂથને તે રીતે વધુ સુસંગત અને વધુ સરળ બનાવે છે, પરંતુ ઘણી વાર લોકો ઘણા પહેલાથી પાયથોન નો ઉપયોગ કરી રહ્યાં છે, અને પાયથોન 2.7 માં તે સોફ્ટવેરનો ઉપયોગ કરીને ઘણાં સોફ્ટવેર લખાયા છે. ખાસ કરીને ઘણા સોફ્ટવેર કે જેમને વિધેયોની વૈજ્ઞાનિક અને આંકડાકીય લાઇબ્રેરીઓ જેવા ઉપયોગ કરવા માટે અનુકૂળ લાગે છે, જ્યાં તેઓ તેનો ઉપયોગ ન કરતા હોય, તેઓ ફક્ત આ લાઇબ્રેરીઓનો જ ઉપયોગ કરી શકશે હજી પણ પાયથોન 2.7 દ્વારા લખવામાં આવે છે. અને જો તમે તેને પાયથોન 3 થી ચલાવશો તો ક્યારેક આ વિધેયો તે અપેક્ષા રાખતા નથી કારણ કે તે અપેક્ષિત છે. તેથી, આને પાયથોન 2.7 પર રહેવા માટે ફરજ પડી છે આખરે અમે આશા રાખીએ છીએ કે કોઈ વ્યક્તિ પાયથોન 2.7 લાઇબ્રેરીઓને પાયથોન 3 પર ખસેડવા માટે પ્રયત્ન કરશે. અને અલબત્ત, નવા કોડને મોટે ભાગે પાયથોન 3 પર વિકસાવવામાં આવી રહ્યો છે, પરંતુ તમારે યાદ રાખવું જોઈએ કે જ્યારે કોઈ વ્યક્તિ કહે છે કે તેઓ પાયથોન વાપરી રહ્યા છે ત્યારે તેઓ આ અંગે વાત કરી રહ્યા છે 2.7 અને ન 3, અને તમારે ગોઠવણ કરવી પડશે. પ્રારંભિક સામગ્રીના હેતુ માટે કે અમે કોર્સમાં કરીશું, ત્યાં પાયથોન 3 અને પાયથોન 2.7 વચ્ચે લગભગ કોઈ ફેરફાર થતો નથી; તેમ છતાં, કેટલીક સુવિધાઓ છે કે જે અમે 2.7 માં સહેજ અલગથી જોશું અને અમે તેમને 3 માં શોધી કાઢીએ છીએ, અને હું આ તફાવતોને પ્રકાશિત કરવાનો પ્રયત્ન કરું છું કારણ કે અમે લાંબા સમય સુધી જઈએ છીએ. પરંતુ પાયથોન 3 માં

આગળ વધવું વર્તમાન સંસ્કરણ છે અને તે ઓછામાં ઓછું 4 અથવા 5 વર્ષ માટે ઘણાં વર્ષોથી વર્તમાન સંસ્કરણ રહ્યું છે. તે ચોક્કસપણે ભાષા છે, જે ભવિષ્યમાં પ્રભુત્વ ધરાવે છે, તેથી તે સારું છે કે તમે નવા સંસ્કરણથી શરૂઆત કરો અને પછી જૂના સંસ્કરણ પર પાછા જાઓ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 03:23)

જ્યાં સુધી આ અભ્યાસક્રમની ચિંતા છે, પાયથોન 3 નું કોઈ પણ વર્ઝન સારું હોવું જોઈએ. મેં કહ્યું તે મુજબની તાજેતરની આવૃત્તિ 3.5.x છે, જ્યાં મારી પાસે વસ્તુ 2 છે, પણ જો તમારી પાસે 3.5 નથી, પરંતુ તમારી પાસે 3.4 કે 3.3 છે તો બધું બરાબર કામ કરવું જોઈએ નહીં. પરંતુ જો તમને રસ હોય તો, તમે નવીનતમ, નવીનતમ સંસ્કરણને ઇન્સ્ટોલ કરી શકો છો. જો તમે લીનક્સનો ઉપયોગ કરો છો, તો તે સામાન્ય રીતે ડિફોલ્ટ રૂપે હોવો જોઈએ કારણ કે ઘણા લિનક્સ ઉપયોગિતાઓને પાયથોન ની જરૂર છે અને તેથી પાયથોન તમારા પર હોવું જોઈએ સિસ્ટમ છે, પરંતુ તે હોઈ શકે છે કે ઉપયોગિતા પાયથોન 2.7 નો ઉપયોગ કરે છે. તેથી, ખાતરી કરો કે તમે પાયથોન 3 ઇન્સ્ટોલ કરો. તમે આ કરવા માટે પેકેજ મેનેજરનો ઉપયોગ કરી શકો છો. હવે જો તમે MAC નો ઉપયોગ કરી રહ્યા છો અથવા તમે Windows વાપરી રહ્યા છો તો પાયથોન ખાસ કરીને ઇન્સ્ટોલ થઈ શકશે નહીં અથવા પાયથોન 3 કદાચ તમારી સિસ્ટમ પર ઇન્સ્ટોલ થઈ શકશે નહીં. અહીં આપેલ URL છે. જો તમે Google પર શોધ કરો છો, તો તમને તે મળશે. ફક્ત પાયથોન 3.5 માટે ઇન્સ્ટોલ અથવા ડાઉનલોડ કરો અને તમને આ URL પર મળશે. તેથી, www.python.org ડાઉનલોડ પાયથોન 350 ડાઉનલોડ કરો. 350, ખરેખર 3.5.0 નો સંદર્ભ લે છે. તેથી, વાસ્તવમાં વર્તમાન આવૃત્તિ જે મેં કહ્યું તે 3.5.0 નથી, પરંતુ 3.5.2 છે. તેથી, તમે ત્યાં સૂચનાઓ મેળવશો - કૃપા કરીને તમારા સિસ્ટમ માટે યોગ્ય છે તે સંસ્કરણ ડાઉનલોડ કરો અને તેને ઇન્સ્ટોલ કરો. આ ફાઇલોને સ્વયં-સમજાવી શકાય તેવી ફાઇલો માટે રચવામાં આવી છે; જો તમારી પાસે કોઈ સમસ્યા હોય તો તમે જે સમસ્યાનો સામનો કરી રહ્યાં છો તેમાં મદદ માટે ઓનલાઇન શોધો અથવા તમારી આસપાસની કોઈ વ્યક્તિને પૂછો. સોફ્ટવેર ઇન્સ્ટોલ કેવી રીતે કરવું તે તમને કહેવામાં ઘણો સમય ગાળવા આ કોર્સનો હેતુ નથી. તેથી, હું આશા રાખું છું કે તમે આ કરી શકો છો, જેથી અમે વાસ્તવિક પ્રોગ્રામિંગ ભાગ સાથે આગળ વધી શકીએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 05:05)

જો તમે અન્ય પ્રોગ્રામિંગ ભાષાઓથી પરિચિત હોવ તો, ધ્યાનમાં રાખવાનું એક વધુ વસ્તુ, દુભાષિયા અને કમ્પાઇલર્સ વચ્ચે તફાવત છે. તેથી, મુખ્ય સમસ્યા એ છે કે પાયથોન અથવા C અથવા C++ અથવા Java જેવા પ્રોગ્રામિંગ ભાષા પર અમને સૂચનાઓ સમજવા અને લખવા માટે લખવામાં આવે છે. તેથી, આ અંશે ઊંચા સ્તર સૂચનો છે પ્રશ્નની બીજી બાજુએ, કમ્પ્યુટર્સને નીચલી સ્તર સૂચનોની જરૂર છે. તેથી, જ્યારે આપણે નામો અને મૂલ્યો વિશે વાત કરીએ, જેમ કે હું, જે અથવા આપણે સૂચિ વિશે વાત કરીએ છીએ, તો અન્ડરલાઇન કમ્પ્યુટર સીધી આ બાબતોનું વિશ્લેષણ કરી શકશે નહીં, તેથી અમને ભાષાંતરની જરૂર છે. જો તમને પ્રથમ વ્યાખ્યાન યાદ છે, અમે ચેર વ્યવસ્થા વિશે વાત કરી. તેથી, અમે કહ્યું કે ચેર ઉચ્ચ સ્તરની વસ્તુ તરીકે ગોઠવે છે, અને અમે જણાવ્યું હતું કે, 8 પંક્તિઓમાં 10 થી 10 અક્ષરોમાં 10 અક્ષરો, 10 દરેક અધિકાર. અમે કહ્યું છે કે તેઓ તમને વિગતવાર સૂચનો આપે છે તે વિગતમાં તફાવત હોઈ શકે છે અને આ ચોક્કસપણે શું થાય છે. ક્રમમાં એક્ઝેક્યુટેબલ ફાઇલ કહેવામાં આવે છે કે જે આપણે આવે છે તે કંઈક છે જે મશીન સમજી શકે તે સ્તરે પરત આવે છે. જ્યારે કે, આ પ્રોગ્રામ જે અમે આ અભ્યાસક્રમ પર સંશોધન કરવા જઈ રહ્યા છીએ અને જે બધા પ્રોગ્રામરો સામાન્ય રીતે કામ કરે છે તે ઊંચા સ્તરે હોય છે, જેને કોમ્પ્યુટર દ્વારા સીધી રીતે સમજી શકાય તેમ નથી, તેથી અમે કોઈક આ અંતરને તોડવું પડશે. કમ્પાઇલર એક પ્રોગ્રામ છે જે ઉચ્ચ સ્તરની પ્રોગ્રામિંગ ભાષા લે છે અને તે ભાષા પ્રોગ્રામિંગ ભાષા માં પ્રોગ્રામિંગ ભાષા માં પ્રોગ્રામ કરે છે. તેથી, તે પાયથોન માં હાઇ લેવલ પ્રોગ્રામ લે છે, જો સીટીમાં નહીં, સી અથવા સી++ અથવા જાવા અથવા કોઈ વસ્તુમાં અને સીધી મશીન સાથે એક્ઝેક્યુટ કરી શકે છે. બીજી તરફ, ઉચ્ચ-સ્તરની ભાષા સાથે વ્યવહાર કરવાની અન્ય

રીત તે અર્થઘટન કરવાનો છે. તેથી, સામાન્ય રીતે અંગ્રેજીમાં દુભાષિયો એ કોઈ વ્યક્તિ છે જે જુદા જુદા ભાષાઓની વાત કરતા લોકો વચ્ચે રહે છે અને પાછળથી અનુવાદ કરે છે. એક દુભાષિયો એ એક પ્રોગ્રામ છે જે તમે સાથે ક્રિયાપ્રતિક્રિયા કરો છો, અને તમે તમારી ભાષામાં દુભાષિયો સૂચનોને ફીડ કરો છો, આ કેસમાં પાયથોન; અને ઈન્ટરપ્રીટર અંતર્ગત બહાર પાડે છે કે કેવી રીતે તેમને નીચે લીટી પર ચાલવું. તેથી, તમે તેને વિન્ડોઝ, અથવા મેક, અથવા લિનક્સના દુભાષિયા પર ચલાવી રહ્યા છો તે ગેરંટી આપે છે કે ઉચ્ચ સ્તરે તમે જે જવાબ જુઓ છો તે લગભગ વાસ્તવિક પ્લેટફોર્મથી સ્વતંત્ર છે જે તમે તેને ચલાવી રહ્યા છો. તેથી, પાયથોન અને મોટી અર્થઘટનવાળી ભાષા છે અને આપણે આ હકીકતથી પરિચિત હોવા જોઈએ.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 07:31)

અમે અજોડ રીતે નીચેની રીતે ઉપયોગ કરીએ છીએ; અમે પ્રથમ દુભાષિયા ચલાવો તેથી, યાદ રાખો કે દુભાષિયો કાર્યક્રમ છે. અમે પ્રથમ દુભાષિયા જગાવીએ છીએ; અને જ્યારે દુભાષિયો ચાલતું હોય ત્યારે, આપણે અજમાયશી આદેશો ચલાવવા માટે દુભાષિયોને પસાર કરીએ છીએ. દુભાષિયા સાથે વ્યવહાર કરવા માટેની સરસ વસ્તુ તે છે કે તમે કેલ્ક્યુલેટર સાથે રમી શકો છો. તમે તેને આદેશો ફીડ કરી શકો છો અને જુઓ કે તે શું કરે છે, તેથી તે ખૂબ જ અરસપરસ છે. અલબત્ત, તે કંટાળાજનક છે, જો તમારે મોટી પ્રોગ્રામ્સ લખવાનું હોય, તો એક પ્રોગ્રામ લાવવાનો એક માર્ગ છે જે પહેલાથી જ સ્ટાન્ડર્ડ ટેક્સ્ટ એડિટરનો ઉપયોગ કરીને અને ફાઇલમાંથી લોડ થઈ રહ્યો છે. તેથી, નીચે બતાવેલ હરોળમાં મેં બતાવ્યું છે કે જેથી આ એક મિનિટે આપણે જોઈશું જે દુભાષિયો તમને બતાવે છે. જ્યારે તમે દુભાષિયો દાખલ કરો છો, ત્યારે તે તમને આદેશ ચલાવવા માટે પૂછશે અને આ એક આદેશ છે જે તમે દુભાષિયોને પ્રદાન કરો છો. તે કહે છે તો મેં સંગ્રહિત કર્યું છે. મારી પાસે એક ફાઇલ છે જેને name name dot py કહેવાય છે, ખાસ કરીને તે એ ફાઇલ આયાત કરવાથી પાયથોન પ્રોગ્રામ છે, જે બધી વ્યાખ્યાઓ અને કાર્યો અને કોડ છે જે અહીં લખાયેલ છે. તેથી, આ દુભાષિયોને તે કોડમાં લખેલા દરેક વસ્તુને લઈને તેના વર્તમાન વાતાવરણમાં મૂકશે, જેથી તે કાર્યોનો ઉપયોગ કરી શકાય. તેથી, આ વસ્તુઓ થોડી સ્પષ્ટ થઈ જશે અને પછી ડેમોમાં હું તમને બતાવીશ અને પછી તમે આ સાથે આસપાસ રમી શકો છો. અને પછી આગળના અઠવાડિયે, આપણે અજોડ વિગતોમાં જઈશું જે પાયથોન પ્રોગ્રામમાં જાય છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 09:16)

અહીં એક વિન્ડો છે જે ટર્મિનલને દર્શાવે છે જે વિન્ડોઝ પર આદેશ પ્રોમ્પ્ટની જેમ અને અનન્ય જેમ શેલનો ઉપયોગ કરશે. તેથી, જો હું 1 ના કહું, તો તે મને મારા વર્તમાન વિસ્તારની ફાઇલોની યાદી બતાવે છે. અને એક્સ્ટેન્શન ડોટ પાઇ સાથેની બધી ફાઇલો ખરેખર પિથન પ્રોગ્રામ્સ છે. આમાં, હું પાયથોન 3.5 કહીને પાયથોન ઇન્ટરપ્રીટરને બોલાવી શકું છું કારણ કે તે વર્ઝન જે હું વાપરી રહ્યો છું. જો હું તેને બોલાવતો હોઉં, તો તે કેટલાક સંદેશા ઉત્પન્ન કરશે જે મને બતાવશે કે કયા પ્રકારની ફક્શન સિસ્ટમ છે. તેથી, તે મને કહે છે કે હું દાખલા તરીકે 3.5.2 નો ઉપયોગ કરી રહ્યો છું અને તે કદાચ એકદમ તાજેતરની આવૃત્તિ છે, તે મને કહે છે કે તે એક એપલ અને મૂર્ખામી ભરેલી વાહિયાત વાત મૂર્ખામી ભરેલી વાહિયાત વાત પર હોય છે, પરંતુ પછી શું મહત્વનું છે પછી એક પ્રોમ્પ્ટ સ્થળ ઉત્પન્ન કરે છે જ્યાં હું આદેશો દાખલ કરી શકું છું અને આ ત્રણ મોટા સમય દ્વારા સૂચવાય છે. હવે, પિથન ઇન્ટરપ્રીટર પ્રોમ્પ્ટ પર, તમે સીધા વસ્તુઓ લખવાનું શરૂ કરી શકો છો. તેથી, ઉદાહરણ તરીકે, તમે કહી શકો છો કે હું ઇકવલ ટુ 5 છું. તે કહે છે કે નામ લેવા માટે હું 5 ની વેલ્યુ આપીશ. હવે જો હું ટાઇપ કરું તો તે મને કહે છે કે કિંમત 5 છે; જો હું આઇ વત્તા 1 જેવા અભિવ્યક્તિ લખું છું, તો તે મને કહે છે કે હું 6. તેથી, તમે તેને કેલ્ક્યુલેટર તરીકે વાપરી શકો છો. તેથી જો તમે ઇચ્છો તો તમે સરળ અંકગણિત કરી શકો છો. તેથી, તમે તેની સાથે ક્રિયાપ્રતિક્રિયા કરી શકો છો. હવે, તમે વિધેયો પણ વ્યાખ્યાયિત કરી શકો છો યાદ રાખો કે કેવી રીતે આપણે એક ફક્શન વ્યાખ્યાયિત કરી છે, આપણે def વાપરીએ છીએ, ફક્શન નામનો ઉપયોગ કરી શકીએ છીએ. તેથી, આપણે કહી શકીએ, DEF બે વખત x આ બે વાર કાર્ય છે, આ એક દલીલ x લે

છે. અને તમે આશા રાખી શકો કે હું બે વખત x પરત કરું છું. હવે એક પાયથોન એનો ઉપયોગ કરે છે, જેમ કે આપણે પહેલાના વ્યાખ્યાનો ઉલ્લેખ કર્યો છે, તે સ્પષ્ટ કરવા માટે કે કંઈક બીજાનું કંઈક છે. તેથી, વ્યાખ્યામાં તેના પગલાઓનો સમૂહ છે. તેથી, મને તે કહેવું જોઈએ કે પગલાંની આ જુમલો તે ઇન્ડેન્ટ કરીને આ વ્યાખ્યા પ્રમાણે છે; તે જ્યાં સુધી તમે સમાન ઇન્ડેન્ટેશનને એકસરખી રીતે ઉપયોગમાં લેતા હો ત્યાં સુધી તે કોઈ રીતે ઇન્ડેન્ટ કરતું નથી. જો તમે બે જગ્યાઓનો ઉપયોગ કરી રહ્યાં છો, તો બે જગ્યાઓ ટેબનો ઉપયોગ કરો, પરંતુ જગ્યાઓની સંખ્યાને ભેગી કરતા નથી અને ટેબ્સ અને સ્થાનોને ભેગવતા નથી, કારણ કે આ તમને ભૂલ સંદેશાઓ ફોર્મ પાયથોનને ગૂંચવણમાં મૂકે છે તો, ચાલો આપણે બે જગ્યાઓ વાપરીએ. ચાલો આપણે ઉદાહરણ માટે એક નવું નામ y બનાવીએ, અને y એ બે વખત x છે. હવે તે હજુ પણ વ્યાખ્યાઓ માટે મને પૂછવા માટે ચાલુ છે, તેથી પ્રોમ્પ્ટને ડોટ ડોટ ડોટ પર બદલવામાં આવ્યું છે. હવે મને તે જ રીતે દાખલ કરવું જોઈએ અને વળતર y કહેવામાં આવે છે. તેથી, મેં શું કર્યું છે તે હું કહું છું કે આ ઇન્ક્શન વેલ્યુ x માં લે છે, એક્સ 2 ની ગણતરી કરે છે, અને તેને y નામમાં સંગ્રહ કરે છે, અને y નું મૂલ્ય y આપે છે, જમણે હવે, જ્યારે હું આ સાથે કરીશ, હું એક ખાલી રેખા આપું છું અને આ કાર્ય હવે વ્યાખ્યાયિત થયેલ છે. હવે, બે વખત 7 અર્થમાં છે, જે બે વાર 932 થી પણ બરાબર જણાય છે. તેથી, પાયથોન ખૂબ જ અનુકૂળ છે, કારણ કે તમે ફ્લાય પર જતા હોવ ત્યારે તમે થોડા વ્યાખ્યાયિત કાર્યો કરી શકો છો. હમણાં, અમે અહીં અમારા જુસીડીને પણ વ્યાખ્યાયિત કરી શકીએ છીએ, પરંતુ જેમ તમે કદાચ અપેક્ષા રાખી શકો કે કોઈ ભૂલ વગર વગર ટાઇપ કરવા માટે કોઈ કાર્ય ખૂબ જટિલ છે, અને બીજું, તમે કાર્ય સાથે ફરતે રમી શકો છો અને તેને બદલી શકો છો અને તેને રાખવાની જરૂર નથી. તે ફરીથી અને ફરીથી લખીને આ માટે, આપણે શું કરવાની જરૂર છે તે પહેલા ફાઇલમાં ઇન્ક્શનને ટાઇપ કરો અને પછી ફાઇલને અહીં લોડ કરો. ચાલો આમાંથી નીકળી જઈએ. તેથી, આમાંથી બહાર નીકળવાનો એક માર્ગ એ છે કે કૌંસ છોડી દો.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 12:49)

અને પછી તમે આ પ્રોમ્પ્ટ પર પાછા આવો છો જે ડોલર છે જે બહારના ટર્મિનલ છે અથવા કમાંડ પ્રોમ્પ્ટ છે. તો, મેં પહેલાથી જ કંઈક બનાવ્યું છે ચાલો આપણે પ્રારંભ કરીએ, તો હું એમએએક્સ તરીકે ઓળખાતી એડિટરનો ઉપયોગ કરું છું, જો તમે વિન્ડોઝનો ઉપયોગ કરી રહ્યા હોવ તો તમે કોઈપણ લે એડ એડિટરનો ઉપયોગ કરી શકો છો, તમે નોટપેડ વાપરી શકો છો, જો તમે ઉપયોગ કરી રહ્યા છો અને લીનક્સ, તો તમે ઇમેક્સ અથવા વી વાપરી શકો છો અથવા તમે કેટલાક સરળ એડિટરનો ઉપયોગ કરી શકો છો. જેમ કે જુએડિટ કે કે, જે કંઈપણ આરામદાયક છે, પરંતુ તે ફક્ત ટેક્સ્ટ એડિટર જ હોવી જોઈએ, તે કોઈપણ ફોર્મેટિંગ ન કરવું જોઈએ, જેમ કે કાર્યાલય કાર્યવાહીનો ઉપયોગ કરશો નહીં જેમ કે તમે કાર્યાલય જાણો છો અથવા આના જેવી કઈ. તમે કંઈક કે જે અમે ફક્ત લખાણ ફાઇલો manipulates.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 13:27)

જો હું જુસીડી 1 ડોટ પીઈ પર નજર કરું છું, તેથી એક સરસ વસ્તુ જે ઇમેક છે તે ચોક્કસ વસ્તુઓ સૂચવવા માટે મને રંગો બતાવે છે. તેથી, def આ ખૂબ જ પ્રથમ જુસીડી પ્રોગ્રામ છે જે આપણે લખ્યું હતું, જે યાદી એફએમની ગણતરી કરે છે, પછી યાદી એફએમ પછી યાદી સીએફ, અને પછી તે સીએફમાં છેલ્લો ઘટકો આપે છે. તેથી, આ જુસીડીનું પ્રથમ વર્ઝન છે. તેથી, આ તે કોડ છે જે આપણે પહેલાં લખ્યું હતું. યાદ રાખવાનો મુદ્દો એ છે કે મેં ખાતરી કરી લીધી છે કે આ તમામ ઇન્ડેન્ટેશન એ જ સંખ્યામાં જગ્યાઓ છે. તેથી, આ યાદ રાખવાનું છે હવે, તમે આની જેમ કંઈક ટાઇપ કરી રહ્યા છો, તો પછી તમે તેને સંગ્રહો અને બહાર નીકળો

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 14:03)

હવે તમે તમારા પાયથોન માં પાછા જાઓ, અને તમે તે ફાઇલ જુસીડી 1 આયાત સ્ટારથી સંગ્રહિત કરો, જેનો અર્થ એ થાય કે ફાઇલ gcd1 ડોટ પીઇ લે છે અને તમામ કાર્યો લોડ કરે છે જે ત્યાં વ્યાખ્યાયિત અને તેમને ઉપલબ્ધ કરાવે છે. અહીં મને હમણાં, જો હું કહીશ કે 7 અલ્પવિરામનો જુસીડી અમને ઉદાહરણ તરીકે દોરી જાય છે, દાખલા તરીકે, 14 અને 63, તો તે મને જુસીડી 7 કહે છે. હવે જો તમે 9999 અને 10000 જેવી મોટી સંખ્યામાં લેતા હોવ તો તે એક વધુ આંકડો છે. જુઓ, તમે જાણશો કે તે મને જવાબ આપતો નથી અને પછી મને જવાબ આપે છે. તેથી, તે આ ફક્ત સમજાવવા માટે છે કે આ ધીમા જુસીડી અધિકાર છે. તેથી, તે કેટલો સમય લીધો તે જુઓ. તે જવાબ ઉત્પન્ન કરે તે પહેલાં તે થોડી સેકન્ડના દૃશ્યક્ષમ તફાવત ધરાવે છે. અને આ ઉદાહરણ છે કે આ એક ખૂબ કાર્યક્ષમતા જુસીડી નથી. તેથી, આ પાયથોન દુભાષિયા સાથેની એક સમસ્યા જે હું જોઉં કે જો આપણે હલ કરી શકીએ તો તે છે કે જો મેં પહેલેથી જ એક ફાઇલ લોડ કરી છે તો તે બહાર નીકળવા માટે સલામત છે અને પછી ફાઇલને અપડેટ કરતાં અન્ય ફાઇલને ફરીથી લોડ કરો.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 15:25)

ચાલો હું યુક્લિડની વાતનો છેલ્લો સંસ્કરણ પાછો ફરીથી લોડ કરું, જે આપણે લખ્યું છે તે રીમાઇન્ડર વર્ઝન છે.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 15:32)

તે કહે છે કે જો એમ પછી ઓછા મૂલ્યોનું મૂલ્ય બદલાશે તો જો બીજી લાઈન અહીં કહે છે કે જો એમની વહેંચણી એન દ્વારા કરવામાં આવે તો તે 0 છે, તે m નો ભાજક છે અને પછી અન્યથા પરત આવે છે. કોલ માટે n અને તેના સ્મૃતિપત્ર દ્વારા gcd કોલ બદલો. તેથી, આ અમારી પાસે એક વર્ઝન પણ છે જ્યાં આપણે લૂપ પર પાછા આવીએ છીએ. ચાલો આ વર્ઝનનો ઉપયોગ કરીએ. જ્યારે આવૃત્તિ કહે છે કે જ્યાં સુધી સ્મૃતિપત્ર 0 ન હોય ત્યાં સુધી, અમે એમ અને n ને n અને રીમાઇન્ડરને અપડેટ કરતા રાખીએ છીએ, અને છેલ્લે તમે n ની વેલ્યુ પરત કરો છો.

(સ્લાઇડ સમયનો સંદર્ભ લો: 16:13)

હું આ ચોક્કસ વસ્તુ લેવાનો છું અને તેને પાયથોન માં લોડ કરું છું. તેથી, ફરી હું પ્રથમ દુભાષિયો પાયથોન નો ઉપયોગ કરું છું, પછી હું gcdEuclid2a આયાત સ્ટારથી કહું છું. હવે હું તે જ મોટા મૂલ્ય આપવા જઈ રહ્યો છું જે આપણે પહેલા જોયું અને જે મને લાગે છે 9999999 અને 1000000 કહે છે. અને હવે તમે જુઓ છો, તમને ઝટપટ જવાબ મળે છે. વાસ્તવમાં, તમે જોશો કે જો હું તેને વધુ સંખ્યામાં આપીશ તો પણ, તે સંપૂર્ણપણે ઝડપી કામ કરવાની આશા રાખવી જોઈએ. તેથી, ઝડપમાં એક નાટ્યાત્મક સુધારો છે જે આ સરળ ઉદાહરણમાં પણ દૃશ્યમાન છે, જો આપણે એક ચપળ વિચાર દ્વારા નિષ્કપટ વિચારને બદલીએ છીએ. એલ્ગોરિધમની શક્તિ વાસ્તવમાં એક પ્રોગ્રામ બનાવવી છે જે અન્યથા એક ગતિમાં નામાંકિત ધીમું કાર્ય હશે જે તમને સ્વીકાર્ય છે. તમારી સિસ્ટમ પર લોડ પેથન કરો, પાયથોન ના દુભાષિયાને અજમાવો અને આ અઠવાડિયાની વસ્તુમાં જોઈ હોય તે કોડ સાથે ફરક કરો, ભૂલોને જુઓ, જ્યારે તમે ફાઇલને આયાત કરો છો કે જેમાં ભૂલો છે દાખલા તરીકે, જો હું એહનો પ્રયાસ કરું છું જે એક ફંક્શનનો ઉપયોગ કરે છે જે અસ્તિત્વમાં નથી, જો હું કોઈ ફંક્શનનો ઉપયોગ કરું છું જે મેં વ્યાખ્યાયિત કર્યું નથી અને જે પાયથોન ને સમજી શકતો નથી તો તે મને આની જેમ ભૂલ આપશે. તે કહેશે લૂપ વ્યાખ્યાયિત નથી. જો હું કંઈક 5 કરતાં મોટી કરતાં 7 જેટલા વિચિત્ર લખું છું, તો તે કહેશે કે આ અયોગ્ય સિન્ટેક્સ છે. દુભાષિયો જો અભિવ્યક્તિનો અર્થ નથી કરતી તો અભિવ્યક્તિની તપાસ કરશે, તો તે ફરિયાદ કરશે. અને ક્યારેક ભૂલ સંદેશાઓ સમજવા માટે સરળ હોય છે, ક્યારેક તે સમજવામાં ઓછા સરળ હોય છે; જેમ આપણે આગળ વધીએ છીએ તેમ આપણે આમાં તપાસ કરીશું. પરંતુ, દુભાષિયોનો ઉદ્દેશ ક્યાં છે તે તમે ચલાવો છો તે તમને ચલાવવા માટે અથવા તમે કહો છો કે તમે જે

લખ્યું છે તે કોઈક એક્ઝેક્યુટેબલ નથી અને શા માટે સમજાવે છે. તેથી, તેની સાથે આસપાસ રમી દો અને કેટલાક પારિવારિકતા મેળવો કારણ કે આ અમારી બ્રેડ અને માખણ બની રહ્યું છે કારણ કે આપણે આગળ વધીએ છીએ.

(સ્લાઈડ ટાઇ નો ઉલ્લેખ કરો: 18:18)

આપણે આ અભ્યાસક્રમમાં પાયથોન ના અમુક વિશિષ્ટ લક્ષણો પર વિચારવું જઈ રહ્યા છીએ, પરંતુ આપણે ત્યાં જઈ શકીએ છીએ કે ત્યાં કંઈક છે જેને તમે સમજી શકતા નથી અથવા કંઈક નવું જે તમે ઈચ્છો છો તમારા પોતાના પ્રયાસ કરો. તેથી, અન્ય સ્રોતોનો વપરાશ કરવાનો વિચાર હંમેશા સારો છે પાયથોન ઓનલાઈન દસ્તાવેજીકરણ વાસ્તવમાં એક અજોડ સ્થળ છે જે પાયથોન ની વિગતો અને ખાસ કરીને, ખૂબ વાંચનીય ટ્યુટોરીયલ છે; ખાસ કરીને, જો તમારી પાસે પહેલેથી જ પાયથોન પ્રોગ્રામીંગ સાથે અજોડ છે તો તમારા માટે પાયથોન શીખવા માટે કદાચ શ્રેષ્ઠ સ્થાન છે. તો, અહીં URL છે, docs.python.org/3 આ પાયથોન 3 ટ્યુટોરીયલ index dot html માટે છે. જો તમે હમણાં જ docs.python.org/3 પર જાઓ છો, તો તમને વધુ વિગતવાર સંદર્ભ માર્ગદર્શિકાઓ પણ મળશે, જે તમને પછીના તબક્કે જરૂર પડી શકે છે. જ્યારે તમે મુશ્કેલીઓ અનુભવો છો ત્યારે તમે તે સ્થાનોમાંથી એક તરીકે રાખો છો. અને ત્યાં બે પુસ્તકો છે જે સંભવતઃ ઉપયોગી છે કે જે અજમાયશમાં શું આવરી લેવામાં આવે છે તે ઉપરાંત પાયથોન ને સમજવા માટે જો તમને લાગે કે કંઈક સ્પષ્ટ નથી. તેથી, આ પુસ્તક પાયથોન માં ડાઇવ કહેવાય છે જે પાયથોન 3 માટે અનુકૂળ છે. અને ત્યાં અજાયબ પાયથોન કહેવાય પુસ્તક છે જે સામાન્ય રીતે પેટેનની સંદર્ભમાં કોમ્પ્યુટેશનલ વિચારધારા વિશે છે. આ બન્નેનો સરસ લાભ છે કે તેઓ ઓનલાઈન ઉપલબ્ધ છે, તેથી તમારી પાસે ખરીદો કંઈપણ નથી; તમે તેને નેટ પર તમારા બ્રાઉઝર દ્વારા બ્રાઉઝ કરી શકો છો.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 19:41)

આ અઠવાડિયાની બહાર જવા પહેલાં, યાદ રાખો કે શીખવાની પ્રોગ્રામિંગ એક પ્રવૃત્તિ છે; તમે પ્રોગ્રામિંગ સૈદ્ધાંતિક રીતે શીખવા શકતા નથી. વિષયની પ્રશંસા કરવા માટે તમારે કોડ લખવો અને અમલ કરવો પડશે. તમારે ભૂલો કરવી પડશે; તમારી ભૂલોથી શીખો; શું કામ કરે છે તે જાણો, શું કામ કરતું નથી અને માત્ર પછી તમે પ્રોગ્રામિંગ માટે સાચી પ્રશંસા મળશે. કારણ કે અમે પાયથોન સાથે જઈ રહ્યા છીએ કારણ કે પાયથોન પાસે અન્ય પ્રોગ્રામિંગ લેંગ્વેજ્સની સરખામણીમાં ખૂબ સરળ વાક્યરચના છે. અમે પહેલાથી જ ઔપચારિક રીતે શીખવાની પાયથોન વિના, જુસીડી માટે કેટલાક સારા કાર્યક્રમોને જોયા છે અને આશા છે કે તમે તેમને ઉભો કરી શકતા ન હોવા છતાં પણ. સમજવામાં થોડુંક સમજણ સાથે પાયથોન પ્રોગ્રામ શું કરી રહ્યું છે તે સમજવું ખૂબ જ મુશ્કેલ નથી. એવા ઉદાહરણોનું પ્રેક્ટિસ કરવા માટે સમય કાઢો કે જે અમે આ સમયને જોયો છે. આપણે પ્રોગ્રામિંગ કસરત આપીશું કારણ કે આપણે આગળ વધીશું; અને જ્યાં સુધી તમે આ કસરતો કરતા નથી અને પોતાને પાયથોન ને લલચાવીને કંઈક સહેલું થઈ જાઓ છો, તમે પ્રોગ્રામિંગ અને પાયથોન બંનેને ખરેખર ક્યારેય શીખશો નહીં. યાદ રાખવા માટેની બીજી વાત એ છે કે એકવાર તમે એક ભાષા શીખ્યા હોય, છતાંય લક્ષણો અને વાક્યરચના ભાષાથી લઈને અલગ અલગ હોય છે, તે બીજી ભાષા પસંદ કરવા માટે ખૂબ જ સરળ છે, કારણ કે તેની બધી પ્રોગ્રામિંગ તેના પર આધારિત છે. . તેમ છતાં વાક્યરચના અલગ અલગ હોઈ શકે છે, વિચારો નથી. વિચારો આખરે પ્રોગ્રામ શું લખે છે, પરંતુ પ્રોગ્રામિંગ ભાષા નો અસ્ખલિત સ્પીકર હોવો જોઈએ, તમારે તેને પ્રેક્ટિસ કરવું જોઈએ. તેથી, પ્રયાસ કરો.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રોફેસર. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
અઠવાડિયું - 02 વ્યાખ્યાન - 01

સૌપણી નિવેદન, મૂળભૂત પ્રકારો - ઈન, ફ્લોટ, બૂલ છેલ્લા અઠવાડિયે, અમે નોટેશનની રજૂઆત કરવામાં આવી હતી. Uct ઉદાહરણનો ઉપયોગ કરીને ગાણિતીક નિયમો. અમે અનૌપચારિક રીતે કેટલાક પાયથોન કોડ જોયો, જે અમે સમજી શકીએ પરંતુ અમે વાસ્તવમાં ઔપચારિક પિથન સિન્ટેક્સમાં દાખલ થયા નથી. ચાલો આપણે કેટલાક વાસ્તવિક પાયથોન પગલા સાથે શરૂઆત કરીએ.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 00:17)

એક લાક્ષણિક પાયથોન પ્રોગ્રામ આ રીતે લખવામાં આવશે, અમારી પાસે એક્ઝેક્યુટ કરવા માટેનાં વિધેયોના સમૂહ દ્વારા અનુસરવામાં કાર્ય વ્યાખ્યાઓનો સમૂહ છે. તેથી યાદ રાખો કે અમે કહ્યું છે કે પાયથોન ને સામાન્ય રીતે અર્થઘટન કરવામાં આવે છે, તેથી દુભાષિયો એ એક પ્રોગ્રામ છે, જે પાયથોન કોડને વાંચશે અને ઉપરથી નીચે સુધી ચલાવશે. તેથી, દુભાષક હંમેશા તમને પાયથોન કોડની શરૂઆતથી શરૂ કરે છે અને ઉપરથી નીચે સુધી વાંચે છે હવે, ફંક્શનની વ્યાખ્યા એક પ્રકારનું નિવેદન છે, પરંતુ તે વાસ્તવમાં કંઈ પણ થતું નથી, પિથન દુભાષિયો ફક્ત ફંક્શનની જેમ જ કાર્ય વ્યાખ્યાને યાદ રાખે છે. જેથી પછીથી જો કોઈ વાસ્તવિક નિવેદન આ કાર્યને સંદર્ભિત કરે છે તો તે શું કરવું તે જાણે છે. આ પ્રકારની સંસ્થામાં એક્ઝેક્યુશન વાસ્તવમાં સ્ટેટમેન્ટથી શરૂ થશે, જે સ્ટેટમેન્ટ 1 કહેવાય છે. તેથી, પ્રથમ તમારે k ફંક્શનને ડાયજેસ્ટ કરીશું અને પછી સ્ટેટમેન્ટ n સુધી 1, 2, 3, 4 નો અમલ શરૂ કરીશું.

(સ્લાઇડ સમયનો સંદર્ભ લો: 01: 09)

હવે આ કરવા માટે કોઈ કારણ નથી. તેથી, પાયથોન વાસ્તવમાં તમને ફંક્શનની વ્યાખ્યાઓ અને નિવેદનોને મુક્તપણે મિશ્રિત કરવાની પરવાનગી આપે છે, અને વાસ્તવમાં, કાર્યની વ્યાખ્યાઓ એક પ્રકારનાં નિવેદનો પણ છે, ફક્ત તેનું પરિણામ તરત જ બનતું નથી, પરંતુ કાર્યમાં યાદ રાખવામાં આવે છે. પરંતુ એક એવી વસ્તુ છે જે પાયથોન ને આગ્રહ રાખશે કે જો કોઈ વિધેયને અમલમાં મૂકવા માટે વિધેયનો ઉપયોગ કરવામાં આવે તો તે કાર્ય પહેલાથી વ્યાખ્યાયિત હોવું જોઈએ; ક્યાં તો તે એક બિલ્ટ ઇન વિધેય હોવી જોઈએ અથવા તેની વ્યાખ્યા પૂરી પાડવામાં આવશ્યક છે. તેથી, તે આ પ્રકારની જમ્બોલ્ડ અપ ઓર્ડરનો ઉપયોગ કરી શકે છે, અમારે સાવચેત રહેવું જોઈએ કે તેનો ઉપયોગ કરવામાં આવે તે પહેલાં વિધેયો વ્યાખ્યાયિત થાય છે. આ રીતે નિવેદનો અને વિધેય વ્યાખ્યાઓનો ક્રમ જોતાં, તે પ્રોગ્રામ વાંચવા અને તે શું કરી રહ્યું છે તે સમજવા માટે ખૂબ મુશ્કેલ બનાવે છે. જોકે તે પાયથોન દ્વારા જરૂરી નથી, જેમ કે, તે ખૂબ આગ્રહણીય છે કે તમામ કાર્ય વ્યાખ્યા પ્રોગ્રામના શીર્ષ પર મુકવી જોઈએ અને કોડના મુખ્ય ભાગમાં રચાયેલા તમામ નિવેદનો પછીથી અનુસરવા જોઈએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 02:07)

એક નિવેદન શું છે? પિથનમાં સૌથી મૂળભૂત નિવેદન એ એક નામ મૂલ્ય અસાઇન કરવા માટે છે. તેથી, આપણે ઉદાહરણો જોઈએ છીએ અને આપણે ઉદાહરણો જોયા છે અને અહીં કેટલાક ઉદાહરણો છે. તેથી, પ્રથમ નિવેદનમાં હું એક નામ છે અને તેને મૂલ્ય 5 આપવામાં આવ્યું છે; બીજા નિવેદનમાં, j એ અલગ નામ છે અને તે 2 વખત i તેથી, આ અભિવ્યક્તિમાં, હું અહીં અભિવ્યક્તિ માટે અવેજીમાં મુકવામાં

આવશે. તેથી, જો મને પહેલાથી પહેલાં કોઈ મૂલ્ય ન સોંપવામાં આવ્યું હોય, તો પાયથોન ને ખબર નહીં પડે કે હું શું કરીશ અને તે ભૂલ તરીકે ફલેગ કરવામાં આવશે. જ્યારે તમે જમણી બાજુના નામનો અભિવ્યક્તિના ભાગ તરીકે ઉપયોગ કરો છો, ત્યારે તમારે ખાતરી કરવી પડશે કે તે પહેલાંથી માન્ય મૂલ્ય ધરાવે છે. અને જેમ આપણે જોયું તેમ, તમારી પાસે નિવેદનો પણ હોઈ શકે છે જે ફક્ત મૂલ્યને અપડેટ કરે છે. તેથી, જ્યારે આપણે કહીએ કે j જ બરાબર j વત્તા 5 તે એક ગાણિતિક નિવેદન નથી, જ્યાં j ની મૂલ્ય j વત્તા 5 ની બરાબર છે. પરંતુ તેના બદલે જમણી બાજુએ જે j નું જૂનું મૂલ્ય છે તેને 5 ઉમેરીને અપડેટ કર્યું અને પછી તે નવી મૂલ્ય j તરીકે બદલાઈ જાય છે. આ અસાધનમેન્ટનું નિવેદન છે, આ સમાનતા મૂલ્ય ગણતરી કરેલ ફોર્મને જમણા હાથમાં સોંપે છે જે તમામ નામોની વર્તમાન કિંમતોને આપવામાં આવે છે જો નામ ડાબી બાજુએ આપવામાં આવ્યું હોય, તો તે જ નામથી બંને બાજુએ દેખાશે. ડાબી બાજુ એક નામ છે અને સામાન્ય રીતે જમણા હાથમાં એક અભિવ્યક્તિ છે. અને અભિવ્યક્તિમાં, તમે એવી વસ્તુઓ કરી શકો છો કે જે કાનૂની છે, જે અભિવ્યક્તિમાં મૂલ્યોના પ્રકારો આપેલ છે. તેથી, કિંમતો પ્રકારો છે; જો તમારી સંખ્યા હોય, તો તમે અંકગણિત કામગીરી કરી શકો છો; જો તમારી પાસે કેટલીક અન્ય વસ્તુઓ છે, તો તમે અન્ય કામગીરી કરી શકો છો. તેથી, કયા ઓપરેશન્સ માન્ય છે તે મૂલ્યો પર આધાર રાખે છે અને આ તકનીકી રીતે નામ પ્રકાર આપવામાં આવે છે. તેથી, જ્યારે આપણે કિંમતોના પ્રકારો કહીએ છીએ ત્યારે તે વાસ્તવમાં સ્પષ્ટ કરે છે કે મૂલ્યોના તે વર્ગ પર કયા પ્રકારના કાર્યો કાનૂની રીતે ઉપલબ્ધ છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 04:01)

તેથી મોટાભાગના મૂળભૂત પ્રકારનું મૂલ્ય જે કોઈ વિચારે છે તે સંખ્યાઓ છે. હવે પાયથોન માં અને મોટાભાગની પ્રોગ્રામિંગ ભાષા માં નંબરો બે વિશિષ્ટ સ્વરૂપોમાં આવે છે કારણ કે તમે તેમને પૂર્ણાંક અને સંખ્યાઓ જેને આંશિક ભાગો હોય તેને કહી શકો છો. તેથી, પાયથોન માં આ બે પ્રકારોને પૂર્ણાંક અને ફ્લોટ કહેવામાં આવે છે. તેથી, પૂર્ણાંક સંખ્યાને દર્શાવે છે, જેનો કોઈ દશાંશ ભાગ નથી, જેનો કોઈ આંશિક ભાગ નથી. તેથી, આ સંપૂર્ણ સંખ્યાઓ છે કે તેઓ નકારાત્મક હોઈ શકે છે. તેથી, આ પ્રકારનાં પૂર્ણાંકના મૂલ્યોના કેટલાક ઉદાહરણો છે. પ્રશ્નની બીજી બાજુએ, જો આપણે આંશિક ભાગો હોય તો આ પ્રકારના ફ્લોટના મૂલ્યો છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 04:43)

સામાન્ય રીતે ગણિતમાં આપણે વિશિષ્ટ વર્ગ તરીકે પૂર્ણાંકોને વિચારી શકીએ છીએ વાસ્તવિક સંખ્યાઓ તેથી, વાસ્તવિક સંખ્યાઓ આંશિક ભાગો સાથે મનસ્વી સંખ્યાઓ છે, પૂર્ણાંકો તે અસંખ્ય સંખ્યાઓ છે જે કોઈ આંશિક રીતે નથી. પરંતુ પ્રોગ્રામિંગ લેંગ્વેજમાં આ બે વચ્ચે એક વાસ્તવિક તફાવત છે અને તે છે, કારણ કે આ સંખ્યાઓ સંગ્રહિત છે. તેથી, જ્યારે પાયથોન ને મૂલ્યો યાદ રાખવા પડે છે ત્યારે તે આ મૂલ્યને કેટલાક આંતરિક સ્વરૂપમાં પ્રસ્તુત કરે છે અને આ માટે મર્યાદિત રકમની જગ્યા લેવાની રહે છે. જો તમે નીચે લખી રહ્યા હોવ તો, મેન્યુઅલ ઇન્ટેક્સ રકમ કહેશે કે તમે તેને કાગળના શીટ પર લખશો અને કાગળની શીટ અને તમારા હસ્તાક્ષરનાં કદ પર આધારીત એક ભૌતિક મર્યાદા હોઈ શકે છે કે તમે કેટલી સંખ્યામાં તે ઉમેરી શકો છો કાગળ શીટ એ જ રીતે કોઈપણ પ્રોગ્રામિંગ લેંગ્વેજ, સંખ્યાઓનું સંગ્રહ કરવા માટે અને કેટલાંક સંખ્યાઓનો ઉપયોગ કરે છે તે અગાઉ કેટલાંક પ્રોગ્રામિંગ ભાષાઓ આંતરિક રીતે દ્વિસંગી પ્રતિનિધિત્વનો ઉપયોગ કરશે તે નક્કી કરશે. તેથી, આપણે ધારણ કરી શકીએ છીએ કે પૂર્ણાંક અથવા વાસ્તવિક સંખ્યા એ દરેક સંખ્યા છે કે જે શૂન્યની મર્યાદિત શ્રેણી અને તેના મૂલ્યને રજૂ કરે છે. હવે, જો આ પૂર્ણાંક બને તો તમે બાઈનરી ક્રમને બાઈનરી નંબર તરીકે ગણશો જેમ તમે શાળામાં શીખ્યા હોત. તેથી, આંકડા 2 ની સત્તાઓ દર્શાવે છે, સામાન્ય રીતે ત્યાં એક વધારાનું દ્વિઅંકી આંક હશે 0 અથવા 1 એ સૂચવે છે કે તે વત્તા કે ઓછા છે અને તેઓ નકારાત્મક સંખ્યાઓ રજૂ કરવાના અન્ય વધુ કાર્યક્ષમ રીત હોઈ શકે છે, પરંતુ ખાસ કરીને તમે ધારણા કરી શકો છો કે પૂર્ણાંકો મૂળભૂત છે બાઈનરી નંબરો તેઓ બાયનરી નોટેશનમાં પૂર્ણાંક તરીકે લખાયેલા છે. હવે જ્યારે આપણે નોન-ઇન્ટિગ્રેજ પર

આવીએ છીએ ત્યારે આપણી પાસે બે મુદ્દાઓ છે, આપણે એ મૂલ્ય યાદ રાખવું જોઈએ જે અંકોની સંખ્યા છે જે આંશિક ભાગ બનાવે છે અને પછી આપણે સ્કેલ યાદ રાખવું પડશે. તેથી, વૈજ્ઞાનિક સંજ્ઞામાં એક નંબરનો વિચાર કરો, તેથી, જ્યારે આપણે ભૌતિક વિજ્ઞાન અને રસાયણશાસ્ત્રમાં વસ્તુઓનો ઉપયોગ કરીએ છીએ ત્યારે સામાન્ય રીતે બે ભાગ હોય છે, અમારી પાસે કિંમત છે જે મૂલ્યના ઘટકો છે અને અમારી પાસે આપણે કેવી રીતે પાળીશું તે દશાંશ ચિહ્નના સંદર્ભમાં. તેથી, આ કહે છે કે દશાંશ ચિહ્ન 24 અંકો જમણી તરફ ખસેડો. તેથી, આ પ્રથમ ભાગને મન્ટિસા અધિકાર કહેવામાં આવે છે અને તેને એક્સ્પોનન્ટ કહેવાય છે. તેથી, જ્યારે આપણી સંખ્યા એ મેમરીમાં છે, જો તે પૂર્ણાંક હોય, તો પછી સમગ્ર સ્ટ્રિંગને એક વેલ્યુ ગણવામાં આવે છે, જ્યારે આપણે અપૂર્ણાંકના બ્લોક ધરાવે છે. પછી અમારી પાસે તેના કેટલાક ભાગ છે, જે મન્ટિસા છે અને બીજા ભાગ છે, જે એક્સ્પોનન્ટ છે. દ્વિસંગી અંકોની તે જ અનુક્રમ જો આપણે તેને પૂર્ણાંક તરીકે વિચારીએ તો અલગ મૂલ્ય હોય છે અને જો આપણે તેને ફ્લોટ તરીકે વિચારીએ તો અલગ મૂલ્ય હોય છે. તેથી, તમે પૂછો કે શા માટે ફ્લોટ કરો ફ્લોટિંગ પોઇન્ટ માટે કોમ્પ્યુટર સાયન્સ માટે જૂનો શબ્દ છે; તે હકીકતનો ઉલ્લેખ કરે છે કે આ દશાંશ ચિહ્ન નિશ્ચિત નથી. તેથી પૂર્ણાંકને પૂર્ણાંકના અંતમાં ફિક્સ્ડ ડેઝિમમ પોઇન્ટ તરીકે ગણવામાં આવે છે, ફ્લોટિંગ બિંદુ નંબર એ ખરેખર એક સંખ્યા છે જ્યાં દશાંશ ચિહ્ન અલગ અલગ હોય છે અને તે કેટલી બદલાય છે તે ઘાત પર આધાર રાખે છે. તેથી મૂળભૂત રીતે મૂળભૂત તફાવત છે કે જે રીતે તમે પૂર્ણાંકોને પ્રતિનિધિત્વ કરો છો અને કોમ્પ્યુટરની અંદર ફ્લોટિંગ પોઇન્ટ નંબરો છે અને તેથી, બે વચ્ચે તફાવત હોવાનું સાવચેત રહેવું જોઈએ. તો, સંખ્યાઓ સાથે આપણે શું કરી શકીએ?

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 07:59)

અરે અમારી પાસે સામાન્ય અંકગણિત ઓપરેશન્સ વત્તા, બાદબાકી, ગુણાકાર છે, જેમાં પ્રતીક તારો આધુનિક x અને વિભાગ છે, જેમાં પ્રતીક સ્લેશ છે. હવે નોંધ લો કે પ્રથમ ત્રણ ઓપરેશન્સ માટે તે ખૂબ જ સ્પષ્ટ છે જો મારી પાસે 2 ints હોય અને હું તેમને ગુણાકાર કરું અથવા તેમને ઉમેરો અથવા બાદબાકી કરું તો મને પૂર્ણાંક મળે છે. પણ મારી પાસે 2 ફ્લોટ છે, હું એક ફ્લોટ, ડિવિઝન મેળવશે, જો હું 7 કહેતો હોઉં અને 2 વડે ભાગતો હોઉં તો, દાખલા તરીકે, બન્ને ints છે તો મને 3 પોઇન્ટ 5 મળશે. હવે સામાન્ય પાયથોન માં તમે ints અને ફ્લોટ્સ ભળવા માટે પરવાનગી આપે છે, તેથી હું 8 વત્તા 2 બિંદુ 6 લખી શકો છો, તેમ છતાં ડાબી એ પૂર્ણાંક છે અને અધિકાર ફ્લોટ છે અને તે યોગ્ય રીતે મને 10 પોઇન્ટ આપશે. તે અર્થમાં પાયથોન એ હકીકત છે કે ફ્લોટ્સ છે પૂર્ણાંકનું સામાન્ય સ્વરૂપ તેથી, આપણે હંમેશાં બિંદુ 0 સાથે ફ્લોટ તરીકે પૂર્ણાંક તરીકે વિચારી શકીએ છીએ. તેથી, જો આપણે તે રીતે વિચારવું અને અભિવ્યક્તિ સાથે સમાવિષ્ટ કરવું હોય તો, આપણે ફ્લોટ પર પૂર્ણાંકને અપગ્રેડ કરી શકીએ છીએ, પરંતુ વિભાગ હંમેશા ફ્લોટ્સનું ઉત્પાદન કરે છે. તો, મિશ્રિત અભિવ્યક્તિના ઉદાહરણ તરીકે 7 ભાગ્યા 3 ગુણ્યા 5, જ્યાં મારી પાસે પૂર્ણાંક અને ફ્લોટ હોય અને આ વિભાગનો પરિણામ બે બિંદુ 0 અને 7 માં બે પરિણામો 3.5 થાય.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 09:15)

હવે કેટલાક ઓપરેશન્સ છે જ્યાં આપણે ઓપરેન્ડ્સના પૂર્ણાંક પ્રકૃતિને સાચવવા માગીએ છીએ. અમે વારંવાર એક gcd માં જોયું છે જે મોડ્યુલસ ઓપરેટર છે, બાકી ઓપરેટર. પરંતુ રેકિમેન્ટર મારફતે પસાર થતા રીક સંલગ્ન ઓપરેટર એ ભાગભંડોળ ઓપરેટર છે. તેથી, જો હું ડબલ સ્લેશનો ઉપયોગ કરું છું તો તે મને ભાગ આપે છે. તો, 9 ડબલ સ્લેશ 5 કહે છે કે 5 બરાબર અપૂર્ણાંક વિના 9 બરાબર છે અને તે 1 છે, કારણ કે 5 ગુણ્યા 1 એ 5 અને 5 ગુણ્યા 2 બરાબર 10 જે 9 કરતા મોટો અને બાકી 4 છે. તેથી, 9 ટકા 5 એ 4 હશે. બીજું એક ઓપરેશન જે તદ્દન સ્વાભાવિક અને સામાન્ય છે તે એક નંબરને બીજા નંબર પર વધારવા માટે છે અને આ ડબલ સ્ટાર દ્વારા સૂચવાય છે. 3 ડબલ સ્ટાર 4 એ છે કે આપણે સામાન્ય રીતે 3 ની 3 ઘાત લખીશું, 3 ગુણ્યા 3 ગુણ્યા ચાર બરાબર છે અને આ 81 છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 10:12)

હવે લોગ, વર્ગમૂળ, પાપ અને બધા જે પાચથોન માં બનેલ છે, પરંતુ આ મૂળભૂત રીતે લોડ નથી થતાં. જો તમે પાચથોન અરસપરસ શરૂ કરો તો તમારે આને સ્પષ્ટપણે શામેલ કરવાનું રહેશે. યાદ રાખો કે અમે કહી શકીએ છીએ આયાત નિવેદનનો ઉપયોગ કરીને અમે લખીએ છીએ તે ફાઇલમાંથી કાર્યો શામેલ છે ગણિત તરીકે ઓળખાતા ગાણિતિક વસ્તુઓ માટે કાર્યોના સમૂહમાં બિલ્ટ છે. તેથી, આપણે ગણિતના આયાત સ્ટારથી ઉમેરવું જ જોઈએ; આ પાચથોન પ્રોગ્રામની અંદર પણ કરી શકાય છે, તે ફક્ત દુભાષિયા પર જ કરવું જરૂરી નથી. તેથી, જ્યારે આપણે પાચથોન પ્રોગ્રામ લખીએ છીએ જ્યાં આપણે લોગ, વર્ગમૂળ અને પાપનો ઉપયોગ કરવો છે અને આવા જેવા, તો આપણે આ ફંક્શનનો ઉપયોગ કરતા પહેલા ગણિતના આયાત સ્ટારમાંથી રેખાને ઉમેરવી જોઈએ.

(સ્લાઈડ ટાઇમ નો ઉલ્લેખ કરો: 10:58)

આપણે ત્રણ ખ્યાલો જોયાં છે - નામો, જે આપણે મૂલ્યો યાદ રાખવા માટે ઉપયોગ કરીએ છીએ, મૂલ્યો જે વાસ્તવિક સંખ્યાઓ છે જે અમે નામોને આપીએ છીએ અને અમે કહ્યું છે કે કોઈ પ્રકારની કલ્પના છે. તેથી, પ્રકાર નિર્ધારિત કરે છે કે આપણી પાસે જે મૂલ્યો છે તે આપણું કાર્યવાહી કાનૂની છે. તેથી, પાચથોન અને અન્ય ભાષાઓ વચ્ચેનું મુખ્ય તફાવત એ છે કે નામોમાં કોઈ પણ આંતરિક પ્રકાર નથી. હું અગાઉથી નથી કહેતો કે મારું નામ પૂર્ણાંક તરીકે અથવા નામ x ફ્લોટ છે. નામોની પાસે માત્ર તે જ પ્રકાર છે જે હાલમાં તેમની પાસે છે તે મૂલ્ય દ્વારા સોંપેલ છે. નામનો પ્રકાર નિશ્ચિત નથી. C અથવા C++ અથવા Java જેવી ભાષામાં અમે અમારા નામો અગાઉથી જાહેર કરીએ છીએ. અમે તેમને જાહેર કરીએ છીએ અને અગાઉથી કહીએ છીએ કે તેઓ કયા પ્રકારના છે. તેથી, જો આપણે એક અભિવ્યક્તિમાં હું જોઉં છું તો મને ખબર છે કે મને આ પ્રકારનું ઇન્ટ કહેવામાં આવ્યું હતું કે આ એક્સ પ્રકારનું ફ્લોટ અને તેથી પર જાહેર કરવામાં આવ્યું હતું. હવે પાચથોન માં આ કિસ્સો નથી.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 12:00)

તેથી, ચાલો આપણે આ ઉદાહરણને સમજાવીએ. તેથી, પાચથોન નું મુખ્ય લક્ષણ એ છે કે પ્રોગ્રામ વિકસિત થતાં એક નામને વિવિધ પ્રકારના મૂલ્યો સોંપવામાં આવી શકે છે. તો, જો આપણે અસાઇનમેન્ટ સાથે શરૂ કરીએ તો હું 5 બરાબર હોઉં છું કારણ કે 5 એ પૂર્ણાંકમાં એક પ્રકાર પૂર્ણાંક છે. હવે જો આપણે અભિવ્યક્તિ લઈએ, જે પૂર્ણાંક પેદા કરે છે જેમ કે 7 ગુણ્યા 1, હું પૂર્ણાંક રહે છે. હવે જો આપણે 3 ની કિંમતને ભાગાકાર કરીએ તો, આ બિંદુએ જો આપણે ક્રમ અનુસરતા હતા તો હું 7 છું. તેથી, 7 ભાગ્યા 3 એ 2.33 થશે અને આ એક ફ્લોટ હશે. એટલા માટે, કારણ કે આ બિંદુએ ફ્લોટમાં ઓપરેશનના પરિણામોને ફ્લોટ પ્રકારની કિંમત સોંપવામાં આવી છે. હવે જો આપણે કેટલાક પછીના તબક્કે ચાલુ રાખીએ તો અમે તેને લઈને મૂલ્ય 2 વખત જ આપીશું, કારણ કે j એ ફ્લોટ હતું હવે હું ફ્લોટ બની છું. દુભાષિયામાં એક ઉપયોગી પ્રકાર કહેવાય છે. તેથી, જો તમે શબ્દનો પ્રકાર લખો અને અભિવ્યક્તિ મૂકો અને કાં તો કોઈ નામ અથવા એક એક્સેન્સને કૌંસમાં, તો તે તમને વાસ્તવમાં અભિવ્યક્તિનો પ્રકાર જણાવશે. હવે જ્યારે પ્રોગ્રામ ઉત્પન્ન થતો હોવાથી પાચથોન નામના મુલ્યના પ્રકારને બદલવાની આ સુવિધાને મંજૂરી આપે છે, આ એવી કોઈ વસ્તુ નથી જે ભલામણ કરવામાં આવે. કારણ કે જો તમે એક અને ક્યારેક તેની ફ્લોટ જુઓ છો અને ક્યારેક તેની પૂર્ણાંક છે તો પ્રોગ્રામર તરીકે અને તમારા કોડને સમજવાનો પ્રયાસ કરનાર વ્યક્તિ માટે તે ફક્ત તમને ગૂંચવણમાં મૂકે છે. તે પહેલાં આપણે કહ્યું હતું તે જ રીતે અમે અમારા પાચથોન કોડને ગોઠવવા માંગીએ છીએ જેથી અમે નિવેદનો અમલમાં મુકતા પહેલાં તમામ કાર્યોને વ્યાખ્યાયિત કરીએ, તમારા ધ્યાનમાં ઓછામાં ઓછા, અલગ અલગ નામ કેવી રીતે ઊભા થાય છે અને વળગી રહો આનો ઉપયોગ અસંખ્ય તરીકે અથવા તો ફ્લોટ્સ તરીકે થાય છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 13:54)

ચાલો આપણે અમુક કોડ એક્ઝેક્યુટ કરીએ અને તપાસ કરીએ કે આપણે જે કહી રહ્યા છીએ તે વાસ્તવમાં બને છે. તેથી, જો આપણે પાચથન ઇન્ટરપ્રિટર શરૂ કરીએ અને અમે કહીએ છીએ કે હું 5 બરાબર છે, તો જો આપણે આ આદેશનો ઉપયોગ કરીએ તો તે આપણને i નો પ્રકાર કહે છે. તેથી, તે ફોર્મમાં પરત કરે છે જે બરાબર પારદર્શક નથી, પરંતુ તે કહે છે કે હું વર્ગ પૂર્ણાંક છે. તેથી, તમે શબ્દ પૂર્ણાંક જોશો, જો હું કહું કે જે j 7 બરાબર 5 છે અને હું જે પ્રકારનાં પ્રકાર માટે પૂછું છું તો તે કહેશે કે j એ ક્લાસ ફ્લોટનું છે. તેથી, આ સમીકરણોના પ્રકારોને દર્શાવવા માટે આંતરિક ઇન્ટ અને ફ્લોટસ આંતરિક રીતે ઉપયોગમાં લેવાય છે. હવે જો હું કહું કે હું 2 ગુણ્યા જ બરાબર છે, કારણ કે મેં સૂચવ્યું છે કે મારી પાસે મૂલ્ય 15 બિંદુ 0 છે, કારણ કે j એ ફ્લોટ છે અને તેથી, ગુણાકારને એક ફ્લોટમાં પરિણમ્યું અને ખરેખર જો આપણે આ બિંદુ પર હું આ પ્રકારનું પૂછીશ તો કહે છે કે હવે હું ફ્લોટ છું. ધ્યાનમાં રાખવાનું બિંદુ એ છે કે નામ પોતે છે નિશ્ચિત પ્રકારો નથી કે તેઓ અગાઉથી પ્રકારો સોંપેલ નથી. તે મૂલ્ય પર આધાર રાખે છે કે જે હાલમાં સોંપેલ છે તે છેલ્લી અભિવ્યક્તિ મુજબ તે નામમાં સંગ્રહિત છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 15:05)

અમારા બધા કાર્યોમાં અસ્પષ્ટ રૂપે મૂલ્યોનો બીજો એક મહત્વપૂર્ણ વર્ગ ઉપયોગ કરે છે બુલિયન મૂલ્યો જે સત્ય અથવા ખોટી રચના દર્શાવે છે. તેથી, આ પ્રકારના બે સ્થિરાંકો અથવા બે મૂળભૂત મૂલ્યો છે જેમાં પિથનને મૂડી "T" સાથે સાચું કહેવામાં આવે છે અને મૂડી "F" સાથે ખોટા છે. તેથી, સાચું એ મૂલ્ય છે જે કંઈક સાચી કહે છે. તેથી, જ્યારે આપણે યાદ રાખીએ છીએ કે આપણે કંઈક લખ્યું છે કે જો કંઈક થાય તો x કંઈક બરાબર છે, $x \bmod 7$ કંઈક બરાબર છે, આપણા જીસીડી કાર્યમાં કંઈક. તેથી, આવા અભિવ્યક્તિનું આઉટપુટ જ્યાં આપણે બીજી કોઈ અભિવ્યક્તિ સાથે તુલના કરીએ છીએ ડાબી પર એક અભિવ્યક્તિની જમણી બાજુએ એક અભિવ્યક્તિની સરખામણી કરો કે શું આ સરખામણીઓ સફળ થાય છે કે નિષ્ફળ જાય છે જ્યારે તે સફળ થાય છે કે નહીં તે સાચું છે અને જ્યારે તે નિષ્ફળ જાય છે તે ખોટું છે. આ અમારા પ્રોગ્રામના એક્ઝેક્યુશનને નિયંત્રિત કરવા માટે ઉપયોગમાં લેવાય છે. તેથી, આપણે આ મૂલ્યોને રેકૉર્ડ કરવાનો અને તેને ચાલાકી કરવાનો એક માર્ગ હોવો જરૂરી છે. મૂળભૂત મૂલ્યો સાચું કે ખોટા છે અને સામાન્ય રીતે ત્યાં ત્રણ કાર્યો છે જે આ મૂલ્યો પર કામ કરે છે. તેથી, કિંમત ન negates તેથી, સાચું ખોટું વિરુદ્ધ છે. તેથી, સાચા પર લાગુ નહીં અમને ખોટા પર લાગુ નથી ખોટા આપણે અમને સાચું આપણે અને સામાન્ય રીતે ઇંગલિશ અર્થ નીચે અને તેથી, જ્યારે અમે કહીએ છીએ કે કંઈક સાચી છે અને કંઈક બીજું સાચું છે, તો અમે તેનો અર્થ એ કે બરાબર તે બંને સાચા છે. તેથી, બુલિયન પ્રકારનું x અને y બે મૂલ્યો x હશે અને વાચ ક્ષણ પર પૂરા પાડવામાં આવશે. X પાસે એક મૂલ્ય સાચું છે અને y પણ મૂલ્ય સાચી છે. જો તેમાંના કોઈ સાચું ન હોય તો આઉટપુટ x અને y ખોટી છે. "અથવા" ફરીથી ઇંગ્લીશનો અર્થ છે, પરંતુ કોમ્પ્યુટર સાયન્સ અને તર્કશાસ્ત્રનો અર્થ અમે જે કહીએ છીએ તેનાથી થોડો અલગ છે. તેથી, સામાન્ય રીતે જ્યારે અમે કહીએ છીએ અથવા અમારો 1 અથવા અન્ય અર્થ છે તેથી, તમે કહી શકો કે ક્યાં તો હું જાગ્યો છું અથવા મારી બસ ચૂકીશ. તેથી, તમે શું કરશો એનો અર્થ એ છે કે આ બન્નેમાંથી એક બનશે, તે અસંભવિત છે કે તમે એમ માનો છો કે તમે સમયસર જાગશો અને તમારી બસ ચૂકી જશે. તે જ્યારે આપણે ઉપયોગમાં લેવું હોય અથવા અંગ્રેજીમાં આપણે સામાન્ય રીતે તેનો અર્થ થાય કે પ્રથમ વસ્તુ બનશે અથવા બીજી વસ્તુ થશે, પરંતુ બન્ને નહીં પરંતુ કોમ્પ્યુટર સાયન્સ અને તર્કશાસ્ત્રમાં છે અથવા તે એક છે, જેને સમાવિષ્ટ અથવા વિશિષ્ટ નથી, તે ફક્ત એક જ નહીં થાય અથવા અન્ય, પરંતુ બંને સમાવેશ થાય છે થઈ શકે છે તેથી, ઓછામાં ઓછા એક સાચું છે તો x અથવા y સાચું છે. તેથી, તેમાંનું એક સાચું હોવું જોઈએ, પરંતુ તે પણ શક્ય છે જ્યારે બન્ને તેઓ સાચા છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 17:28)

બુલિયન મૂલ્યો જે સૌથી વારંવારના રસ્તો છે તે સરખામણીઓ દ્વારા છે જે અમે પહેલાથી જ આમાંથી બે જોયા છે. તો, આપણે સમાન - સમાન જોઈ છે. આ ગણિતની વાસ્તવિક સમાનતા નથી જે એક સમાન છે, જે અસાધનમેન્ટ છે. તેથી, જો x બરાબર વાય ચેક દ્વારા બરાબર થાય, તો x ની વેલ્યુ વાસ્તવમાં મૂલ્ય y જેટલી હોય છે અને જો એમ હોય તો, તે અન્યથા સાચું મૂલ્ય આપે છે, તે ખોટા વળતર આપે છે. અને લાગતાવળગતા અસમાનતા ઓપરેટર ઉદ્ધારવાયક ચિહ્ન છે. તેથી, આ ઉદ્ધારવાયક સમાન નથી તે સમાન નથી માટે પ્રતીક છે અને આ સામાન્ય ગાણિતિક છે અને પછી અલબત્ત તમારી પાસે કિંમતો માટે હોય છે જે તમારી સરખામણીમાં નાની અથવા મોટી સરખામણીમાં હોઈ શકે છે, આ કરતા વધારે છે તે બરાબર છે અને આ બરાબર કરતા વધારે છે. તેથી, આપણી પાસે આ 6 લોજિકલ લોજિકલ કોરીજ ઓપરેટર્સના એરિથમેટિક કોમ્પેરેજન્સ ઓપરેટર્સ છે જે લોજિકલ વેલ્યુ સાચી અથવા ખોટા બનાવે છે.

(સ્લાઇડ સમય જણાવો: 18:24)

અને જે સામાન્ય વસ્તુ અમે કરીશું તે આને જોડે છે. તેથી, આપણે કહી શકીએ કે તે તપાસ કરો કે જ્યારે એન દ્વારા વિભાજિત થયેલ રીમાઇન્ડર 0 છે, તો તે 0 નો 0 હોય છે. તો, જો આપણે કહીએ કે 0 થી વધારે છે અને આ સંખ્યા ને 0 કરતા મોટો અને રિમાઇન્ડર n એ n ની બરાબર 0 બરાબર થાય. તેથી, તે n નો બહુવિધ છે અને n એ 0 નથી. અને આપણે આ પ્રકારની ફાઇલ પ્રકારની સરખામણી કરી શકીએ છીએ, જે આપણે બુલિયન મૂલ્ય કહીએ છીએ અને લે છે આ બુલિયન મૂલ્ય અને તેને એક નામ પર સોંપો. તેથી, આપણે કહી શકીએ છીએ કે n એ મીટરનો ભાજક છે જો n દ્વારા વિભાજિત મીટરનો રીમાઇન્ડર એ 0 ની સમાન હોય. અને આપણે કહી શકીએ કે તે વિભાજક છે તે સાચું છે. તેથી, ભાજક હવે ટાઇપ બૂલની બરાબર છે અને તેના મૂલ્યને સાચું કે ખોટા આધાર છે અથવા ન હોય કે ન હોય તે ન સમાન રીતે મીટર વહેંચે છે કે નહીં.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 19:22)

ચાલો આપણે બુલિયન મૂલ્યોનો ઉપયોગ કેવી રીતે કરીએ તેનું ઉદાહરણ જોઈએ. તો, ચાલો આપણે વિભાજન ઉદાહરણ પર પાછા જઈએ. ગણિતમાં આપણે લખીએ છીએ કે m એ n ને વિભાજિત કરે છે કે m એ n નો ભાજક છે. તેથી, આનો મતલબ એમ છે કે m વખત k એ કેટલાક k માટે n બરાબર છે. તેથી, મીટર n ને વિભાજિત કરે છે જો m ના વિભાજનની રીમાઇન્ડર મીટર 0 છે. જો આમ હોય તો તમે સાચું પાછા ફરો તો તમે ખોટા અધિકાર પાછા ફરો છો. આ એક બહુ સરળ કાર્ય છે, તે બે દલીલો લે છે અને તપાસ કરે છે જો પ્રથમ દલીલ બીજા દલીલને વિભાજન કરે છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 20:00)

હવે આપણે શું કરી શકીએ તે અન્ય કાર્યને વ્યાખ્યાયિત કરે છે જેનું મૂલ્ય અહીંથી આવ્યું છે. તો, આપણે તપાસીએ કે બે આ સંખ્યાના ભાગાકાર છે. તેથી, અમે તપાસીએ કે 2 ને વિભાજિત કરે છે; જો 2 ની ભાગીદારી n છે, તો પછી n પણ છે, આપણે સાચા ઠરાવો; જો 2 n ને વિભાજિત કરતું નથી, n એ વિચિત્ર છે, તો આપણે ખોટી રીતે પાછા આવીએ છીએ. તેથી, એ જ રીતે આપણે પહેલાના કેસની નકારાત્મકતાને વ્યાખ્યાયિત કરી શકીએ છીએ. તેથી, જો 2 ને વિભાજિત થાય તો n એ વિચિત્ર નથી. તમે જવાબ આપો છો કે શું 2 વિભાજિત કરે છે અથવા નહીં, અને જવાબ વિચિત્ર વિચારવા માટે તેને રિવર્સ કરો. તેથી, જો 2 ને વિભાજિત કરે, તો તમે તેને નકારી કાઢો અને વિચિત્ર કહેવું ખોટું છે; જો 2 એ n ને ભાગાકાર કરતું નથી, તો તમે ખોટા પાછા મેળવી શકો છો અને તમે અવગણવું અને વિચિત્ર કહી શકો છો. તેથી, અમે માત્ર એટલું જ ભારવું માગતો હતો કે બુલિયન મૂલ્યોની ગણતરી કરી શકાય છે, સોંપાયેલ, સંખ્યાત્મક મૂલ્યોની જેમ જ પસાર થઈ શકે છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 20:50)

આપણે જે જોયું છે તે સારાંશ આપવા માટે એ છે કે મૂળભૂત મૂલ્યનું વિધાન મૂલ્ય મૂલ્યોનું નામ આપવાનું હોય છે અને તે નક્કી કરે છે કે કયા ઓપરેશનની મંજૂરી છે. તેથી, અમે આંકડાકીય પ્રકારો પર દાખલા તરીકે અંકગણિત ઓપરેશનનો ઉપયોગ કરી શકીએ છીએ, અમે બુલિયન પ્રકારો જેવા, અને, અથવા, જેવા લોજિકલ ઓપરેશનોનો ઉપયોગ કરી શકીએ છીએ, પરંતુ પાયથોન અને પરંપરાગત ભાષાઓ વચ્ચેનો અગત્યનો તફાવત કે જ્યાં આપણે અગાઉથી નામો જાહેર કરીએ છીએ કે તે પાયથોન ને ઠીક કરતું નથી નામો માટે પ્રકારો તેથી, આપણે એમ કહી શકીએ નહીં કે ‘આઈ’માં હંમેશાં ‘ટાઇ’ છે. ‘આઈ’માં એક પ્રકાર હશે જે તેને સોંપેલ છે તેના આધારે. એક નામ તેના વર્તમાનમાં સોંપેલ મૂલ્યમાંથી પ્રકારને પ્રાપ્ત કરે છે અને તેના પ્રકારને બદલી શકે છે કારણ કે પ્રોગ્રામ ઉત્પન્ન થાય છે તેના આધારે મૂલ્યોની સોંપણી કરવામાં આવી છે. આ ખાસ પ્રવચનમાં આપણે શું જોયું છે તે 3 મૂળભૂત પ્રકાર પૂર્ણાંક, ફ્લોટ અને બૂલ છે. જેમ જેમ આપણે આ અઠવાડિયે આગળ વધીએ છીએ, તેમ આપણે તેના પર નિર્ધારિત રસપ્રદ માળખા અને રસપ્રદ કામગીરી સાથે વધુ પ્રકારો જોશું.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર અને એલ્ગારિધમ્સ ઇન પાયથોન
પ્રો. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ
ઇન્ડિયન ઇન્સ્ટીટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 02 લેક્ચર - 02

સ્ટ્રિંગ્સ અમે હવે જોયું છે કે પાયથોન મૂલ્યને યાદ રાખવા માટે નામોનો ઉપયોગ કરે છે. મૂલ્યો વાસ્તવિક જથ્થા છે જે અમે અમારા પ્રોગ્રામમાં ચાલાકીથી કરીએ છીએ, આ નામોમાં સંગ્રહિત થાય છે. મૂલ્યો પ્રકારો છે, અને આવશ્યકપણે મૂલ્યના પ્રકાર નક્કી કરે છે કે ઓપરેશન કેવી રીતે માન્ય છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 00:02)

અમે જે પ્રકારો જોયા છે તે મૂળભૂત આંકડાકીય પ્રકારો છે - પૂર્ણાંક અને ફ્લોટ, અને લોજિકલ પ્રકાર બૂલ જે સાચું અથવા ખોટા મૂલ્યો લે છે. તેથી, સંખ્યાત્મક પ્રકારો માટે, આપણી પાસે અંકગણિત ઓપરેશન્સ છે, અમારી પાસે અન્ય ક્રિયાઓ છે જે વધુ જટિલ છે. બુલિયન પ્રકારો માટે આપણી પાસે અને, અથવા, નથી, જે આપણને સાચા અને ખોટા મૂલ્યોને ચાલાકી કરવા દે છે. અને પછી આપણી પાસે આ તુલનાત્મક ઓપરેટર્સ સમાન, મોટા અને તેથી વધુ છે, જે આપણને બે જુદા જુદા જથ્થાઓની સંબંધિત મૂલ્યો ચકાસવા માટે પરવાનગી આપે છે અને નક્કી કરે છે કે શું તેઓ એકબીજા સાથે કેટલાક ક્રમમાં છે. અમે કહ્યું હતું કે અગત્યની બાબત એ છે કે પાયથોન માં નામોની પાસે કોઈ નિશ્ચિત પ્રકાર નથી. તેથી, અમે એમ કહી શકીએ નહીં કે હું ટાઇપ ઇન્ટર અથવા એક્સ પ્રકારના ફ્લોટનું છે, તેના બદલે તે મૂલ્યો પર આધારિત છે અને ખાસ કરીને, જો કોઈ કિંમતનો ઉલ્લેખ કર્યા વિના નામ પ્રથમ વખત વાપરવામાં આવે તો પાયથોન ફરિયાદ કરશે. અમે પહેલાથી જ અન્ય પ્રોગ્રામિંગ ભાષાઓ જેવા નામો જાહેર કરવાની જરૂર નથી, પરંતુ જ્યારે પણ આપણે સૌપ્રથમ એક નવું નામનો ઉપયોગ કરીએ છીએ; તેનો પહેલો ઉપયોગ ડાબી બાજુએ અસાઇનમેન્ટ સ્ટેટમેન્ટમાં હોવો આવશ્યક છે. તેથી, આપણે જમણી બાજુએ એક અભિવ્યક્તિમાં નામનો ઉપયોગ કરીએ તે પહેલાં તે માન્ય મૂલ્ય હોવું જોઈએ.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 01:36)

ગણતરી દ્વારા આંકડાકીય પ્રકારોનો અર્થ એ છે કે ગણતરીમાં આ દિવસોમાં જ રસ છે. અમે જે ગણતરી કરીએ છીએ તે વાસ્તવમાં ટેક્સ્ટ સાથે કામ કરે છે. તેથી, જ્યારે આપણે એક દસ્તાવેજ તૈયાર કરીએ છીએ, દાખલા તરીકે, કોઈ વર્ડ પ્રોસેસર અથવા કોઈ અન્ય વસ્તુઓનો ઉપયોગ કરીને રજૂઆત માટે, તો પછી અમે ખરેખર લખાણને હેરફેર કરી રહ્યા છીએ; તેથી અમે ટેક્સ્ટની આસપાસ ખસેડીએ છીએ, બદલવા માટે કંઈક માટે શોધ કરી રહ્યા છીએ અને તેથી વધુ. પણ જ્યારે અમે માહિતી પોતે હેરફેર કરવામાં આવે છે, ઘણી વાર માહિતી ઘણી સોતોમાંથી આવે છે. અમારી પાસે એવા કોષ્ટકો છે કે જે કોઈક દ્વારા ટાઇપ કરવામાં આવે છે અથવા ડિવાઇસ દ્વારા જનરેટ થાય છે અને અમારે તેને સ્પ્રેડશીટમાં આયાત કરવું પડશે. અને પછી જો આપણે બીજા પ્રોગ્રામનો ઉપયોગ કરીને તેમને ચાલાકી કરવા ઇચ્છતા હોઈએ તો, આપણે તેમને સ્પ્રેડશીટમાંથી નિકાસ કરવા માંગીએ છીએ જે સામાન્ય રીતે તે ટેક્સ્ટ ફાઇલોનો ઉપયોગ કરીને કરવામાં આવે છે જેમાં સ્પ્રેડશીટના કોલમ્સને કોમા દ્વારા અલગ કરીને વ્યવસ્થિત રીતે સંગ્રહિત કરવામાં આવે છે. તેથી, આમાં ટેક્સ્ટ પ્રક્રિયા પણ શામેલ છે અને છેલ્લે, આપણામાંના મોટાભાગના કમ્પ્યુટર વાસ્તવમાં ઇન્ટરનેટ સાથે કામ કરતા સમયનો ઉપયોગ કરે છે. ઇન્ટરનેટનો ઉપયોગ કરતી વખતે અમે જે સૌથી સામાન્ય વસ્તુઓ કરીએ છીએ તે એક છે ક્વેરીઝ લખો અને ઇન્ટરનેટ પર મેળ ખાતાં દસ્તાવેજો અથવા અન્ય સોતો શોધો. તેથી, આ મોટાભાગની શોધ ક્વેરી પ્રક્રિયા હાલમાં ટેક્સ્ટનો ઉપયોગ કરીને કરવામાં આવે છે. તે ટેક્સ્ટ સાથે મેળ ખાય છે કે જે અમે

દસ્તાવેજો વિશે કેટલીક માહિતીને પણ ટેક્સ્ટમાં સર્વસામાન્ય રીતે આપીએ છીએ અને તે નક્કી કરે છે કે કયા દસ્તાવેજો અમારી ક્વેરી માટે સૌથી સુસંગત છે. તેથી, ટેક્સ્ટ પ્રક્રિયાનો સામાન્ય રીતે ગણતરીમાં મહત્વનો ભાગ છે. અને સરળતા જેમાં તમે પાચથોન માં લખાણને ચાલાકી કરી શકો છો તે એક કારણ છે જે ઇન્ટરનેટ એપ્લિકેશન્સ સહિત અનેક વસ્તુઓને પ્રોગ્રામ કરવા માટે ખૂબ જ લોકપ્રિય ભાષા બની છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 03:18)

પાચથોન ટેક્સ્ટ માટે ટાઇપ સ્ટ્રિંગનો ઉપયોગ કરે છે, જે આંતરિક રીતે str તરીકે ઓળખાય છે તેથી, આપણે str શબ્દને બદલે શબ્દ શબ્દનો ઉપયોગ કરીશું, કારણ કે તે કહેવું સરળ છે તેથી, શબ્દમાળા મૂળભૂત રીતે અક્ષરોની શ્રેણી છે અને પ્રોગ્રામિંગ ભાષાની જેમ નહિ પણ, પાચથોન ની લંબાઈ 1 થી એક અક્ષરને અલગ પાડવા માટે ચોક્કસ અક્ષર પ્રકાર નથી. તેથી, પાઈથનમાં ટેક્સ્ટ માટે માત્ર એક જ પ્રકાર છે, જે શબ્દમાળા છે, અને એક અક્ષર સ્ટ્રિંગથી અસ્પષ્ટતા ધરાવે છે લંબાઈ 1. તેથી, ત્યાં બે પ્રકારની વસ્તુઓ નથી; તે નથી કે આપણી પાસે એક અક્ષર છે અને પછી શબ્દમાળા અક્ષરોનો ક્રમ છે, શબ્દમાળા એ પ્રતીકોનું અનુક્રમ છે અને એક પ્રતીક માત્ર લંબાઈ 1 નું અનુક્રમ છે. આ પ્રકારના મૂલ્યો લખવામાં આવે છે કારણ કે આપણે સામાન્ય રીતે અવતરણની મદદથી અંગ્રેજીમાં કરીએ છીએ. જ્યારે આપણે સ્પષ્ટ કિંમત લખી લેવા માગીએ છીએ ત્યારે આપણે શરૂઆતની સીમા નિર્ધારિત કરવા માટે અને શબ્દમાળાને અંતે અવતરણ ચિહ્નોનો ઉપયોગ કરીએ છીએ. તેથી, અમે કોઈ પણ પ્રકારનું ક્વોટ વાપરી શકીએ છીએ, તેથી એક જ ક્વોટ આ કિસ્સામાં દર્શાવશે કે નામ શહેરને 'ચેન્નાઈ' શબ્દનો ઉપયોગ કરવામાં આવ્યો છે. નોંધ કરો કે જ્યારે આપણે આ કેપિટલ C જેવા પ્રતીકો લખીએ ત્યારે નાના કે તેથી અલગ હોય છે. તેથી, આપણે આ બે અવતરણમાં બરાબર પ્રતીકો જોયા છે કારણ કે નામ શહેરની સ્ટ્રિંગને સોંપેલ મૂલ્ય. હવે આપણે પણ ડબલ અવતરણ વાપરી શકીએ છીએ; અને બે અવતરણચિહ્નો વાપરવા માટે એક કારણ છે જો તમે સ્ટ્રિંગના ભાગરૂપે એક ક્વોટ વાપરવાની જરૂર છે. આ કરવા માટેનો એક માર્ગ છે; અને તે કરવા માટેની બીજી રીત ખરેખર બેક સ્લેશ લખવાનું છે. જો તમે સ્ટ્રિંગની મધ્યમાં બેક સ્લેશ અને ઓ ક્વોટ લખો છો, તો તેનો અર્થ એ છે કે આ ક્વોટ શબ્દમાળાના અંતમાં નથી અને પ્રતીક તરીકે લેવામાં આવે છે, પરંતુ અન્ય વસ્તુઓમાં અવતરણની જેમ ખાસ વસ્તુઓનો સમાવેશ કરવાની ઘણી સરળ રીત છે. અવતરણ બદલવા માટે છે તેથી, એક જ અવતરણમાં બે અવતરણચિહ્નો સામેલ હોઈ શકે છે, અને ડબલ અવતરણમાં કોઈ પણ મૂંઝવણ વગર સિંગલ ક્વોટ શામેલ હોઈ શકે છે. તેથી, આ કહે છે કે નામનું શીર્ષક "હાઈલીયર્સ ગાઈડ ટુ ગેલેક્સી" મૂલ્યને સોંપવામાં આવ્યું છે. હવે, જો તમે સ્ટ્રિંગમાં બમણો અવતરણ અને સિંગલ અવતરણ બંને ભેગા કરવા માંગતા હો તો? તેથી, પાચથોન ને ટ્રિપલ ક્વોટ તરીકે ઓળખાતી ખૂબ અનુકૂળ વસ્તુ છે. તેથી, તમે ત્રણ સિંગલ ક્વોટ્સ ખોલી શકો છો, અને પછી તમે બહુવિધ ડબલ અવતરણ અને સિંગલ અવતરણ સાથે તમને જે ગમે તે લખી શકો છો. તેથી, જો તમે '''કહેવા માગો છો, તો તેમણે કહ્યું હતું કે તેમના પ્રિય પુસ્તક "હાઈલીયર્સ ગાઈડ ટુ ધ ગેલેક્સી"''અવતરણમાં છે. " પછી આ વેલ્યુ સ્ટ્રિંગમાં બેવડું અવતરણ ધરાવે છે અને તેની અંદર એક ક્વોટ પણ છે. તેથી, આપણે તેને બે અવતરણચિહ્નોથી જોડી ના શકીએ અને આપણે તેને સિંગલ ક્વોટ્સમાં જોડી શકતા નથી, કારણ કે તેમાંના કોઈપણ અસ્પષ્ટ હશે સિવાય કે આપણે આ સ્લેશનો ઉપયોગ પહેલા મેં કહ્યું હતું. તેથી, જો આપણે પાછા સ્લેશનો ઉપયોગ કરવા નથી માંગતા, તો તમે ટ્રિપલ ક્વોટનો ઉપયોગ કરી શકો છો.

(સ્લાઇડ સમયનો સંદર્ભ લો: 06:06)

ચાલો જોઈએ કે કેવી રીતે તે પાચથોન દ્વિભાષીમાં કાર્ય કરે છે. તેથી, આપણે 'ચેન્નાઈ' ની સમકક્ષ કહી શકીએ છીએ અને હવે આની કિંમત કહેવામાં આવી છે અને અમે જોઈએ છીએ કે તે એક જ ક્વોટ સાથે અહેવાલ છે. જો આપણે ઓના પ્રકાર માટે પૂછતા હોય, તો તે કહે છે કે ઓ એ વર્ગનું વર્ગ છે.

તેથી, આ આપણને કહે છે કે આંતરિક પિથનને ખબર પડે છે કે s એ શબ્દમાળા છે. જો આપણે કહીએ છીએ ટી બરાબર છે તો ફક્ત અક્ષર x , પછી ટીનો પ્રકાર પણ એક શબ્દ છે. તેથી, એક અક્ષર અને બહુવિધ અક્ષરો વચ્ચે કોઈ તફાવત નથી. હવે જો આપણે કહીએ તો આપણે તેને ટૂંકા કરીએ છીએ કે શીર્ષક "હચહીકર્સ" બરાબર છે, જો તમે ટાઇટલના મૂલ્ય માટે પૂછતા હોવ તો, તે તમને ડબલ અવતરણની બહાર અને સિંગલ ક્વોટની બહાર બતાવે છે. તેથી, આ સૂચવે છે કે આ એકમાત્ર શબ્દમાળા છે અને ફરીથી શીર્ષકનો પ્રકાર str છે. અને છેલ્લે, જો હું કહું કે માયવોટ બરાબર છે અને હું ત્રણ અવતરણનો ઉપયોગ કરું છું અને હું '' '' હીચહાઇકર'સ '' 'નો ઉપયોગ કરું છું. તેથી, મારી પાસે "હચિકા'નો ડબલ અવતરણચિતક છે અને હચહીકરે પોતે એક જ ક્વોટ ધરાવે છે. અને હું તેના આસપાસ ત્રિપિ ક્વોટ્સનો ઉપયોગ કરું છું પછી મારા ક્વોટ યોગ્ય રીતે બતાવવામાં આવે છે. હવે નોંધ લો કે જ્યારે તે મારા ક્વોટ દર્શાવે છે, તે ત્રણ અવતરણ દર્શાવતું નથી. તેમાં કોઈ અન્ય એક ક્વોટની બહાર મૂકે છે અને તે બતાવે છે કે આ આંતરિક સિંગલ ક્વોટ બેક સ્લેશ સાથે પ્રકાશિત કરવામાં આવ્યો છે. તેથી, બેક સ્લેશ સિંગલ ક્વોટ એ પાયથોન ની રીત છે અને ઘણા પ્રોગ્રામિંગ લેંગ્વેજ્સે એમ કહીને રસ્તો આપ્યો છે કે આગળના અક્ષરને તેના માટે શું અર્થ થાય છે તેવું માનવામાં આવતું નથી, પણ તે છે. તેથી, એક જ ક્વોટ તરીકે માત્ર એક જ ક્વોટ લો, અવતરણના અંત તરીકે સારવાર ન કરો. બીજી વસ્તુ જે તમે સિંગલ ક્વોટ સાથે કરી શકો છો તે ખરેખર બહુવિધ રેખાઓ લખી છે. તેથી, હું આ પ્રથમ વાક્ય કરું છું, અને પછી બીજી લાઇન, અને પછી ત્રીજા વાક્ય, અને પછી ક્વોટ બંધ કરો પછી મારી ક્વોટ બેક સ્લેશ n સાથે પ્રથમ વાક્ય તરીકે બતાવવામાં આવે છે. તેથી, બેક સ્લેશ ફરી એક વિશિષ્ટ અક્ષર છે જે નવી લીટી સૂચવે છે; પછી બીજી લાઇન, પછી નવી લીટી, અને પછી ત્રીજા વાક્ય. અમે કહ્યું હતું કે તે પાયથોન ને ટેક્સ્ટને હેરફેર કરવા માટે ખૂબ જ ઉપયોગી છે અને એક અન્ય વસ્તુ જે તમે કરવા માંગો છો તે વાસ્તવમાં વાંચેલ છે અને ટેક્સ્ટની ફકરો અથવા દસ્તાવેજની બહુવિધ રેખાઓ કહે છે અને તે હકીકત વિશે ચિંતા કરવાની જરૂર નથી કે આ ઘણી રેખાઓ છે સ્ટ્રિંગ તરીકે ટેક્સ્ટ મૂલ્ય તરીકે તેને સ્ટોર કરો. પાયથોન માં આ ખૂબ શક્ય છે તમે બહુવિધ રેખાઓ એક જ મૂલ્યમાં એમ્બેડ કરી શકો છો.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 09:00)

જેમ આપણે કહ્યું છે કે શબ્દમાળા એ ક્રમ અથવા અક્ષરોની સૂચિ છે. તો, આ સૂચિમાં અમે વ્યક્તિગત અક્ષરો કેવી રીતે મેળવી શકીએ? ઠીક છે, આ અક્ષરો પોઝિશન્સ છે અને પાયથોન સ્થિતિ માં 0 થી સ્ટ્રિંગ શરૂઆતમાં છે. તેથી, જો મારી પાસે શબ્દમાળામાં n અક્ષરો હોય, તો સ્થિતિ 0 થી n ની 1 ઘોષિત થાય છે. તેથી, માનીએ છીએ કે આપણી પાસે સ્ટ્રીંગ છે હેલો, તે 5 છે અક્ષરો તેથી, શબ્દમાળામાં હોદ્દાઓને 0, 1, 2, 3 અને 4 કહેશે; તેથી આ રીતે આપણે સ્થાનો લેબલ કરીએ છીએ. અને પાયથોન ની અન્ય સગવડ તે છે કે આપણે વાસ્તવમાં તે પાછળની તરફ લેબલ કરી શકીએ છીએ. આપણે કહી શકીએ કે આ પોઝિશન માઇનસ 1 છે; ઘણી વખત તમે કહી શકો છો કે સ્ટ્રિંગનું છેલ્લું પાત્ર લો અને કંઈક કરો. તેથી, લંબાઈને યાદ રાખવાની જગ્યાએ અને પછી અંત પર જાઓ, તે કહેવું અનુકૂળ છે કે છેલ્લા અક્ષર લો. તેથી, બાદ 1 મા અક્ષર લો. તેથી, આપણે ખરેખર આ જોયું અને અમે જીસીડી કર્યું, અમે વાતની છેલ્લી તત્વ વિશે વાત કરી કે સામાન્ય પાત્રોની સૂચિ છે, અને અમે જણાવ્યું હતું કે બાદબાકી -1 ની તત્વ n એ છેલ્લી તત્વ છે. આ ક્રમાંકન યોજના કે જે અનૌપચારિક રીતે સૂચિ માટે ઉપયોગમાં લેવાય છે તે જી.સીડીડી ઉદાહરણમાં ઔપચારિક રીતે સમજાવીને વિના, તે વાસ્તવમાં સમાન નંબરવાળી યોજના છે જે શબ્દમાળામાં સ્થાનો માટે વપરાય છે. અમારી પાસે ઓછા 1, ઓછા 2, ઓછા 3, ઓછા 4, ઓછા 5 છે, તેથી મહત્વની બાબત એ યાદ રાખવું એ આગળ વધવું એ છે કે, તમે 0 થી શરૂ કરો છો અને પછાત થતા તમે ઓછા 1 થી શરૂ કરો, કારણ કે દેખીતી રીતે, ઓછા 0 બરાબર છે 0. તેથી, જો આપણે સૌથી વધુ વસ્તુ માટે ઓછા બાદ 0 નો ઉપયોગ કરીએ છીએ તો આપણે પ્રથમ મૂલ્ય અથવા અંતિમ મૂલ્ય વિશે વાત કરી રહ્યા છીએ કે કેમ તે ભયંકર મૂંઝવણ હશે. તેથી, ફોરવર્ડ પોઝિશન શરૂઆતથી 0 થી શરૂ થાય છે અને છેલ્લી એલિમેન્ટમાંથી રિવર્સ પોઝિશન શૂન્યથી શરૂ થાય છે.

(સ્લાઈડ ટાઇમ નો ઉલ્લેખ કરો: 10:37)

એકવાર આપણી પાસે આ છે તો આપણે જોઈ શકીએ છીએ કે આપણે આ ચોરસ-બ્રેકેટ નોટિયોનો ઉપયોગ કરીએ છીએ વ્યક્તિગત સ્થાનો કાઢવા n . તેથી, $s1$, તેથી તે સ્થાન 1 પરનો અક્ષર એ ઈ છે અને જો હું પાછળની તરફ ચાલું છું તો s minus 2 એ 1 છે.

(સ્લાઈડ સમયનો સંદર્ભ લો: 10:59)

શબ્દમાળાઓ સાથે એક સૌથી વધુ મૂળભૂત વસ્તુઓ એક કરી શકે છે; એક મોટા શબ્દમાળામાં બે મૂલ્યોને એકીકૃત કરવા માટે અને આને કોન્ટેનરનાશન કહેવામાં આવે છે; તેમને એક પછી એક મૂકવા. અને આ માટે ઉપયોગમાં લેવાયેલા ઓપરેટર વત્તા છે. તેથી, વત્તા, અમે સંખ્યાત્મક મૂલ્યો માટે જોયું; શબ્દમાળાઓ માટે સમાન પ્રતીક વત્તા શબ્દમાળાઓ ઉમેરી શકતા નથી; દેખીતી રીતે, તે શબ્દમાળાઓ ઉમેરવા માટે અર્થમાં નથી, પરંતુ તે તેમને juxtaposes, અન્ય પછી એક મૂકે છે તેથી, જો અમારી પાસે હેલ્લો શબ્દ છે, જેમ આપણે પહેલાં કર્યું, અને અમે આ લઈએ છીએ, અને અમે એક નવી સ્ટ્રિંગ લઈએ છીએ અને અમે તેને s માં ઉમેરીએ છીએ. પછી આપણને સ્ટ્રિંગ ટી મળે છે, જેની કિંમત એ ભાગ છે જે હેલ્લો વત્તા ભાગમાં ઉમેરાયો હતો. તેથી, વત્તા માત્ર સરળ ઓપરેટર છે જે બે શબ્દમાળાઓ લે છે અને તેમને બાજુએ લાકડી રાખે છે.

(સ્લાઈડ સમયનો સંદર્ભ લો: 11:52)

ચાલો દુભાષિયોમાં એક ઉદાહરણ જોઈએ. માત્ર એક બિંદુ પર ભાર મૂકે છે; ધારણા મુજબ મેં કહ્યું હતું કે હેલ્લો છે અને ટી ત્યાં છે, પછી એસ વત્તા ટી એ ત્યાં હેલ્લો હશે. હવે નોંધ લો કે ત્યાં કોઈ જગ્યા નથી. તેથી, વત્તા શાબ્દિક મૂકે છે અનુસરતા ટી, તે વિરામચિહ્ન, કોઈપણ અલગ, કોઈપણ જગ્યા રજૂ કરતું નથી અને તમે ઇચ્છો તે પ્રમાણે આ છે. જો તમે અલ્પવિરામ અથવા જગ્યા મૂકવા માંગતા હોવ તો તમારે તે કરવું જ પડશે, તેથી જો તમે કહો કે તે જગ્યાએ જગ્યા હતી ત્યાં ટી ખાલી જગ્યા છે જે ત્યાંથી અનુસરવામાં આવે છે, હવે જો હું s વત્તા ટી કહું તો મને એક જગ્યા મળે છે. હેલ્લો અને ત્યાં વચ્ચે આ નોંધવું અગત્યનું છે કે વત્તા સીધા વસ્તુઓને એકસાથે મૂકે છે તે કોઈ પણ વિરામચિહ્ન અથવા બે મૂલ્યો વચ્ચે કોઈ અલગતાને ઉમેરતા નથી. તેથી, એવું છે કે તમારી પાસે એક નવી સ્ટ્રિંગ છે જે ઘણી જૂની શબ્દમાળાઓથી બનેલી છે જેની સીમાઓને સંપૂર્ણપણે અદૃશ્ય થઈ જાય છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 12:47)

આપણે ફંક્શન લેનનો ઉપયોગ કરીને સ્ટ્રીંગની લંબાઈ મેળવી શકીએ છીએ. તેથી, લેન s ની લંબાઈ આપે છે તેથી, આ અક્ષરોની સંખ્યા છે તેથી, યાદ રાખો કે જો અક્ષરોની સંખ્યા n છે તો સ્થિતિ 0 થી n ઓછા 1 છે. તેથી, અહીં સ્ટ્રીંગની લંબાઈ 5 હશે, અહીં સ્ટ્રીંગ ટીની લંબાઈ 5 વત્તા 7 - 12 હશે. ઘણા અન્ય રસપ્રદ કાર્યો છે કે જે શબ્દમાળાઓનો ચાલાકી કરવા માટે ઉપયોગ કરી શકે છે, તમે વસ્તુઓને શોધી અને બદલી શકો છો, તમે કંઈકની પ્રથમ ઘટના શોધી શકો છો અને આ રીતે, અને પછીથી તેમાંની કેટલીક જોશો, જ્યારે આપણે શબ્દમાળાઓ અને ટેક્સ્ટ પ્રોસેસિંગ અને વધુ વિગતોમાં ફાઇલોમાંથી ડેટા વાંચીને.

(સ્લાઈડ સમયનો ઉલ્લેખ કરો: 13:26)

એક ખૂબ જ સામાન્ય વાત છે કે જે આપણે શબ્દમાળાઓ સાથે કરવા માંગીએ છીએ તે શબ્દમાળાના ભાગને બહાર કાઢવું છે. આપણે શરૂઆતનો પ્રારંભ કરી શકીએ છીએ, પ્રથમ શબ્દ અને તે જેવી વસ્તુઓ. પાચથોન માં આવું કરવા માટેનો સૌથી સરળ માર્ગ એ છે કે જેને સ્લાઇસ કહેવામાં આવે છે. સ્લાઇસ

સેગમેન્ટ છે, એક સેગમેન્ટ એટલે કે હું લાંબી સ્ટ્રિંગ લઈશ જે હું પાત્રની સૂચિ તરીકે વિચારી શકું છું અને હું તે ભાગને કેટલાક પ્રારંભ બિંદુથી કેટલાક અંત સુધી પહોંચવા માંગું છું.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 13:55)

આ તે પાયથોન જે સ્લાઇસ કહે છે. તેથી, જો આપણે કહીએ કે પહેલાથી હેલો છે, તો પછી એક સ્લાઇસ માટે આપણે આ પ્રારંભિક બિંદુ અને કોલોન દ્વારા અલગ અંત બિંદુ આપીએ છીએ. તેથી, આપણે આ ચોરસ કૌંસ નોટેશનને બરાબર રીતે ઉપયોગ કરીએ છીએ, જો કે આપણે સ્ટ્રિંગનો ભાગ કાઢીએ છીએ, પરંતુ તે ભાગ જે આપણે કાઢીએ છીએ તે એક પોઝિશન નથી, પરંતુ 1 થી 4 ની સ્થિતિ છે. હવે પાયથોન માં, આપણે જોયું કે આપણે આ રેંજ ફંક્શન જે અમે છેલ્લી વખત લખ્યું હતું, તે જણાવ્યું હતું કે, જો હું 1 થી એમની સંખ્યા જોઈએ તો મને 1 થી મીટર વત્તા 1 લખવા આવશ્યક છે, કારણ કે પાયથોન માં રેંજ ફંક્શન એક પદની અંતિમ પદની ટૂંકા ગાળા અટકાવે છે. શ્રેણી તેથી, તે જ રીતે, સ્લાઇસ એક સ્લાઇસમાં છેલ્લા ઇન્ડેક્સના ટૂંકા ગાળામાં અટકે છે. તેથી, જો હું આવું કરું તો યાદ રાખો કે હેલોમાં 0, 1, 2, 3, 4 ની સ્થિતિ છે, તેથી 1 થી 4 ની સ્લાઇસ 1 થી શરૂ થાય છે 2 થી જાય છે, 3 થાય, પરંતુ 4 ન જાય, તો તે છે માત્ર ઇથી એલ - બીજા એલ.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 14:59)

સામાન્ય રીતે, જો હું સ કોલ કોલન લખું તો તે સી પર શરૂ થાય છે અને એસજે બાદ 1 થી અંત થાય છે. કેટલાક શોર્ટકટ્સ છે જે યાદ રાખવા અને ઉપયોગમાં સરળ છે; વારંવાર તમે શબ્દમાળામાં પ્રથમ એન અક્ષરો લેવા માંગતા હોવ, પછી તમે 0 ને ભૂલી જશો, અને ફક્ત 0 થી સર્વસામાન્ય રીતે શરૂ કરો, તેથી તેને છોડો, તેથી ફક્ત કોલન અને જ્હોન કહેવું શરૂ કરો. તો આ આપણને બધા સ્થાને 0 1 અપડે જ -1 આપશે. તેથી, જો હું પ્રથમ સ્થાન છોડું તો તે 0 થી શરૂ થશે. એ જ રીતે, જો હું છેલ્લી પોઝિશન છોડું છું તો તે સ્ટ્રિંગના અંત સુધી ચાલશે. તેથી, જો હું દરેક વસ્તુને પછીથી ઈચ્છું તો પછી હું કહી શકું છું કે કોલ કોલન અને આ સરવાળાની લંબાઈ 1 ની નીચે જશે, પરંતુ જો હું સ્પષ્ટપણે એક સ્લાઇસ તરીકે લખું તો, હું માત્ર ઓની લંબાઈ લખીશ. તેથી, આવશ્યકપણે આ મુખ્ય કારણ છે કે પાયથોન ને આ સંમેલન છે જ્યારે હું 1 થી મીટર વત્તા 1 ની શ્રેણી જેવી કંઈક લખું ત્યારે અહીં આ વધારાની વત્તા 1 મળે છે. તેથી, આ વત્તા 1 માટેનું મુખ્ય કારણ માઇનસ 1 લખવાનું ટાળવાનું છે. જો મારી પાસે છેલ્લું પાત્ર શામેલ હોવું જોઈએ અને જો હું 0 માં ક્રમાંક શરૂ કરું તો, દરેક વખતે હું સ્ટ્રિંગના અંતમાં જવા માંગું છું. s ની લંબાઈ 1 કહે છે. તે માત્ર લંબાઈના વાક્યને વધુ અનુકૂળ છે, અને સર્વસામાન્ય રીતે તે ધારે છે કે તે જાણે છે કે તે લંબાઈની લંબાઈ નહીં, પરંતુ s minus 1 ની લંબાઈ છે. તેથી, આ સંપૂર્ણ મૂંઝવણ જો તમે તે હકીકત વિશે પાયથોનને કહી શકો છો કે રેન્જના જમણા હાથની બાજુમાંના તમામ ટૂંકા અંતની બધી રેંજ છે, તે હકીકતથી ઊભી થાય છે કે તમે ઘણીવાર કંઈકથી તેની લંબાઈ સુધી અથવા સૂચિમાં ચલાવવા માંગો છો અથવા શબ્દમાળા અને જ્યારે તમે કહો કે તમે નથી માંગતા તે બાદબાકી કરવાનું કહેવાનું યાદ રાખો.

(સ્લાઈડ ટાઇમ નો ઉલ્લેખ કરો: 16:45)

ચાલો પાયથોન ના દુભાષિયામાં બીજા સાથે રમીએ. તેથી, જો હું કહું કે હેલો બરાબર છે તો આપણે જોયું કે જો હું 1 થી 4 કરું તો મને 'eા' મળશે. જો હું કોલન 3 કહું તો મને 'હેલ' મળે છે જે 0 1 2 છે. જો હું 2 કોલોન કહું તો મને 'લોલો' મળે છે 2 3 4. જો હું કહું 3 2 1, તો આ કહે છે: 3 સ્થાને શરૂ કરો અને 1 ઓછા 1, જે 0 છે, તે સ્થાને જાઓ. તેથી, પાયથોન તમને કોઈ ભૂલ આપતું નથી, તે આ તમામ અમાન્ય રેંજ લે છે, ઉદાહરણ તરીકે, અંત બિંદુનો પ્રારંભિક બિંદુ માન્ય શ્રેણી વ્યાખ્યાયિત કરતું નથી, અને તે કહે છે આ ખાલી શબ્દમાળા છે. પ્રશ્નની બીજી બાજુએ, જો હું કંઈક કહેવું 0 થી 7 પર જાઓ, જ્યાં સ્ટ્રિંગમાં 7 માં સ્થાન નથી, અહીં પાયથોન તેના બદલે કોઈ ભૂલ આપશે નહીં, તે ફક્ત છેલ્લી સ્થિતિમાં જશે જે ખરેખર શબ્દમાળામાં અસ્તિત્વ ધરાવે છે. તેથી, સામાન્ય રીતે આ રેન્જ

મૂલ્યો સંવેદનશીલ રીતે ગણવામાં આવે છે, જો તમે એવા મૂલ્યો આપો છો જે અર્થમાં નથી કરતા. જ્યાં સુધી શક્ય હોય ત્યાં પાચથોન કંઈક સ્લાઇસ વ્યાખ્યા સાથે સંવેદનશીલ કરવા પ્રયત્ન કરે છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 17:57)

ભલે અમારી પાસે વ્યક્તિગત સ્થાનો અથવા વ્યક્તિગત સ્લાઇસેસ અથવા સ્ટ્રિંગની અંદર સિક્વન્સની એક્સેસ હોય, પણ અમે કોઈ શબ્દનો કોઈ ભાગ લઈ શકતા નથી અને તેને બદલીને તેને બદલી શકીએ છીએ. તેથી, અમે સ્થાનમાં કોઈ સ્ટ્રિંગને અપડેટ કરી શકતા નથી. ધારો કે, અમે અમારી સ્ટ્રિંગ "હેલો" લેવા માંગીએ છીએ અને તેને શબ્દ "મદદ!" માં બદલીશું તો તે સરસ રહેશે જો આપણે ત્રીજા અને ચોથા સ્થાને લઈ શકીએ. તેથી, 0, 1, 2, 3, 4, 5, 0, 1, 2, 3, 4, તેથી યાદ રાખો, જો આપણે કહી શકીએ કે આને ap માં કરો અને આને ઉદ્ધારવાયક ચિહ્ન તરીકે બનાવો, જેથી હું હેલ્લોને બદલે સહાય મેળવી શકે છે

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 18:36)

આપણે ફેરફાર 3 ની જેમ કંઈક લખવા ઈચ્છીએ છીએ, મૂલ્ય 3 ને સોંપી દો. હવે, કમનસીબે પાચથોન આને મંજૂરી આપતું નથી. તેથી, તમે તેના ભાગને બદલીને સ્થાનમાં સ્ટ્રિંગ અપડેટ કરી શકતા નથી. વાસ્તવમાં, જો તમે આનો પ્રયાસ કરો છો, તો તમને ખરેખર એક ભૂલ સંદેશો મળશે, ચાલો જોઈએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 18:54)

અહીં આપણી પાસે ચારમાં વ્યાખ્યાયિત કરેલા શબ્દમાળા હેલો છે, અને જો હું હવે કહેતો છું કે $s[3]$ બરાબર p છે, તો તે કહે છે કે તે આઇટમની સોંપણીને સમર્થન આપતું નથી, જે તે આપણે શું કરવાનો પ્રયાસ કરીએ છીએ કહે છે કે તમે સ્ટ્રીંગના ભાગો બદલી શકતા નથી કારણ કે તે રહે છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 19:12)

આ કરવાને બદલે, શબ્દમાળા લેવાનો પ્રયાસ કરો અને તેના ભાગને બદલવાને બદલે તમારે શું કરવાની જરૂર છે તે વાસ્તવમાં સ્લાઇસેસ અને કોફ્ટેટેનેશનની કલ્પનાનો ઉપયોગ કરીને અસરકારક રીતે નવી સ્ટાઇલ તૈયાર કરે છે. અહીં આપણે શું કરવા ઈચ્છીએ છીએ તે છે કે આપણે તે સ્ટ્રીંગનો પહેલો ભાગ લઈ જવા માંગીએ છીએ. આ પ્રથમ ત્રણ અક્ષરો છે, અને પછી આપણે આને p ઉદ્ધારવાયક ચિહ્ન તરીકે બદલવું છે. તેથી, આપણે શું કહી શકીએ 0, 1, 2 લીધેલું અપડેટ એસ છે જે 0 થી 3 ની સ્લાઇસ છે અને તેને નવા શબ્દમાળા p ઉદ્ધારવાયક ચિહ્ન સાથે જોડવું. તેથી, આ તમે પાચથોન માં શબ્દમાળાને સંશોધિત કરો છો, પરંતુ મહત્વની વાત એ છે કે આ એક નવું ઓ છે જેનો અમે દાવો નથી કરી રહ્યા છીએ કે આ ઓ એ જૂના ઓ જેવા જ છે. ત્યાં આપણે જૂના શબ્દમાળામાંથી નવી સ્ટ્રિંગ બનાવીએ છીએ અને કદાચ તે પાછું એક જ નામમાં સંગ્રહિત કરીએ, તે અંશતઃ જેવું છે જ્યારે આપણે કહીએ કે j બરાબર j વત્તા 5 છે, અમે ખરેખર એમ કહી રહ્યાં છીએ કે આપણે જે માટે નવું મૂલ્ય બનાવ્યું છે અને સંગ્રહિત છે તે પાછા જ. અહીં ફરીથી આપણે નવી સ્ટ્રિંગ બનાવી રહ્યા છીએ અને તેને પાછું મુકીએ છીએ, પરંતુ અમે તેને બદલતા નથી. હવે આ ક્ષણે સુધારણા અને નવું મૂલ્ય બનાવવાની વચ્ચેનો તફાવત ખૂબ જ મહત્વપૂર્ણ લાગશે નહીં, પરંતુ જેમ જેમ આપણે આગળ વધવું તે મહત્વપૂર્ણ બનશે. તેથી, શબ્દમાળાઓ જે અવિશ્વસનીય મૂલ્યો તરીકે ઓળખાય છે, તમે ખરેખર નવો મૂલ્ય બનાવવા વગર તેમને બદલી શકતા નથી; જ્યારે, સૂચિમાં આપણે જોશું કે જે વધુ સામાન્ય પ્રકારનો ક્રમ છે તે બદલી શકાય છે, તો તમે સૂચિમાં એક ભાગ લઈ શકો છો અને પછી તેને બીજી જગ્યાએ બદલી શકો છો. તેથી, અમે આ વિશે વધુ પછીથી જોશું, આ એક ખૂબ મહત્વનું ખ્યાલ છે. હવે યાદ રાખો કે શબ્દમાળાઓ બદલી શકાતી નથી.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 20:42)

આપણે જે જોયું છે તે સારાંશ માટે છે કે લખાણ મૂલ્યો ગણતરી માટે મહત્વપૂર્ણ છે, અને પિથન પાસે પ્રકારો - શબ્દમાળા અથવા str છે, જે લખાણ મૂલ્યોને દર્શાવવા માટે અક્ષરોનો અનુક્રમ છે. અને એક અક્ષર માટે એક અલગ પ્રકાર માટે કોઈ તફાવત નથી; પિથનમાં કોઈ એક અક્ષર પ્રકાર નથી, એક અક્ષર માત્ર લંબાઈ 1 ની સ્ટ્રિંગ છે. અમે ઇન્ડેક્સની સ્થિતિ દ્વારા વ્યક્તિગત અક્ષરોને બહાર કાઢીએ, અમે પેટા શબ્દમાળાઓ કાઢવા માટે સ્લાઇસેસનો ઉપયોગ કરી શકીએ છીએ, અને અમે કોન્ટાનાટેશન ઓપરેટર વત્તા , પરંતુ શબ્દમાળાઓ અસમર્થનીય છે. અમે કોઈ સ્ટ્રિંગના નામની મુલ્ય ન લઈ શકીએ અને તેને સ્થાનાંતરિત કરી શકીએ છીએ. અમે સ્લાઇસેસ અને કોન્ટેટેશનનો ઉપયોગ કરીને તેને હેરફેર કરીને એક નવું મૂલ્ય બનાવી શકીએ છીએ, પરંતુ અમે તેને સીધી અપડેટ કરી શકતા નથી, કારણ કે શબ્દમાળા નિર્વિવાદ છે.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગારિધમ્સ ઇન પાયથોન
પ્રો. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટીટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 02 લેક્ચર - 03

સૂચિ અત્યાર સુધીમાં આપણે પાયથોનમાં મૂલ્યોના કેટલાક મૂળભૂત પ્રકારો જોયાં છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 00:02)

તમે આંકડાકીય પ્રકારો સાથે પ્રારંભ કર્યો છે, જે બે કેટેગરીમાં પૂર્ણાંક અને ફ્લોટમાં વહેંચાયેલ છે. તેથી, પૂર્ણાંક પૂર્ણાંક અથવા પૂર્ણાંક રજૂ કરે છે, અને ફ્લોટ મૂલ્યો રજૂ કરે છે જે દશાંશ ચિહ્ન ધરાવે છે. અને આ માટે, આપણી પાસે અંકગણિત ઓપરેશન્સ જેમ કે વત્તા, બાદ, સમય, વિભાજન અને અન્ય વિધેયો કે જે અમે ગણિત લાઇબ્રેરીનો ઉપયોગ કરીને આયાત કરી શકીએ છીએ, જે પાયથોન માં બનેલ છે. તો પછી અમે એક નવી પ્રકારનું મૂલ્ય દાખલ કરીએ છીએ, જે લોજિકલ મૂલ્યો માટે સાચું અને ખોટું છે, જે પ્રકારનું બૂલ છે તેટલું પરિચિત નથી. અમે આ મૂલ્યો પર કામ કરી શકીએ છીએ જેમ કે કાર્યો જેમ કે, કિંમતને વિરુદ્ધ 'અને' અને 'અથવા' પર નકારી કાઢે છે. અને જ્યારે આપણે આંકડાકીય મૂલ્યો વચ્ચે તુલના કરીએ છીએ ત્યારે આવી સરખામણીના પરિણામ સામાન્ય રીતે એક બૂલ મૂલ્ય છે અને આપણે 'ન' અને 'અને' અને 'જટિલ પરિસ્થિતિઓ બનાવવા માટે આ સરખામણીઓને ભેગા કરી શકીએ છીએ. પહેલાના વ્યાખ્યાનમાં, અમે શબ્દમાળાઓ જુઓ. તેથી, શબ્દમાળાઓ લખાણ પ્રતિનિધિત્વ કરવા માટે વપરાય છે. તે અક્ષરોનો ક્રમ છે અને તે એક અનુક્રમ છે કારણ કે આપણે ક્રમની સ્થિતિ વિશે વાત કરી શકીએ છીએ. સ્થિતિ 0 ની શરૂઆતની શરૂઆત કરે છે અને n એ એક પર જાઓ જ્યાં n એ સ્ટ્રિંગની લંબાઈ છે. જો આપણે સ્ટ્રીંગ મૂલ્ય માટે ચોરસ કૌંસ હું કહીએ છીએ તો આ નંબરિંગ કન્વેન્શનનો ઉપયોગ કરીને આપણે ith પોઝિશન મેળવી શકીએ છીએ. અને એક સ્લાઇડસ આપણને પોઝિશન i માંથી એક સબ શ્રેણીને આપે છે જેમાંથી જમા ઓછા એક લિન્કિડ એસ ચોરસ બ્રેકેટ i કોલોન જે. મૂળ ક્રિયાઓ જે આપણે શબ્દમાળાઓ સાથે કરી શકીએ છીએ તે વત્તા કામગીરીનો ઉપયોગ કરીને તેમને એકસાથે ગુંદર કરવાની છે. પ્લસનો અર્થ છે શબ્દમાળાઓ માટે જોડાણ કરવું અને અંકગણિત અર્થમાં નહીં, આપણે લેન ફંક્શનનો ઉપયોગ કરીને સ્ટ્રિંગની લંબાઈને કાઢી શકીએ છીએ અને અમે કહ્યું છે કે આપણે વધુ જટિલ સ્ટ્રિંગ ફંક્શનને પાછળથી જોશું.

(સ્લાઇડ સમયનો સંદર્ભ લો: 01:59)

આજે આપણે યાદીઓ પર આગળ વધીએ છીએ. સૂચિ પણ મૂલ્યોનો ક્રમ છે, પરંતુ સૂચિમાં સમાન પ્રકારની જરૂર નથી. તેથી, આપણી પાસે પરિબળો કહેવાય તેવી સૂચિ હોઈ શકે, જેમાં સંખ્યા 1, 2, 5, 10 હોય. આપણી પાસે નામો તરીકેની યાદી છે, જે આણંદ, ચાર્લ્સ અને મુક્ષિત છે. પરંતુ આપણી પાસે એક સૂચિ પણ હોઈ શકે છે, જેને અમે મિશ્ર કહેવાય છે, જેમાં સંખ્યા અથવા બુલિયન અને સ્ટ્રિંગનો સમાવેશ થાય છે, હવે તે સૂચિ ધરાવે છે જે અલગ અલગ સ્થાનો પર વિવિધ પ્રકારનાં મૂલ્યો ધરાવે છે, પરંતુ પાયથોન ચોક્કસપણે તેને પરવાનગી આપે છે. જ્યારે આપણે સામાન્ય રીતે સૂચિબદ્ધ હોઈશું કે જે તમામ પૂર્ણાંકો અથવા તમામ શબ્દમાળાઓ અથવા બુલિયન મૂલ્યો છે, તે આ કિસ્સામાં હોઈ શકે છે કે સૂચિનાં જુદા જુદા ભાગોમાં વિવિધ પ્રકારો હોય છે. સૂચિ તે જ રીતે એક ક્રમ છે જે શબ્દમાળા છે અને તેની પાસે 0 થી n ની -1 ની જગ્યા છે જ્યાં n એ સૂચિની લંબાઈ છે. તેથી, આપણે હવે આપેલ સ્થિતિમાં મૂલ્યો કાઢીએ છીએ અથવા આપણે સ્લાઇસિસ કાઢીએ છીએ. આ ઉદાહરણમાં જો આપણે સૂચિના પરિબળોને લઈએ છીએ અને ત્રીજા સ્થાને અમે યાદ રાખીએ કે સ્થિતિ 0, 1, 2, 3 ને લેબલ કરવામાં

આવે છે, તો 3 ના અવયવ 10 છે. એ જ રીતે, જો તમે સૂચિને મિશ્રિત કરો છો અને આપણે આમાંથી સ્લાઇસ લઈએ છીએ 0 થી 2 ઓછા 1 પછી આપણે ઉપ અને 3 અને 2 નો સમાવેશ સૂચિ મેળવીએ છીએ.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 03:23)

સ્ટ્રિંગની જેમ, યાદીની લંબાઈ ફંક્શન લેન દ્વારા આપવામાં આવે છે. તેથી, નામના લેન 3 છે કારણ કે ત્યાં 1, 2, 3 નામો છે. યાદ રાખો કે લંબાઈ માત્ર એક સામાન્ય લંબાઈ છે, જ્યારે સ્થાનો 0 થી n ની -1 ઘાત થાય છે.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 03:43)

સૂચિ અને શબ્દમાળાઓ અને અમે અત્યાર સુધી જે જોયું છે તેમાં એક તફાવત છે. અમે કહ્યું હતું કે એક અક્ષરનો કોઈ ખ્યાલ નથી. શબ્દમાળામાં જો આપણે એક સ્થાન પર મૂલ્ય લઈએ છીએ અથવા આપણે કોઈ શબ્દમાળાને લંબાઈ 1 ની પેટા શબ્દમાળા લે છે, તો આપણે તે જ વસ્તુ મેળવી શકીએ છીએ. તેથી, જો અમારી પાસે શબ્દમાળા h છે, જે પોઝિશન 1, 2, 3, 4, 5 માફ 1, 2, 3, 4 ની લંબાઈ 5 છે. અને જો આપણે 0 સ્થાને માંગીએ તો આ આપણને અક્ષર 'h' આપશે. પરંતુ પિથનમાં અક્ષર 'h'; શબ્દ સ્ટ્રિંગ h માંથી અસ્પષ્ટતા ધરાવે છે, જો આપણે સબ ક્રમ માટે 0 થી 1 માંગીએ, પરંતુ 1 સહિત નહીં તો ફરીથી આપણને સ્ટ્રિંગ h મળે છે. તેથી, એક કિસ્સામાં એવું લાગે છે કે એક કિસ્સામાં અમે એક લંબાઈ એક પેટા સ્ટ્રિંગ બનાવી છે, અમને એક અક્ષર મળ્યું છે, પરંતુ પાયથોન અલગ નથી. તેથી, 0 ની એચ બરાબર એક મૂલ્ય તરીકે બરાબર છે જે સ્લાઇસ એચ 0 કોલોન 1 છે.

(ધીમું સમયનો સંદર્ભ લો: 04:39)

હવે, આ સૂચિમાં સામાન્ય રીતે થશે નહીં. તેથી, જો અમારી પાસે સૂચિ હોય તો સૂચિમાં ફરીથી 0, 1, 2, 3 કહે છે. અને હવે આપણે 0 મા સ્થાને લઈએ છીએ કે આપણને મૂલ્ય મળે છે, આપણને મૂલ્ય 1 મળે છે, તો આપણને સૂચિ મળી નથી. બીજી બાજુ જો આપણે 0 થી લઈને 1 સહિત ન જોડીએ તો આપણે 1 ની કિંમતની પરિબલોની પેટા સૂચિ મેળવીએ છીએ. તેથી, 0 નું પરિબલ 1 છે, 0 કોલોન 1 ના પરિબલો પણ સ્લાઇસ છે 1, પરંતુ અહીં આપણી પાસે એક જ મૂલ્ય છે, અહીં અમારી પાસે એક સૂચિ છે અને તેથી, આ બે વસ્તુઓ એકબીજાના બરાબર નથી ફક્ત યાદ રાખો કે શબ્દમાળામાં આપણે કોઈ એક સ્થાન અને એક લંબાઈનો ટુકડો વચ્ચે તફાવત કરી શકતા નથી. તેઓ અમને બરાબર સમાન પ્રકારનું મૂલ્ય અને તે જ કિંમત પોતે આપે છે. જ્યારે સૂચિમાં લંબાઈનો ટુકડો એક સૂચિ છે, પદ પરથી મૂલ્ય તે પદ પર એક જ મૂલ્ય છે.

(સ્લાઇડ સમય જણાવો: 05:39)

હવે, પુનરાવર્તિત સૂચિમાં અન્ય સૂચિ હોઈ શકે છે, તેથી તેને નેસ્ટિંગ કહેવામાં આવે છે. ઉદાહરણ તરીકે, અમારી પાસે નેસ્ટેડ સૂચિ હોઈ શકે છે આ શરૂઆતમાં એક મૂલ્ય છે જે બીજી સૂચિ છે. આ પોઝિશન 1 છે. આ પોઝિશન, માફ પદ 0 છે. આ પોઝિશન 1 છે અને આ પોઝિશન 2 છે. પોઝિશન 1 એ એક જ સરળ મૂલ્ય છે પૂર્ણાંક 0 એક પૂર્ણાંક 4 પદ 0 એ સૂચિ છે, જે બદલામાં પોતે બે પોઝિશન 0. અને 1. અને મૂલ્ય પોઝિશન 1 એ પોતે બીજી સૂચિ છે. તેથી, તે પુનરાવર્તિત સૂચિનું ત્રીજા સ્તર છે જે એક જ મૂલ્ય તરીકે 37 છે. એ જ રીતે, સ્થાને 2 પરનું મૂલ્ય પોતે શબ્દમાળા છે અને તેથી, આ સીધક છે તે એક ક્રમ છે અને તે પોતાનું સ્થાન છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 06:33)

જો આપણે આ ઉદાહરણને જોવો તો આપણે જોઈ શકીએ છીએ કે જો આપણે પુનરાવર્તિતમાં 0 મી સ્થાને જોવા માગીએ છીએ, તો આપણે એમ કહીએ છીએ કે આ મૂલ્ય મળે છે અને આ મૂલ્ય 2 અને બીજી બાજુ, જો આપણે પ્રથમ સ્થાને નંબર 1 માંગીએ તો આપણને 4 મૂલ્ય મળે છે. અને હવે જો આપણે 2 ની સ્થિતિ જોઈએ, જે આ સૂચિ છે, તે પછી આપણે 0 મા સ્થાને જુઓ જે આ સ્ટ્રિંગ છે અને તેમાં આપણે ત્રીજા અક્ષર માટે જોઈએ છીએ જે 0, 1, 2, 3 આ એલ અધિકાર છે. 2 ની પુનરાવર્તિત અમને લેબલની સૂચિમાં છેલ્લા મૂલ્ય પર લઈ જશે જેમાં આપણે સ્થાન 0 પર નજર રાખીએ છીએ, જે નેસ્ટ કરેલી સૂચિમાં પ્રથમ મૂલ્ય છે. અને તે આપણે 3 પોઝિશનને જોઈએ છીએ જે તે પદમાં સમાયેલ ક્રમનો ત્રીજો અક્ષર છે અને આપણને અક્ષર 1 અથવા સ્ટ્રિંગ 1 ખરેખર મળે છે. તે જ રીતે આપણે સ્લાઇસેસ પણ લઈ શકીએ છીએ. તો, આપણે આ 0 મી સ્થાને લઈ શકીએ છીએ જે આ સૂચિ છે, પછી આપણે 1 થી શરૂ કરીને શરૂ થતી સ્લાઇસ માટે કહીએ છીએ, પરંતુ 2 સહિત નહીં, એટલે કે, આપણે આ મુલ્યથી શરૂ કરીએ છીએ. અને તેથી આપણે સૂચિ 37 ધરાવતી સૂચિ મેળવી શકીએ છીએ. નોંધ લો કે આંતરિક યાદી મૂલ્ય છે, જમણા આ એ મૂલ્ય છે જે પોઝિશન 1 અને પોઝિશન 2 ની વચ્ચે હોય છે અને બાહ્ય લિસ્ટ છે કારણ કે, જ્યારે આપણે સૂચિમાં એક સ્લાઇસ લઈએ છીએ ત્યારે આપણને એક સૂચિ મળે છે. આ સૂચિની 2 સૂચિ 37 ની સૂચિ છે જે અમને ફક્ત 37 ની સૂચિ આપે છે, જેણે મૂલ્ય 2 નાખ્યું છે, પરંતુ અમને પેટા સૂચિ મળે છે. સૂચિ માટે આપણે આ પહેલાં જે ઉલ્લેખ કર્યો છે તે એક સ્લાઇસ આપણને એક સૂચિ આપે છે.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 08:18)

ચાલો આપણે આ પ્રમાણે ચાલીએ કે આપણે જે કહ્યું તે પ્રમાણે વર્તે છે. અહીં આપણે તે ઉદાહરણ સાથે માત્ર પાચથોન ઇન્ટરપ્રિટર લોડ કર્યો છે. નેસ્ટેડ એ આ સૂચિ છે અને જો તમે હમણાં કહીએ તો નેસ્ટેડ 0 તમને 2, 37 મળે તો નેસ્ટ કરેલ 1 આપણને મૂલ્ય 4 મળે છે. હવે જો આપણે નેસ્ટ કરેલ કહીએ તો અમે આ સૂચિ મેળવીએ છીએ. અમે કહીએ છીએ નેસ્ટ 2, 0 તો પછી તે સૂચિને ડ્રોપ કરે છે અને અમને ફક્ત એક સ્ટ્રિંગ આપે છે અને જો આપણે કહીએ તો 2, 0, 3 નેસ્ટ કર્યું તો આપણે પહેલા શબ્દમાળા તરીકે જોયું. અને પછી, અમે કહ્યું હતું કે હવે અમે પુનરાવર્તિત ઘટકો માટે અપડેટ કરી શકીએ છીએ, કોઈ પણ અપડેટ માફ કરશો અમે નેસ્ટ કરેલ 0 ને જોઈ શકીએ છીએ અને આ સ્લાઇસ 1 કોલોન 2 લઈ શકીએ છીએ અને આ પ્રથમ લિસ્ટમાં જાય છે અને અમને સૂચિ ધરાવતી સૂચિ આપે છે જે 37 છે. તેથી, બાહ્ય સૂચિ છે કારણ કે તે એક સ્લાઇસ છે અને આંતરિક એક છે કારણ કે યાદીની પ્રથમ આઇટમની સ્થાને કિંમત નેસ્ટ કરેલી છે તેની યાદી 37 છે.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 09:25)

સૂચિ અને સ્ટ્રિંગ વચ્ચેનું એક મૂળભૂત તફાવત અથવા ખરેખર કોઈ મૂલ્યો જે આપણે પહેલાં જોયાં તે છે કે સૂચિને સ્થાનમાં અપડેટ કરી શકાય છે. તેથી, જો અમારી પાસે સૂચિવાળી એક યાદી છે જે પહેલાંની હતી અને આપણે કહીએ છીએ કે તે 1 ની પુનરાવર્તિત છે. યાદ રાખો જ્યારે આપણે કોઈ સ્ટ્રિંગમાં પોઝિશન બદલવાનો પ્રયાસ કરીએ છીએ ત્યારે અમને ભૂલ મળી છે. અમે હેલો દ્વારા બીજા પૃષ્ઠને બદલીને ફક્ત p કહીને કહી શકીએ છીએ કે આપણે પોઝિશન 3 ને પીએફ દ્વારા બદલવાની જરૂર છે, પરંતુ સૂચિ માટે આ મંજૂરી છે. જો આપણે 4 ને સ્થાને 7 થી લઈ જવા માંગતા હોઈએ, તો આપણે કહી શકીએ કે નેસ્ટેડ એક બરાબર સાત છે અને આ આપણને 2, 37, 7 ની સૂચિ આપશે અને પછી હેલો અને આપણે આની અંદર પણ કરી શકીએ છીએ. આપણે કહી શકીએ કે આપણે આ સૂચિમાં જવા માંગીએ છીએ જે નેસ્ટ થયેલ છે 0 પછી આપણે આ સૂચિમાં જવા માંગીએ છીએ જે નેસ્ટ થયેલ 0, 1 છે, તો આપણે આ મુલ્યમાં જઈએ અને આ મુલ્યને બદલવા માંગીએ છીએ. અમે પ્રારંભિક મૂલ્યની સ્થાને 1 પર નેસ્ટ કરેલી સૂચિની પદ 0 પર મૂલ્ય બદલવા માંગીએ છીએ. આપણે કહીએ છીએ કે 0, 1, 0 બરાબર 19 નેસ્ટ કર્યું છે અને આ ફેરફાર ત્રીસથી ઓગણીસમાં થાય છે, તેથી આ માન્ય છે. પાચથોન નોટેશનમાં આપણે શું કહીએ છીએ કે યાદીઓ અમર્યાદિત છે, તેથી પરિવર્તન બદલાશે. સૂચિને સ્થાનાંતરિત કરી શકાય છે જો આપણે કોઈ સૂચિ લઈ શકીએ અને તેની સ્ટ્રક્ચરને સ્ટ્રિંગથી વિપરીત બદલી શકીએ છીએ જો આપણે

કોઈ સ્ટ્રિંગને બદલવાનો પ્રયાસ કરીએ છીએ તો આપણે ખરેખર નવી સ્ટ્રિંગ બનાવવાની છે અને નામ આપીએ છીએ, પરંતુ આ જ નામની યાદીમાં આપણે અન્ય ભાગોને અસર કર્યા વગર તેના ભાગો અપડેટ કરો.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 11:03)

ફરી એકવાર ચાલો તપાસ કરીએ કે આપણે જે કર્યું છે તે ખરેખર કામ કરે છે. તેથી, જો હું કહીશ કે 1 ની પુનરાવર્તન તે બરાબર 7 છે. પછી આ યાદી ને 4 ની જગ્યાએ સ્થાનાંતરિત 7 તરીકે સ્થાનાંતરિત થાય છે જો હું 0 ની પુનરાવર્તિત કહું છું, જે પ્રથમ યાદી 1 છે, જે બીજા છે નેસ્ટ કરેલી સૂચિ 0 બરાબર 19 છે, જમણે. તેથી, આ કહે છે કે જાઓ અને એકથી 1 9 સુધી 37 માં આવી જાય છે અને ખરેખર આ સાચું થાય છે તેથી, આ યાદી અને શબ્દમાળાઓ વચ્ચે તફાવત છે. યાદીઓ અમર્યાદ્ય મૂલ્યો છે જે આપણે બાકીની યાદીમાં નામ પર અસર કર્યા વિના આપેલ સ્થિતિ પર મૂલ્યો બદલી શકીએ છીએ અને બદલી શકીએ છીએ.

(સ્લાઈડ ટાઇમ: 11:45)

અસ્થાયી અને અસંબદ્ધ મૂલ્યો વચ્ચે તફાવતને સમજવું અગત્યનું છે, કારણ કે તે અસાઇનમેન્ટમાં એક મહત્વપૂર્ણ ભૂમિકા ભજવે છે. અને પછી તમે જોશો કે, જ્યારે આપણે વિધેયોના મૂલ્યોને દલીલો તરીકે પસાર કરીએ છીએ ત્યારે શું થાય છે તે એક મુખ્ય ભૂમિકા ભજવે છે. ચાલો જોઈએ કે જ્યારે આપણે નામો આપીએ ત્યારે શું થાય છે. ધારો કે અમે અસાઇનમેન્ટની નીચેની શ્રેણીમાંથી પસાર થઈએ છીએ. શરૂઆતમાં આપણે નામ x ને મુલ્ય 5 આપીએ છીએ અને ત્યારબાદ આપણે વેલ્યુ વાચ નામ અને મુલ્ય વાચ નામ આપીએ છીએ અને ત્યારબાદ આપણે x થી સાત ને ફરીથી સોંપે છીએ. અમે x એ 5 થી શરૂ કર્યું ત્યારથી અમે કહ્યું કે y પણ 5 છે, કારણ કે y એ x ની મુલ્ય છે. અને હવે આપણે x થી 7 બદલાય છે. તેથી, આપણે જે પ્રશ્ન પૂછવા ઈચ્છીએ છીએ તેમાં y નું મૂલ્ય બદલાયું છે. ચાલો આ કરીએ અને જુઓ કે તે જમણી શું થાય છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 12:38)

ચાલો x બરાબર 5, y બરાબર x સાથે શરૂ કરીએ. તેથી, જો આપણે y ની મુલ્ય માગીએ છીએ તો તે 5 ની અપેક્ષા મુજબ છે. હવે આપણે x થી સાતને બદલો એ પ્રશ્ન છે કે તે y 5 અથવા y 7 છે અને ખરેખર y હજુ 5 છે અને આ સંપૂર્ણપણે કુદરતી છે જ્યાં સુધી આપણી સમજણ એ જાય કે આપણે શું કર્યું, જ્યારે આપણે y ની મુલ્ય x ની કિંમત પર સેટ કરીએ. તેથી, ચાલો આપણે તેને 5 બનાવવું; અમે એમ ન કહી શકીએ કે તે સમાન મૂલ્ય x કાયમ તેથી આગળ છે.

(સ્લાઈડ સમયનો ઉલ્લેખ કરો: 13:07)

જેમ કે y નું મૂલ્ય વાસ્તવમાં બદલાયું નહોતું અને તે પ્રશ્ન શા માટે બદલવો જોઈએ. બધા પછી તે કુદરતી લાગે છે કે જ્યારે આપણે બીજું નામ ની કિંમત માટે કિંમત સોંપી છે, તો પછી આપણે શું કરી રહ્યા છે તે નકલ કહે છે કે કિંમત અને તેની નવી નકલ કરો. તેથી, x 5 છે, કારણ કે એક્સ તરીકે સમાન મૂલ્ય વર્તમાનમાં છે તે તેનો અર્થ એ નથી કે y અને x પોઈન્ટને સમાન મૂલ્ય બનાવવા માટે તેનો અર્થ એ કે y નો અર્થ પણ થાય છે. તેથી, જો x ને 7 માં અપડેટ થાય તો તે y .

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 13:42)

આ પ્રશ્ન ખરેખર એટલો સરળ નથી જ્યારે આપણો અંતઃપ્રજ્ઞા કહે છે કે સોંપણી એ હંમેશા કિંમતની નકલ કરવી જોઈએ. કેટલાક કિસ્સાઓમાં તે થાય છે કે બંને નામો એક જ મૂલ્ય તરફ સંકેત કરે છે. તેથી, અવિરત મૂલ્યો માટે આપણે ધારણ કરી શકીએ છીએ કે આપણે જે અંતઃપ્રજ્ઞા છીએ તે કહે છે કે જ્યારે પણ આપણે કોઈ નામ મૂલ્ય આપીએ છીએ ત્યારે તે મૂલ્યની નવી નકલ મળે છે. આ બધા પ્રકારો પર લાગુ પડે છે જે આપણે આજેના પ્રવચનથી જોયાં છે, એટલે કે ઇન્ટ ફ્લોટ બૂલ અને શબ્દમાળા આ બધા નિર્દોષ છે. જો આપણે એ પ્રકારની સોંપણી કરીએ છીએ કે જ્યાં આપણે x ને કંઈક સોંપી હોય તો y ની સમાન કિંમત x છે અને પછી x અપડેટ કરો, y બદલાશે નહીં. એક મૂલ્યને અપડેટ કરવું એ કોપિને અસર કરતું નથી, કારણ કે અમે ખરેખર કોપિ કર્યું છે મૂલ્ય

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 14:30)

જો કે આપણે ધ્યાન દોર્યું છે કે સૂચિમાં પદ્ધતિઓ અલગ છે અને સૂચિ બદલાય છે. તે બહાર આવ્યું છે કે પરિવર્તનીય મૂલ્ય સોંપણી માટે તાજો નકલ બનાવતી નથી. ચાલો નીચેનું ઉદાહરણ જોઈએ, આપણે પહેલી વાર યાદી 1, 3, 5, 7 ને નામની યાદી 1 ને કહીએ, પછી આપણે કહીએ કે યાદી 2 એ યાદી 1 જેવું જ છે. જો અમારી પાસે આ કોપી નોટેશન હોય તો તમારી પાસે સૂચિની બે નકલો હશે તો ધારો કે આપણે હવે યાદી 2 ની મોડિઅટેબિલિટીને બદલીને પોઝિશન 2 પર મૂલ્ય બદલીશું.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 15:08)

પ્રશ્ન એ છે કે યાદી 2, યાદી 2 એ પહેલા 1, 3, 5, 7 અથવા તે પ્રમાણે યાદી 2 એ 1, 3, 4, 7 જેવાં સૂચિ 1 ની જેમ થાય છે. તો, ચાલો આપણે જોઈએ કે દુભાષિયામાં શું થાય છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 15:25)

ચાલો આ ઉદાહરણને પાચથોન માં રન કરીએ. તેથી, આપણે કહીએ કે યાદી 1 બરાબર 1, 3, 5, 7, યાદી 2 બરાબર યાદી 1 છે. યાદી 2 એ ખરેખર 1, 3, 5, અને 7 છે. હવે આપણે સ્થાન સૂચિ 1, 2 ની બરાબર 4 માં અપડેટ કરીએ છીએ. આપણે કહીએ છીએ કે યાદી 1 એ 1, 3, 4, 7, 5 એ 4 દ્વારા બદલવામાં આવ્યો છે. પ્રશ્ન અમે પૂછે છે તે આ અસરગ્રસ્ત યાદી 2 છે કે નહીં અને આપણા અંતર્જ્ઞાનની વિરુદ્ધ છે કે અમારી પાસે મૂલ્યોની કોપિ કરવામાં આવી છે, જેમાં સૂચિ 1 એ આડકતરી રીતે યાદી 2 ની કિંમતને પણ અસર કરી છે તેથી, આ શા માટે થાય છે

(સ્લાઇડ સમયનો સંદર્ભ લો: 16:05)

તો, યાદી 2 4 પણ છે અને આ તે છે, કારણ કે, જ્યારે આપણે એક નામથી બીજા નામથી આને અસાઇનમેન્ટ કરીએ છીએ અને અન્ય કોઈ નામમાં પરિવર્તનક્ષમ મૂલ્ય ધરાવે છે ત્યારે ફક્ત ફેરફારયોગ્ય પ્રકાર જે આપણે અત્યાર સુધી જોયું છે તે એક સૂચિ છે પછી એમ કહીને કે તે બે નકલો છે, તો આપણે ખરેખર કહીએ છીએ કે યાદી 1 એ નિર્દેશ કરે છે જો તમે સૂચિ 1, 3, 5, 7 ની કિંમતને પસંદ કરો. અને હવે અમારી પાસે એ જ સૂચિ માટે બીજું નામ છે જે યાદી 2 છે. જો આપણે આ મૂલ્ય પર જઈએ અને 4 ને બદલીએ, તો યાદી 2 માં પણ આ સ્થાન પર સમાન મૂલ્ય 4 છે. અસફળ અને અસંબદ્ધ પ્રકારો માટે અસાઇનમેન્ટ કેવી રીતે કાર્ય કરશે તે મૂળભૂત મૂળભૂત તફાવત છે. પરિવર્તનક્ષમ પ્રકારો માટે આપણે મૂલ્યની નવી નકલ અને અસંબદ્ધ પ્રકારો માટે અને મોડેબલ પ્રકારોની સોંપણી કરવા માટે અસાઇનમેન્ટની વિચારણા કરી શકીએ છીએ તે નવી કોપિ બનાવી શકતું નથી, તેના બદલે તે બંને નામો બરાબર સમાન મૂલ્ય તરફ નિર્દેશ કરે છે. ક્યાં તો નામ દ્વારા જો આપણે પરિવર્તનક્ષમ મૂલ્યને અપડેટ કરવાનું થયું તો અન્ય નામ પણ અસર કરે છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 17:10)

આ કંઈક છે જે અમે જોશું તે ચોક્કસ પરિસ્થિતિઓમાં ઉપયોગી છે, પરંતુ જો આપણે આને બનવું ન જોઈએ તો શું કરવું જો આપણે સૂચિની સાચી નકલ બનાવીએ. તેથી, યાદ રાખો કે સ્લાઇડ એક સૂચિ લે છે અને અમને એક પદથી બીજા સ્થાન પર પેટા સૂચિ આપે છે. સ્લાઇડ ઓપરેશનનું પરિણામ વાસ્તવમાં એક નવી સૂચિ છે, કારણ કે સામાન્ય રીતે, અમે એક સૂચિ લઈએ છીએ અને અમે કેટલીક મધ્યવર્તી સ્થિતિ માટે અન્ય મધ્યવર્તી સ્થિતિને લઈશું, એટલે દેખીતી રીતે, નવી સૂચિ જૂની સૂચિથી અલગ છે. અમે એ પણ જોયું કે જ્યારે આપણે શબ્દમાળાઓ પર જોયું ત્યારે અમે સ્લાઇડને સ્પષ્ટ કરતી વખતે પ્રથમ સ્થાન અથવા છેલ્લી સ્થિતિ છોડી શકીએ. જો આપણે આ પ્રથમ સ્થાન છોડી દઈએ તો અમે કહીશું કે પ્રથમ સ્થાન 0 છે, તેથી અમે શરૂઆતથી શરૂ કરીએ છીએ. એ જ રીતે, જો આપણે આ જેવી છેલ્લી સ્થિતિ છોડી દઈશું, તો આપણે સંક્ષિપ્તમાં એમ ધારીએ છીએ કે છેલ્લું સ્થાન એ સ્લાઇડની સૂચિની લંબાઈ છે અને તેથી તે છેલ્લી શક્ય કિંમત પર જાય છે. જો આપણે પ્રથમ સ્થાન છોડીએ, તો આપણને 0 મળે છે; જો આપણે છેલ્લી સ્થિતિ છોડી દો, તો આપણે લંબાઈ મેળવી શકીએ છીએ. જો આપણે બન્ને સ્થાને છોડી દો, તો આપણે ફક્ત કોલોનને કંઈપણ પહેલા કશું પહેલેથી મૂકીશું પછી તાર્કિક રીતે આ 0 થશે અને આ લંબાઈ બને છે અમે એક જ વસ્તુમાં બંને લક્ષણો ધરાવે છે અને અમે આને સંપૂર્ણ સ્લાઇડ કહીએ છીએ. હવે ચાલો આપણે આ નિરીક્ષણને ભેગા કરીએ જે આ નિરીક્ષણ સાથેના ટૂંકા કટ સંકેત છે કે દરેક સ્લાઇડ નવી ઉપ સૂચિ બનાવે છે. તેથી, આપણી પાસે જે છે તે ફક્ત એક કોલોન પછી જ છે જે એલ જેવું જ નથી, તે જૂની યાદીમાંથી બનાવેલ નવી સૂચિ છે, પરંતુ તે સમાન અનુક્રમમાં. માં દરેક મૂલ્ય તરીકે છે. આ હવે આપણને યાદી 2 એ યાદી 1 બરાબર છે તે કહેવાને બદલે એક યાદીની નકલ કરવા માટે સરળ ઉકેલ આપે છે, જે તે પછી બંને બનાવે છે. યાદ રાખો કે મારી પાસે આ ન હોય તો મને યાદી 1 મળશે અને યાદી 2 એ જ વાસ્તવિક યાદી તરફ દોરશે. ત્યાં મૂલ્યોની ફક્ત 1 સૂચિ હશે અને તે જ નિર્દેશ કરે છે. પણ જો મારી પાસે આ હોય તો ચિત્ર બદલાય છે તો પછી શું થાય છે તે છે કે સ્લાઇડ ઓપરેશન નવી યાદી બનાવે છે જે બરાબર સમાન લંબાઈ અને સમાન કિંમતો ધરાવે છે અને તે યાદી 2 બિંદુ બનાવે છે. તેથી, આ સૂચિ 1 અને યાદી 2 પછી યાદીમાંથી દરેકને અલગ કરવામાં આવે છે 2 સૂચિમાં કોઈ પણ સુધારો યાદી 1 ને અસર નહીં કરે 1 યાદીમાં કોઈ પણ સુધારો યાદી 2 ને અસર કરશે નહીં ચાલો આપણે જોઈએ કે કેવી રીતે આ અર્થઘટન કરનારને પોતાને રોકે તેવું કાર્ય કરે છે, વાસ્તવમાં પાઇથોન આ સૌપણીને સંભાળે છે.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 19:45)

પહેલાં આપણે સૂચિ 1 થી 1, 3, 5, 7 અને યાદી 2 સાથે શરૂ કરીએ, ચાલો આપણે કહીએ કે સ્લાઇડ છે. તેથી, હવે, જો અપડેટ 2 ની જગ્યાએ 2 ની યાદીમાં છે તો યાદી 1 એ 1, 3, 4, 7 જેવો દેખાય છે. પરંતુ યાદી 2 જે એક કોપી હતી તે જમણી અસર થતી નથી. જ્યારે આપણે એક સ્લાઇડ લઈએ છીએ ત્યારે આપણને નવી સૂચિ મળે છે. તેથી, જો આપણે સમગ્ર સૂચિને એક સંપૂર્ણ સ્લાઇડ તરીકે લઈએ તો અમે જૂની સૂચિની સંપૂર્ણ નકલ મેળવી શકીએ છીએ અને અમે તેને સુરક્ષિત રીતે નવા નામથી સોંપી શકીએ છીએ અને તે હકીકત વિશે ચિંતા કરશો નહીં કે બંને નામો મૂલ્ય શેર કરી રહ્યાં છે.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 20:19)

આ આપણને સમાનતા પર વિષમતા તરફ દોરી જાય છે ચાલો આ પાઇથોન ના નિવેદનોના સેટ પર જુઓ. અમે 1, 3, 5, 7 ની સૂચિ બનાવીએ છીએ અને નામની સૂચિ 1 આપો અને જ્યારે આપણે બીજી સૂચિ 1, 3, 5, 7 બનાવીએ અને તેને નામ યાદી 2 આપીએ. અને આખરે, આપણે યાદી 3 ને યાદી 2 તરીકે સમાન કિંમતો તરીકે દર્શાવીએ છીએ અને કહેવામાં આવે છે કે યાદી 3 વાસ્તવમાં સમાન વસ્તુ તરફ સંકેત કરે છે. તેથી, હવે અમારી પાસે ચિત્રાત્મક રીતે અમારી પાસે ક્યાંય બીજો ક્રમ 1, 3, 5, 7 ની

બે સૂચિ છે. અને શરૂઆતમાં આપણે કહીએ છીએ કે આ યાદીમાં 1 પોઇન્ટ્સ અને યાદી 2 ની આ છેલ્લી સોંપણીમાં કહે છે કે યાદી 3 પણ આને નિર્દેશ કરે છે.

(સ્લાઇડ સમય જણાવો: 21:10)

ત્રણેય યાદીઓ સમાન છે, પરંતુ જે રીતે તેઓ સમાન છે તેમાં તફાવત છે. તેથી, યાદી 1 અને યાદી 2 બે અલગ અલગ યાદીઓ છે, પરંતુ તેમની પાસે સમાન મૂલ્ય છે. તેથી, તેઓ સમાન મૂલ્ય ધરાવતા હોય છે, પરંતુ તે બે અલગ અલગ વસ્તુઓ છે અને તેથી, જો આપણે એક પર કામ કરીએ તો આ સમાનતાને જાળવી રાખવાની જરૂર નથી. બીજી બાજુ યાદી 2 અને યાદી 3 બરાબર બરાબર છે કારણ કે તેઓ સમાન મૂલ્યને નિર્દેશ કરે છે, ત્યાં બરાબર એક સૂચિ છે કે જેમાં તેઓ બન્ને પોઇન્ટ કરે છે. તેથી, જો આપણે સૂચિ 3 અપડેટ કરીએ છીએ અથવા અમે સૂચિ 2 અપડેટ કરીએ છીએ તો તે સમાન રહેવાનું રહેશે. સમાનતાના બે અલગ અલગ વિચારો છે કે શું મૂલ્ય એ જ છે અથવા વાસ્તવિક અંડરલાઇન ઓબ્જેક્ટ છે જેનો આપણે આ નામનો ઉલ્લેખ કરીએ છીએ તે જ છે. બીજા કિસ્સામાં, ઓબ્જેક્ટને ક્યાં તો નામ પર અપડેટ કરવું બંને નામો સમાન રહેશે.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 21:57)

પાયથોનએ આ ઓપરેશનને સમાન સમાન જોયું છે, જે સમકક્ષ અથવા ગાણિતિક સમાનતા છે, જે તપાસ કરે છે કે જો x અને y ના નામો સમાન કિંમત હોય. તેથી, આ એ હકીકતને પકડી લેશે કે યાદી 1 એ યાદી 2 ની બરાબર છે છતાં પણ તે બે અલગ અલગ યાદીઓ છે તેમ છતાં તે સમાન મૂલ્ય ધરાવે છે. બીજી પ્રકારની સમાનતા જોવા માટે કે યાદી 3 અને યાદી 2 ખરેખર મેમરીમાં સમાન ભૌતિક સૂચિ છે. આપણી પાસે ' છે' નામના અજગરમાં બીજું એક બીજું કી શબ્દ છે. તેથી, જ્યારે આપણે કહીએ છીએ x એ y છે કે આપણે શું કહીએ છીએ, તે છે કે શું એક્સ અને વાય વાસ્તવમાં સમાન મેમરી સ્થાનને તે જ મૂલ્ય તરફ નિર્દેશ કરે છે કે જેમાં x અપડેટ કરવું y અને તેનાથી વિરુદ્ધ થશે. આપણે કહી શકીએ કે x એ વાય ચેક છે જો x અને y એ જ ઓબ્જેક્ટ નો સંદર્ભ લો. આ વર્ણન દ્વારા બરાબર સમાન અને કાર્ય છે; દેખીતી રીતે, જો યાદી 2 યાદી 3 એ સમાન પદાર્થ હોય તો તે હંમેશા સમાન - સમાન હોવું જોઈએ. તેથી, x વાય છે અને x હંમેશા y ની સમાન હોય છે, કારણ કે વાસ્તવમાં તે જ વસ્તુ તરફ સંકેત છે. પરંતુ આ કિસ્સામાં જોકે યાદી 1 યાદી 2 કદાચ અલગ યાદી છે તેઓ હજુ પણ બરાબર - બરાબર છે, કારણ કે કિંમત સમાન છે. બીજી બાજુ જો હું ઓપરેશનને જોઉં છું તો યાદી 1 યાદી 2 એ યાદી 3 એ true છે, કારણ કે અમે જોયું છે કે આ સોંપણી સૂચિની નકલ કરશે નહીં, તે ફક્ત યાદી 3 બિંદુને એક જ વસ્તુ યાદી 2 બનાવશે. બીજી બાજુ યાદી 1 પર યાદી 2 false છે, કારણ કે તે બે અલગ અલગ યાદી છે. તેથી, ફરી એકવાર પોતાની જાતને આને ચકાસવા માટે આપણી જાતને ખાતરી કરવા માટે કે આ વર્ણન વાસ્તવમાં સચોટ છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 23:41)

ચાલો દુભાષિયામાં તે ત્રણ રેખાઓ અજગર લખીએ. તેથી, આપણે કહીએ છીએ કે યાદી 1 એ 1, 3, 5, 7 યાદી 2 એ પણ 1, 3, 5, 7 અને યાદી 3 એ યાદી 2 છે. હવે, આપણે પૂછીએ કે યાદી 1 એ યાદી 2 સમાન છે અને તે ખરેખર સાચું છે, પરંતુ જો આપણે પૂછીએ કે યાદી 1 એ યાદી 2 છે તો તે ખોટું કહે છે. તેથી, આનો અર્થ એ છે કે યાદી 1 અને યાદી 2 એ સમાન મૂલ્યને શારીરિક રીતે નિર્દેશિત કરે છે. તેથી, અમે એક અપડેટ કરીએ છીએ જે તે અન્યને અપડેટ કરશે નહીં. બીજી બાજુ, જો આપણે પૂછીએ કે યાદી 2 એ યાદી 3 છે તો આ સાચું છે. દાખલા તરીકે, જો આપણે યાદી 2, 2 ની બરાબર 4 કરીએ, જેમ કે આપણે પહેલાંના ઉદાહરણમાં પૂર્ણ કર્યું છે, તો યાદી 2 હવે 1, 3, 4, 7 બની ગયું છે. તેથી, જો આપણે પૂછીએ કે યાદી 1 એ આ બિંદુએ યાદી 2 બરાબર છે ખોટા છે તે મૂલ્યો. તેથી, કારણ કે યાદી 1 1, 3, 5, 7 અને યાદી 2 એ 1 3 4 7; તેમ છતાં, જો આપણે કહીએ કે યાદી 2 એ યાદી 3 બરાબર છે તો તે સાચું છે, કારણ કે

યાદી 3 એ યાદી 2 એ સૂચિમાં છે જો તે બંને એક જ ભૌતિક યાદી છે અને તેથી જ્યારે અમે યાદી 3 યાદી 2 અપડેટ કરીએ છીએ ત્યારે તે યાદી 3 દ્વારા અપડેટ કરવામાં આવશે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 24:54)

શબ્દમાળાઓની જેમ, આપણે વત્તા ઓપરેટરનો ઉપયોગ કરીને એકસાથે યાદીઓને એકીકૃત કરી શકીએ છીએ. તેથી, વત્તા જોડાણો છે. તેથી, જો આપણી પાસે સૂચિ 1 છે, તો મૂલ્ય 1, 3, 5, 7, 2 એ મૂલ્ય 4, 5, 6, 8. છે. પછી યાદી 3 ની સમાન યાદી 1 વત્તા લિસ્ટ 2 એ અમારા માટે મૂલ્ય 1, 3, 5, 7, 4 બનાવશે. 5, 6, 8. પરિવર્તનક્ષમ અને નિર્વિકાર મૂલ્યોના આપણા સંદર્ભમાં ઓળખી લેવાની એક મહત્વપૂર્ણ બાબત એ છે કે વત્તા હંમેશાં એક નવી સૂચિ ઉત્પન્ન કરે છે. જો આપણે કહીએ કે યાદી 1 1, 3, 5, 7 છે અને પછી આપણે આ યાદીને યાદી 2 નામની નકલ કરીએ છીએ. આપણે પહેલા જોયું કે અમારી પાસે 1, 3, 5, 7 છે અને આપણી પાસે બે નામો યાદી 1 અને યાદી 2 છે. હવે જો આપણે યાદી 1 વત્તા નવ કહીને લિસ્ટ 1 અપડેટ કરીએ તો તે ખરેખર એક નવી યાદી બનાવશે જે અંતમાં નવ હશે અને તે સૂચિ 1 નું નિર્માણ કરશે અને યાદી 2 હવે તે જ હક નહીં. તેથી, યાદી 1 અને યાદી 2 હવે સમાન ઓબ્જેક્ટ પર નિર્દેશ નહીં કરે. ચાલો આની પુષ્ટિ કરીએ.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 26:00)

અજગરના ઈન્ટરપ્રીટરમાં ચાલો આપણે સૂચી સેટ કરીએ 1 એ 1, 3, 5, 7 ની સમાન હોય અને કહીએ કે યાદી 2 બરાબર યાદી 1 છે. પછી આપણે જોયું તે પહેલાં જો આપણે કહીએ કે યાદી 1 એ યાદી 2 છે, તો આપણે સાચા છીએ. જો બીજો બાજુ આપણે યાદી 1 ની જૂની સૂચિ 1 ની જૂની મૂલ્ય અને નવું મૂલ્ય 9 નું પુનઃ સોંપણી કરીશું. આ યાદી 1 થી 1, 3, 5, 7, 9 સુધી વિસ્તરે છે. હવે આપણે જોશું કે યાદી 2 એ યથાવત છે. તેથી, યાદી 1 અને યાદી 2 ઘટ્ટ બની ગયા છે કારણ કે જે સમયે આપણે લાગુ કરીએ છીએ તે સ્લાઇડસને લેવા જેવું છે. જ્યારે પણ અમે લાગુ કરીએ ત્યારે અમે ખરેખર નવી સૂચિ મેળવીએ છીએ. તેથી, સૂચિ 1 એ લાંબા સમય સુધી સૂચિ પર ધ્યાન આપી રહી છે જે તે મૂળ રૂપે નિર્દેશ કરતી હતી. તે જૂની યાદીમાંથી બનાવેલી નવી સૂચિ તરફ ધ્યાન દોરે છે જે 9 અંતમાં તેને જોડે છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 26:49)

સારાંશ માટે આપણે હવે સૂચિ તરીકે ઓળખાતી નવી પ્રકારનું મૂલ્ય જોયું છે. તેથી, સૂચિ માત્ર મૂલ્યોનો ક્રમ છે આ મૂલ્યો એક સમાન પ્રકારની હોવાની જરૂર નથી, અમારી પાસે મિશ્ર યાદી છે અથવા સૂચિ છે, બુલિયન, પૂર્ણાંકો, શબ્દમાળાઓ. જોકે હંમેશાં અમે યાદી જોયેલી હોત, જ્યાં સૂચિની નીચે લીટીની સામગ્રી નિશ્ચિત પ્રકારનું હોય છે. તેથી, તમામ સ્થાનની વાસ્તવમાં સામાન્ય રીતે સમાન પ્રકાર હોય છે, પરંતુ આ અજગર દ્વારા આવશ્યક નથી અને અમે નેસ્ટ લિસ્ટ કરી શકીએ છીએ. તેથી, સૂચિની સૂચિ અને સૂચિની યાદી હોઈ શકે છે અને તેથી વધુ. શબ્દમાળાઓ સાથે, આપણે આ ચોરસ ક્રોસ નોટેશનનો ઉપયોગ પોઝિશન પર મૂલ્ય મેળવી શકીએ છીએ અથવા આપણે સબ સૂચિ અથવા સ્લાઇડ મેળવવા માટે કોલોન નોટેશન સાથે ચોરસ બ્રેકેટનો ઉપયોગ કરી શકીએ છીએ. અજગરની એક નવી સુવિધા, જે અમે સૂચિ સાથે રજૂ કરી છે, તે પરિવર્તનક્ષમ મૂલ્યનો ખ્યાલ છે. તેથી, સૂચિને અપડેટ કરી શકાય છે, અમે સૂચિનાં ભાગો લઈ શકીએ છીએ અને બાકીના ભાગોને અસર કર્યા વિના બદલી શકીએ છીએ, તે મેમરીમાં નવી સૂચિ બનાવી શકતી નથી. એક પરિણામ એ છે કે આપણે સોંપણી વધુ કાળજીપૂર્વક જોવી પડશે. અવિશ્વસનીય મૂલ્યો માટે જે પ્રકારો આપણે અત્યાર સુધી જોયા છે, પૂર્ણાંક, ફ્લોટ, બૂલ અને શબ્દમાળા જ્યારે આપણે x બરાબર y ની મુલ્ય x ની નકલ કરીએ છીએ. તેથી, x અપડેટ કરવું y અને ઊલટું અસર કરતું નથી. પરંતુ જ્યારે આપણી પાસે સૂચી જેવા ફેર કિંમતો છે, ત્યારે આપણે કહીએ છીએ 12 એ 11 પછી 12 અને 11 બંને બિંદુઓ એક જ સૂચિમાં છે. તેથી, એકને અપડેટ કરવું અન્યને અપડેટ કરશે, અને તેથી અમે આ વિશે સાવચેત બનો. જો આપણે ખરેખર એક નકલ બનાવવા માંગીએ છીએ,

તો અમે સંપૂર્ણ સ્લાઇસનો ઉપયોગ કરીએ છીએ. તેથી, આપણે કહીએ છીએ કે 12 એ પહેલાં અથવા પછી કશું સાથે 11 કોલોનના બરાબર છે, તે 0 થી L1 ની લંબાઈ પર છે, અને આ આપણને તાત્કાલિક લિસ્ટ સાથેની તાજી સૂચિ આપે છે જેમ કે 11. અને છેલ્લે, અમે જોયું કે આપણે સમાનતા નો ઉપયોગ કરી શકીએ છીએ અને બે અલગ અલગ ઓપરેટરો તરીકે તપાસ કરવા માટે છે કે શું બે નામો ફક્ત મૂલ્યમાં છે કે શારીરિક રીતે તે જ પ્રકારનું નિર્દેશ કરે છે.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રોફેસર. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટીટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 02 લેક્ચર - 04

કંટ્રોલ ફ્લો છેલ્લા થોડા પ્રવચનોમાં, અમે પાયથોન માં વિવિધ પ્રકારના મૂલ્યો જોયાં છે નંબરો સાથે અને પછી બુલિયન પર ખસેડવાની અને પછી શબ્દમાળાઓ અને યાદીઓ જેમ કે સિક્વન્સ.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 00:02)

હવે ચાલો ક્રમમાં જે ક્રમાંકો એક્ઝેક્યુટ થાય છે તેના પર પાછા જઈએ. યાદ રાખો અમે કહ્યું હતું કે એક લાક્ષણિક પાયથોન પ્રોગ્રામમાં વિધેયોની એકત્રીકરણ હશે અને તેના પછી નિવેદનો કરવામાં આવશે. સામાન્ય રીતે, પિથન દુભાષિયો તે ઉપરથી નીચે સુધી આપણે તેને જે કંઈ પણ વાંચશે તે વાંચશે. તેથી, જ્યારે તે કોઈ કાર્યની વ્યાખ્યા જુએ છે, ત્યારે તે તેને ડાયજેસ્ટ કરશે પરંતુ તેને અમલમાં મૂકશે નહીં. અમે ટૂંક સમયમાં જ વધુ વિગતવાર વિધેય વ્યાખ્યાઓ જોવા મળશે. અને જ્યારે આપણે હવે એવી કોઈ વસ્તુ પર આવીએ જે કોઈ વ્યાખ્યા નથી, તો પાયથોન તેને ચલાવવાનો પ્રયત્ન કરશે, આમાં ફંક્શનનો ઉપયોગ કરવો પડશે, જે કિસ્સામાં વિધેય વ્યાખ્યાયિત કરેલા નિવેદનો ચલાવવામાં આવશે અને તેથી વધુ. જો કે, જો અમારી પાસે આ પ્રકારની સખત સીધી લીટીનો અમારા કાર્યક્રમનો અમલ છે તો અમે ફક્ત મર્યાદિત વસ્તુઓ કરી શકીએ છીએ. આનો અર્થ એ કે અમારી પાસે પગલાંઓની અનિયમિત ક્રમ છે કે જે અમે શું કરવા માગીએ છીએ તેની અમે હંમેશા અનુસરવું જોઈએ.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 01:13)

હવે ઘણી પરિસ્થિતિઓમાં, ઘણા વાસ્તવવાદી પરિસ્થિતિઓમાં, અત્યાર સુધી જે બન્યું તે મુજબ આગામી પગલામાં ફેરફાર કરવાની જરૂર છે, વર્તમાન મૂલ્યો કે જે આપણે જોઈ રહ્યા છીએ તે પ્રમાણે. ચાલો આપણે વાસ્તવિક જીવનનું ઉદાહરણ જોઈએ. ધારો કે, તમે તમારી વસ્તુઓને સવારે બસ સ્ટોપ માટે છોડવા માટે પેકીંગ કરી રહ્યા છો, અથવા તમે તમારી સાથે એક છત્રી લેવા કે નહીં તે તમને તે દિવસે વરસાદી વાતાવરણ લાગે છે તેના પર આધાર રાખે છે. જો તમે દરરોજ છત્રી લઈ જાવ તો તમારી બેગ ભારે બની જાય છે, જો તમે ક્યારેય છત્રી ન લો, તો પછી તમે ભીના હોવાની શક્યતાને જોખમમાં મૂકે છે. તેથી, તમે આ બિંદુએ બંધ થશો અને જુઓ, હવામાનની આગાહી વાંચીને અથવા વિંડોની બહાર જોઈને ગમે તે રીતે આજે વરસાદ થવાની સંભાવના છે? જો વરસાદ પડવાની સંભાવના હોય તો, તેની ખાતરી કરો કે તમારી બેગમાં છત્રીઓ, જો તે ન હોય તો તેને મૂકી દો, અથવા જો તે પહેલાથી જ ત્યાં છે તો તેને છોડો. જો તે વરસાદની શક્યતા ન હોય તો, ખાતરી કરો કે છત્ર બેગમાં નથી, જો તે ત્યાં ન હોય તો તે સાફ છે, તેને બહાર કાઢો. તેથી, આ પ્રકારનું અમલ જે ક્રમમાં બદલાય છે જેમાં નિવેદનોનો અમલ કરવામાં આવે છે તે પ્રોગ્રામિંગ ભાષાઓમાં નિયંત્રણ પ્રવાહ તરીકે ઓળખવામાં આવે છે. જુસીડી કેસમાં અનૌપચારિક રીતે જોવા મળતા ત્રણ મૂળભૂત બાબતો છે. એક એ છે કે આપણે ફક્ત શરતી અમલનું વર્ણન કરીએ છીએ. બીજું એ છે કે આપણે કોઈ ચોક્કસ સમયની

સંખ્યાને પુનરાવર્તન કરવા માંગીએ છીએ અને આ સંખ્યા અગાઉથી જાણીતી છે. અમે આ રૂમમાંથી 10 બોક્સને તે રૂમમાં લઈ જવા માંગીએ છીએ, તેથી 10 વખત અમે અહીંથી એક સમયે એક બોક્સ લઈએ છીએ. બીજી તરફ, ક્યારેક આપણે કંઈક પુનરાવર્તન કરવા માંગીએ છીએ જ્યાં પુનરાવર્તનોની સંખ્યા અગાઉથી જાણીતી નથી. ધારો કે, અમે અમારા ચાના કપમાં ખાંડ મૂકીએ છીએ અને ખાંડ ઓગળી જાય ત્યાં સુધી તેને હલાવવા માંગીએ છીએ. તેથી, આપણે શું કરીશું, અમે તેને એકવાર જગાડીએ છીએ, જો ખાંડ હજી પણ તેને ફરીથી ખાળે છે, તો તપાસો કે હજી પણ ખાંડ અને તેથી વધુ છે, અને ખાંડ એક રાઉન્ડ પછી વિસર્જન થાય છે ત્યારે આપણે શોધીશું કે ત્યાં કોઈ ખાંડ નથી. કપ નીચે અને અમે હલાવવાનું બંધ કરશે. અહીં, અમે ઘણી વખત હલાવવાનું ક્રિયા પુનરાવર્તન કરીશું, પરંતુ અમે બે વાર અથવા પાંચ વખત તે જગાડવો પડશે કે નહીં તે અગાઉથી ખબર નથી, અમે ખાંડ ઓગળી જાય ત્યાં સુધી હલાવીએ છીએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 03:17)

ચાલો શરતી અમલ સાથે શરૂ કરીએ. પાયથોન માં શરતી એક્ઝેક્યુશન લખેલું છે જે આપણે `if statement`; નો ઉપયોગ કરીને જોયું છે. અમે `if`; `else`; હોય, તો અમારી પાસે એક શરતી અભિવ્યક્તિ છે જે મૂલ્ય સાચું અથવા ખોટી રીતે સામાન્ય રીતે તુલના કરે છે, પરંતુ તે એક કાર્યને પણ દાખલ કરી શકે છે જે ઉદાહરણ તરીકે સાચું કે ખોટા વળતર આપે છે, અને અમારી પાસે આ કોલોન છે જેનો અંત દર્શાવે છે શરત અને પછી શરતી રીતે ચલાવવાનું નિવેદન ઇન્ડેન્ટેડ છે, તેથી તે ડાબેથી બંધ છે તેથી પાયથોન જાણે છે કે આ સ્ટેટમેન્ટ આ શરત દ્વારા સંચાલિત છે. અમે આ સાથે સાથે `n` ના જૂના મૂલ્યને લઈને મીટરના એકસાથે સોંપણી કરીએ છીએ અને `n` એ `n` દ્વારા વિભાજિત કરેલ મૂલ્ય લે છે, માત્ર જો `n` દ્વારા ભાગવામાં આવે તો હાલમાં 0 નથી.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 04:03)

બીજું નિવેદન ચલાવવામાં આવે છે શરતી રીતે, જો શરત સાચી અને ઇન્ડેન્ટેશનનું મૂલ્યાંકન કરે તો `if`; ના શરીરને ડિમેટ કરે. શરીર એવા નિવેદનોનો ઉલ્લેખ કરે છે જે સ્થિતિ દ્વારા સંચાલિત હોય છે જે અમે હમણાં જ ચકાસાયેલ છે. તેથી, ચાલો આપણે આનો એક નાનકડો દાખલો જોઈએ. ધારો કે અમારી પાસે કોડ છે જે નીચે મુજબ જુએ છે; જો અમારી પાસે સ્થિતિ છે તો અમારી પાસે બે ઇન્ડેન્ટેડ નિવેદનો છે - નિવેદન 1 અને નિવેદન 2, અને પછી અમારી પાસે એક ત્રીજા નિવેદન છે જે ઇન્ડેન્ટેડ નથી. ઇન્ડેન્ટેશન પાયથોન ને કહે છે કે આ બે સ્ટેટમેન્ટ શરતી રીતે આ શરતની કિંમત પર આધારિત છે. જો કે, સ્ટેટમેન્ટ 3 આ શરત દ્વારા સંચાલિત નથી, કારણ કે તે એ જ સ્તરે પહોંચાડાય છે જો તે પ્રમાણે. તેથી, જ્યાં તમારો ટેક્સ્ટ રહેલો છે તેનું વર્ણન કરીને સંચાલિત કરીને, તમે આ શરત દ્વારા શાસનની શરૂઆત અને અંત નક્કી કરી શકો છો. પરંપરાગત પ્રોગ્રામિંગ ભાષામાં, તમારી પાસે અમુક પ્રકારની વિરામચિહ્ન હોત, ખાસ કરીને બ્લોકની શરૂઆત અને અંત સૂચવવા માટે તાણ જેવું કશુંક, જે સ્થિતિ દ્વારા સંચાલિત થાય છે. પાયથોન વિશેની એક સરસ વસ્તુઓ જે શીખવા માટે અને વાપરવા અને વાંચવા માટે સરળ બનાવે છે તે આ વિરામચિહ્નો અને વાક્યરચના વિષયક વસ્તુઓ સાથે વિતરણ છે જે પ્રોગ્રામિંગ ભાષાઓને સમજવા માટે મુશ્કેલ બનાવે છે. તેથી, જ્યારે તમે પ્રોગ્રામિંગ ભાષા શીખવાનો પ્રયાસ કરી રહ્યાં છો, ત્યારે તમે પ્રોગ્રામિંગ શરૂ કરવા માંગતા હો અને કોલન્સ, અર્ધવિરામ, ઓપન કૌસ અને ક્લોઝ બ્રેસીસ વગેરેને ક્યાં મૂકવા તે સમજવામાં ઘણો સમય નથી.

પાયથોન આને ઘટાડવાનો પ્રયત્ન કરે છે અને જો તમે ચોક્કસ પ્રકારની વસ્તુઓ કરી રહ્યા હો તો તે કોડને લખવા અને લખવા માટે એક આકર્ષક ભાષા બનાવે છે પાયથોન પાસે આ નહીં હોય અને પછી પરિણામ તરીકે આપણે વધુ કલીનર પ્રોગ્રામ જોશું. એક વાત જે આપણે પહેલાં પર ભાર મૂક્યો છે અને હું કહીશ કે આ શાખા એકરૂપ હોવી જોઈએ; બીજા શબ્દોમાં કહીએ તો, તે જગ્યાઓની સંખ્યા જ હોવી જોઈએ. તમે જે સૌથી ખતરનાક વસ્તુ કરી શકો છો તે ટેબો અને જગ્યાઓનો મિશ્રણ વાપરવાનો છે, જ્યારે તમે તમારા કીબોર્ડ પર ટેબને દબાવો છો તો તે કેટલાંક ખાલી જગ્યાઓ દાખલ કરે છે જે તમને સ્ક્રીન પર દેખાય છે ત્યારે તે તમને સમાન લાગે છે, પરંતુ પાયથોન મૂંઝવણ કરતું નથી ટેબ્સ અને જગ્યાઓ તેથી, એક ટેબ ત્રણ જગ્યાઓ અથવા ચાર જગ્યાઓ અથવા તમે જે સ્ક્રીન પર જુઓ છો તે બરાબર નથી; અને વધુ ચિંતિત વસ્તુ એ છે કે પાયથોન તમને કોઈ પ્રકારની ભૂલ સંદેશ આપશે જે સમજવામાં ખૂબ જ સરળ નથી. તેથી, દાખલા તરીકે, બરાબર કેટલાક યુનિફોર્મ વ્યૂહનો ઉપયોગ કરીને યાદ રાખવાથી સ્થિતિમાં ન આવવા માટે ખૂબ ઉપયોગી છે, જ્યારે તમે આવા નેસ્ટેડ બ્લોક ધરાવતા હોય ત્યારે ઇન્ડેન્ટ બે જગ્યાઓ.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 06:37)

ઘણી વાર, અમારી પાસે બે વિકલ્પો છે; જો એક શરત સાચું હોય તો એક લેવાય છે, અને જો તે શરત સાચી ન હોય તો અન્યને લેવા માટે. જો વરસાદ થવાની સંભાવના હોય તો છત્રી બેગમાં છે તેની ખાતરી કરો, તો ખાતરી કરો કે છત્રી બેગમાં નથી. આ કિસ્સામાં ઉદાહરણ તરીકે, જો બાકીની નથી 0 m અને n માટે નવા મૂલ્યો સાથે ચાલુ રહે તો બાકીની 0 હોય તો, આપણે બે મૂલ્યો સાથે નાના એટલે કે જીસીડી મેળવ્યું છે. આ વિશેષ પ્રકારના શબ્દનો ઉપયોગ કરીને દર્શાવવામાં આવે છે જે કોલોન સાથે ફરીથી ઇંગ્લીશની જેમ છે અને ફરીથી, અમે બીજું શું છે તે દર્શાવવા માટે માળામાં છે અને જો અને બીજું એ જ સ્તર પર નેસ્ટ થયેલ હોવું જોઈએ. બીજું વૈકલ્પિક છે, તેથી બીજું વગર 'if'; હોઈ શકે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 07:21)

ટેક્નિકલી રીતે કહીએ છીએ, 'if statement'; માં આપણે જે શરત મૂકી છે તે એક અભિવ્યક્તિ હોવી જોઈએ જે સ્પષ્ટપણે બુલિયન મૂલ્યને સાચી અથવા ખોટી આપે છે. પરંતુ અન્ય ઘણી પ્રોગ્રામિંગ ભાષાની જેમ, પાયથોન આને થોડો આરામ આપે છે અને જુદા જુદા પ્રકારના અભિવ્યક્તિઓ આપે છે જે વિવિધ પ્રકારો ધરાવે છે જેમ કે આપણે અત્યાર સુધી નંબરોની જેમ જોયેલી છે અને સૂચિને સાચી કે ખોટા તરીકે પણ અર્થઘટન કરી છે. ખાસ કરીને, કોઈપણ સંખ્યાની, કોઈ પણ અભિવ્યક્તિ, જે કોઈ નંબર 0 આપે છે, મૂલ્ય 0 નું કોઈ આંકડાકીય અભિવ્યક્તિને ખોટી ગણવામાં આવે છે. તેવી જ રીતે, કોઈપણ ખાલી ક્રમ જેમ કે ખાલી શબ્દમાળા અથવા ખાલી સૂચિને પણ ખોટી ગણવામાં આવે છે. અને જે કંઈપણ આ કિસ્સામાં નથી, તેથી જો મારી પાસે કોઈ સંખ્યા તરીકે નોનઝેરો મૂલ્ય હોય અથવા જો મારી પાસે બિન-અક્ષમતાવાળી શબ્દમાળા હોય, તો સાત અક્ષરો સાથેની એક સ્ટ્રિંગ અથવા બિન-મૂલ્યવાન સૂચિ ત્રણ કિંમતો સાથેની સૂચિ હોય તો આ બધાને સાચું તરીકે અર્થઘટન કરવામાં આવશે. હું તેમને શરતમાં લાકડી રાખું છું. તેથી, આ આપણા અભિવ્યક્તિઓ અને કોડને સરળ બનાવી શકે છે. મીટર ટકા એન કહીને બદલે અમે એમ ટકા n ની સમાન ન કહીએ. યાદ રાખો કે જો તે 0 બરાબર નથી, તો તે સાચું છે. જો આ શરત સાચું હોય તો શું એમ માની લેવું છે કે મીટર એન એ નોઝરોયો મુલ્ય છે, અને જો તે નોનઝેરો મુલ્ય હોય તો આપણે મીટર અને એન ને

બદલવા માંગીએ છીએ, તેથી આપણે એમ કહી શકીએ કે m ટકા એન. તો આ શોર્ટકટ છે, હવે તેનો ઉપયોગ તમે કાળજીપૂર્વક કરી શકો છો જો તમે તેનો ઉપયોગ કરો છો અને જો તમે પરિચિત છો તો શું થઈ રહ્યું છે, તે આ રીતે તમે જે રીતે લખી શકો તે સરળ કરી શકો છો, પણ જો તમે જે શીખી રહ્યા છો તે પરિચિત ન હોય તો, તમે ભૂલો કરી શકો છો તેથી, જો તમને શંકા હોય તો સંપૂર્ણ સરખામણી લખો; જો તમને શંકા નથી અથવા જો તે ખૂબ જ સ્પષ્ટ છે, તો પછી શોર્ટકટ સાથે જાઓ.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 09:12)

અહીં એક સામાન્ય પરિસ્થિતિ છે જે થાય છે; ક્યારેક અમે બે શરતો પૈકી એકની વચ્ચે તપાસ કરવા નથી માગતા, પરંતુ ઘણી શરતોમાંની એક તેથી, ધારો કે આપણી પાસે x , x નામનું મૂલ્ય છે જે મૂલ્ય 1, 2, 3 અથવા 4 લઈ શકે છે અને તેના આધારે કે તે 1, 2, 3 કે 4 છે, તમે ચાર અલગ અલગ બાબતો કરવા માંગો છો જેની સાથે આપણે એફ 1, એફ 2, એફ 3 અને એફ 4 તેથી, જો આપણે આ ચાર માર્ગની પસંદગીનો અનુકરણ કરીએ તો, જો અને બીજું વાપરીશું તો અમારે અમુક સરખામણી પ્રથમ કરવી પડશે. ધારો કે, આપણે પહેલા તપાસો કે x બરાબર 1 છે, તો આપણે એફ 1 ને બોલાવીએ છીએ. જો એક્સ 1 નથી, તો તે અન્યમાંના એક છે, તેથી આ બધું બીજું એક જાય અને બધું નેસ્ટેડ કરે છે. તો પછી આપણે આ કેસમાં તપાસ કરીએ, જો x બરાબર 2 હોય તો આપણે એફ 2 કરીએ છીએ, અમારી પાસે 3 કે 4 છે, તેથી હવે આ બધું ફરી એકવાર નેસ્ટ કરેલું છે. અને છેવટે, આપણી પાસે x બરાબર 3 અથવા 3 છે, તેથી આ x બરાબર 4 છે અને તે પછી આપણે પૂર્ણ કર્યું છે, પરંતુ આની સાથેની મુખ્ય સમસ્યા એ છે કે આ બધા કોડમાં પ્રથમ ઇન્ડેન્ટેડ છે. તેથી, જેમ તમે આ નેસ્ટેડમાં જાઓ છો જો માળખું આ મલ્ટી-વે શાખાને અનુકરણ કરે છે કે જે અમારી પાસે આવશ્યક રૂપે ચાર માર્ગ શાખા છે, તેમાંથી ચારમાંથી એક મૂલ્ય લઈ શકે છે, જેમાં આ ચાર મૂલ્યોમાંની દરેક તમે ચાર અલગ અલગ વસ્તુઓ કરવા માંગો છો. જો આપણે તેને 4, 3, 2-રસ્તાની નિર્ણયો લઈને તેને અનુકરણ કરીએ તો, આપણે 1 કે નહીં 1 તપાસીએ, તો આપણે 2 અથવા 2 ની તપાસ કરીશું, અને પછી આપણે 3 કે નહીં 3 તપાસીએ તો આ બિહામણું માળો હોય અને બીજું, અમારી પાસે આ છે બીજું જો એક દ્વારા તરત અનુસરવામાં

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 10:44)

પાયથોન પાસે આ માટે શોર્ટકટ છે જે બીજું અને બીજું ચેક ELIF માં જોડવાનું છે. તેથી, જમણી બાજુ પર તે બરાબર એ જ છે જે ડાબી બાજુએ છે કારણકે પાયથોન નો સંબંધ છે. તે તપાસ કરે છે કે શરત x બરાબર 1 છે, જો x બરાબર 1 હોય તો તે એફ 1 ને બોલાવશે, અન્યથા, તે બાકીના જગાવે છે હવે બાકીના, અમે હમણાં જ બીજું પતન કર્યું છે અને જો એક નવી શરત સીધી તપાસ કરો; જો આ શરત પછી આ કામ કરે છે; નહિતર, તે બાકીના અને તેથી પર તપાસ કરશે. તેથી, આપણે આ રીતે એલિફ દ્વારા પુનરાવર્તિત અન્યને બદલી શકીએ છીએ, અને તે કોડને વધુ વાંચનીય બનાવી શકે છે કારણ કે તે આપણને કહે છે કે અમે વાસ્તવમાં મલ્ટી-વેસ ચેક કરી રહ્યા છીએ. અને નોંધ કરો કે એલિફનો છેલ્લો શબ્દ ફરીથી બીજું છે. તેથી, જો તમે સાત અલગ અલગ વિકલ્પો કહી છે, અને તમે ફક્ત ખાસ વસ્તુઓ કરી રહ્યા છો તેથી 1, 2, 3, અને 4, 5, 6, 7 એ બધા સમાન હોય છે, પછી આપણે બીજાનો ઉપયોગ કરી શકીએ છીએ; અમે સ્પષ્ટ રીતે બીજા બધા વિકલ્પોની ગણતરી કરી નથી. તેથી, આપણી પાસે ઘણી સ્પષ્ટ શરતો છે જે અમે તપાસ કરીએ છીએ. તેથી, આ રીતે અને નોટિસ દ્વારા અથવા તેઓ બરાબર ન હોવા જોઈએ. તેથી, આપણી પાસે ઘણી સ્પષ્ટ શરતો છે જે આપણે અને જો અનુક્રમ અથવા

એલિફ્સ સાથે તપાસ કરીએ છીએ, અને છેવટે, અમારી પાસે બીજું છે, બીજું ફરીથી વૈકલ્પિક છે, જો તે સામાન્ય હોય તો, પરંતુ મુખ્ય વસ્તુ તે આ લાંબા અંતર્દેશને ટાળે છે અનુક્રમણિકા જે કોડને પછીથી વાંચવા માટે ખૂબ જ મુશ્કેલ બનાવે છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 12:07)

આપણે જે કંટ્રોલ ફ્લોને જોયું તે પ્રથમ શરત છે અને અન્ય પ્રકારની લૂપ છે. લૂપમાં, આપણે પુનરાવર્તિત સંખ્યામાં કેટલીક વખત કરવા માંગીએ છીએ. તેથી, સૌથી વધુ મૂળભૂત આવશ્યકતા એ કોઈ ચોક્કસ સમય છે દાખલા તરીકે, આપણી પાસે નિવેદન છે કે જેના માટે પાંચમાં કીવર્ડ છે. તેથી, શું પાંચથોન કહે છે તે એક નવું નામ લે છે અને તે મૂલ્યોના અનુક્રમ દ્વારા ચલાવવા દો, અને આ શ્રેણીમાં પ્રત્યેક મૂલ્ય માટે આ એક વખત અધિકાર ચલાવે છે. તેથી, આ અનુક્રમમાં છે, તે y દ્વારા ગુણાકાર કરી રહ્યું છે, પછી પરિણામ 2 અને પછી પરિણામ 3 અને આ રીતે. આનાં અંતમાં વાચ ટાઇમ્સ 1 વખત, 2 ગુણ્યા 3 ગુણ્યા 4 હશે અને આપણી પાસે z વત્તા 1 વત્તા 1 વત્તા 1 વત્તા 1 - ચાર વખત રહેશે કારણ કે દરેક વખતે આપણે 1 થી z ઉમેરી રહ્યા છીએ. તેથી, આ લૂપનું પરિણામ હોવું જોઈએ. મુખ્ય વસ્તુ એ બરાબર જેવો છે, જો આપણે લૂપની સંસ્થાને ચિહ્નિત કરવા માટે ઇન્ડેન્ટેશન કર્યું છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 13:05)

વસ્તુઓ પુનરાવર્તન માટે સૌથી સામાન્ય બાબત એ છે કે કંઈક એન ટાઇમ કરવું. તેથી, સૂચિ 1 થી n લખવાને બદલે, આપણે જે જોયું તે એ છે કે આપણે આ બિલ્ટ ઇન ફંક્શન રેન્જનો ઉપયોગ કરીને આ સૂચિની સમાન કંઈક બનાવી શકીએ છીએ. શ્રેણી 0, 1, અમે જણાવ્યું હતું કે તે 0 થી શરૂ થાય છે અને તે ફોર્મ 0, 1, 2 ના બંધારણ n નો 1 પર બંધ કરે છે. તે પાંચથોનમાં સમાન સ્થાયી સંમેલન જેવું જ છે જે કહે છે કે યાદીમાં સ્થાનો 0 થી 0 ની સંખ્યાની લંબાઈની સંખ્યા રેન્જ પણ 0 થી n સુધી નથી, પરંતુ 0 થી n ની 1 ઘાત. તેથી, સૂચિમાં હું લખવા માટે, અમે રેન્જમાં હું લખી શકીએ છીએ, અને આ પાંચથોન ડ્રિસ્ટિકોણથી તે જ વસ્તુ છે, ક્યાં તો આપણે આ સુવિધાની સૂચિ અથવા રેન્જ ફંક્શન દ્વારા આપવામાં આવેલા ગર્ભિત અનુક્રમ પર આ નવું નામ શ્રેણી આપી શકીએ. સામાન્ય શ્રેણીમાં i , j , જેમ કે સ્લાઇડ i થી j , i થી શરૂ થાય છે અને j ઓછા 1 સુધી જાય છે. આપણે શ્રેણી કાર્યો પણ કરી શકીએ છીએ જે નીચે ગણતરી કરી શકે છે અને આપણે રેન્જ ફંક્શનો ધરાવી શકીએ છીએ જે મૂલ્યને અવગણવા માટે આપણે દરેક વૈકલ્પિક મૂલ્ય અને તેથી પર અમે થોડા સમય પછી શ્રેણી પર આ ભિન્નતા જોશો, પરંતુ હવે ફક્ત નોંધ લો કે ક્યાં તો આપણે નિવેદન માટે હોઈ શકે છે જે સ્પષ્ટપણે મૂલ્યોની સૂચિમાંથી પસાર થાય છે. તેથી, અમે એક સૂચિ આપી શકીએ છીએ અને તે સૂચિમાં દરેક મૂલ્યમાંથી પસાર થવા માટે કહીએ છીએ અથવા રેન્જ ફંક્શનનો ઉપયોગ કરીને આપણે n મૂલ્યોની શ્રેણી બનાવી શકીએ છીએ.

(સ્લાઈડ ટાઇમ નો ઉલ્લેખ કરો: 14:35)

ચાલો આનું એક સરળ ઉદાહરણ જોઈએ. ધારો કે, અમે સંખ્યા n ના તમામ પરિબળો શોધવા માંગીએ છીએ, બાકીની સંખ્યાઓ વગર n ને વિભાજીત કરો. જ્યારે આપણે જીસીડ (Gcd) કર્યું ત્યારે યાદ રાખતાં, બધા ભાગલાકો અથવા સંખ્યાના પરિબળો 1 અને n ની વચ્ચે હોવું જોઈએ; આપણી પાસે n

કરતાં મોટી સંખ્યા હોઈ શકતી નથી, જે n નો પરિબલ છે. પરિબલોની સૂચિ ચકાસવાની એક સરળ રીત એ છે કે દરેક સંખ્યા 1 થી n ની અજમાવી જુઓ, અને જો તે વિભાજિત થાય છે, તો તે અહીં એક ખૂબ જ સરળ કાર્ય છે. તેથી, આપણે n નો પરિબલો કહેવાય છે જે તમામ પરિબલોની સૂચિને પરત આપી રહ્યું છે. આંતરીક રીતે અમે આ નામની યાદીમાં નામના ધ્વજનો ઉપયોગ કરીએ છીએ. ફિલિસ્ટ, પરિબલોની આગલી સૂચિ શરૂઆતમાં ખાલી છે. અને હવે ધ્યાનમાં રાખવું કે તમામ પરિબલો 1 અને n વચ્ચે આવેલા છે, અમે તમામ નંબરોને 1 થી n સુધી બનાવીએ છીએ, અને યાદ રાખો કે શ્રેણીની ઉપરની બાઉન્ડને n વત્તા 1 હોવી જરૂરી છે, કારણ કે રેંજ ફંક્શન નીચે એક અટકે છે સંખ્યા જે જમણા હાથની બાજુ છે તેથી, આ ક્રમ 1, 2, 3 ને n સુધી બનાવશે. અને આપણે શું તપાસીએ છીએ કે જ્યારે બાકીની કોઈ સંખ્યા ન હોય તો જ્યારે આપણે એન દ્વારા વિભાજિત થાય, તો આપણે આ યાદીમાં હું ઉમેરો. અને યાદ રાખો કે સૂચિ માટે અને અનુક્રમ માટે પ્લસ સંયોજન છે; તે વાસ્તવમાં સૂચિમાં એક મૂલ્ય ઉમેરે છે, અને આ અમને એક નવી સૂચિ પરત કરવા દે છે આનો અંત, આપણે n ના તમામ પરિબલોની ગણતરી કરી છે અને તેમને યાદી તરીકે પરત આપીએ છીએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 16:07)

પરંતુ જેમ જેમ આપણે આજે પ્રવચનોની શરૂઆતમાં કહ્યું હતું, ક્યારેક આપણે કપમાં ખાંડને ચળવળ જેવા અગાઉની સંખ્યાને જાણ્યા વગર કેટલીકવાર કેટલીક વખત કરવા માંગીએ છીએ. અમે જુસીડી સાથે આનો એક ઉદાહરણ જોયો છે. તેથી આપણે શું કર્યું, જ્યારે અમારી પાસે અન્ય સ્ટેટમેન્ટ છે જેમકે કહેવાય છે. જ્યારે કંઈક શરત થાય તેટલી લાંબી એક્ઝેક્યુટ કરે છે, તેથી અમે આ દેહને એક્ઝેક્યુટ કરીએ છીએ, જ્યાં સુધી આ સ્થિતિ સાચી મૂલ્યાંકન કરે, અને પછી જ્યારે આપણે આ અમલ કરવાનું સમાપ્ત કરીએ છીએ ત્યારે અમે પાછા આવીએ છીએ અને ફરી તપાસ કરીએ છીએ કે આ સ્થિતિ સાચું છે કે નહીં. તેથી ભય એ છે કે દર વખતે આપણે શરતમાં પાછા આવીએ ત્યારે તે સાચું હશે અને આ બાબત ક્યારેય સમાપ્ત થઈ જશે નહીં કારણ કે આપણે જાણીએ છીએ કે તે બરાબર ચલાવશે. તેથી, જ્યારે અમે લૂપ માટે શરૂ કરીએ છીએ ત્યારે અમે ફિક્સ સિક્વન્સ આપીએ છીએ કે ફિક્સ ક્રમની ફિક્સ લંબાઈ છે અને તેથી આપણે તે ઘણા ફાંસીની લૂપ સમાપ્ત કરવી જોઈએ. જ્યારે અમે પાછા આવો ત્યારે અમે ફરી સ્થિતિ તપાસીએ છીએ, પરંતુ એવી સંભાવના છે કે આ સ્થિતિ ક્યારેય સાચી નહીં બને. આપણને ખાતરી કરવી પડે છે કે જ્યારે આપણે જ્યારે લૂપ લખીએ છીએ ત્યારે કોઈક આ ક્રમાંકની અંદર આ એક્ઝેક્યુટને અમલમાં મૂકવા માટે છેવટે આ વસ્તુ ખોટી હોવી જોઈએ, કારણ કે જો આ વસ્તુ ક્યારેય ખોટું નહીં થાય, શરત ક્યારેય નહીં થાય, લૂપ સમાપ્ત નહીં થાય.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 17:29)

જુસીડી સાથે જે ઉદાહરણ આપણે જોયું છે તે અમે અત્યાર સુધીમાં જે લખ્યું હતું તેના બદલાતા હતા. અમે પ્રથમ ચેક અને સ્વેપ પ્રથમ, કે જેથી મોટી સંખ્યા પ્રથમ છે અને હવે અમે તપાસો કે મોટી સંખ્યા નાના દ્વારા વિભાજિત છે. તેથી, જ્યાં સુધી આ કિસ્સો ન હોય ત્યાં સુધી, આપણે m અને n ના મૂલ્યોનું વિનિમય કરીએ છીએ. અમે મીટર નાના સંખ્યા n નો નિર્દેશ કરે છે, અને અમે બાકીના માટે n બિંદુ બનાવીએ છીએ, જે હજી નાની હશે; અને છેવટે આ વસ્તુ ખોટી બની જશે, આપણે એવી પરિસ્થિતિ મેળવીશું જ્યાં n મીટરનું વિભાજન થશે અને બાકીની 0 હશે અને તે સમયે અમને જુસીડી મળી જશે. આ એક સરળ ઉદાહરણ છે, યાદ રાખો કે અમારા પહેલાંના સંમેલન દ્વારા અમે એમ પણ લખી શકીએ

છીએ જ્યારે મીટર એન એ આ વસ્તુ બનાવશે. તેથી, આપણને આ સ્પષ્ટ 0 ની બરાબર જરૂર નથી, કારણ કે મૂલ્ય એમ ટકા n જો તે 0 ન હોય તો તેને સાચું ગણવામાં આવે છે અને લૂપ એક વધુ સમય જાય છે, પરંતુ હવે કેટલીક પરિસ્થિતિઓમાં તે કદાચ સહેલું ન પણ હોઈ શકે આ વાંચો. ફક્ત તમારા પોતાના માટે અને તમારા કોડને વાંચતી વ્યક્તિને સમજાવવા માટે, ફક્ત 0 ની બરાબર ન કહી શકાય તેવું ઉપયોગી હોઈ શકે છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 18:41)

સારાંશ માટે, પાચથોન ઈન્ટરપ્રીટર સામાન્ય રીતે અનુક્રમમાં ઉપરથી નીચે સુધીનો કોડ અમલમાં મૂકે છે. અમે ત્રણ રીતે જોયું છે તે નિયંત્રણ ફ્લોને બદલી શકીએ છીએ. એક `if` સ્ટેટમેન્ટ; નો ઉપયોગ કરી રહ્યું છે, જે શરતી રીતે ચલાવે છે અને આ `if` સુધી વિસ્તરે છે જે અમને મલ્ટી-વે બ્રાન્ચ કરવા દે છે. પછી નિવેદન માટે છે જે આપણને કોઈ ચોક્કસ સંખ્યાને પુનરાવર્તન કરવાની પરવાનગી આપે છે કે જે શ્રેણી અથવા સૂચિમાંથી મૂલ્યોનો ક્રમ પૂરો પાડે છે. અને પછી આપણી પાસે એક પુનરાવર્તન છે જે એક શરત પર આધારિત હોય છે જ્યારે અમે એક શરત મૂકીએ છીએ અને પછી દરેક વખતે જ્યારે શરીરને ફરીથી એક્ઝિક્યુટ કરવામાં આવે છે ત્યારે સ્થિતિ ફરીથી ચકાસવામાં આવે છે તેથી આ સાચું છે કે શરીર પુનરાવર્તન કરે છે.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રોફેસર. માધવન મુકુંદ,
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 02 લેક્ચર - 05

ફંક્શન્સ અમે જોયું છે કે, ક્યારે અને ક્યારે? અમે શરતી અમલ કરી શકે છે, અમે વારંવાર અમલ કરી શકે છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 00:01)

અમારા લાક્ષણિક પાયથોન પ્રોગ્રામમાં છેલ્લો ઘટક કાર્ય છે. કાર્ય શું છે? ફંક્શન નિવેદનોનો સમૂહ છે જે આપેલ કાર્ય કરે છે. અલબત્ત, અમે મુખ્ય પ્રોગ્રામના ભાગ રૂપે કાર્ય કોડ લખી શકીએ છીએ, પરંતુ તેને અલગ કરીને આપણે તાર્કિક રીતે કામના એકમોને અલગ કરી શકીએ છીએ અને ઘણી વખત આ વિધેયોને વિવિધ દલીલો સાથે વારંવાર કહેવામાં આવે છે. તેથી, તેઓ ગણતરીના એકમનું નિર્માણ કરે છે જે સમયાંતરે વારંવાર ઉપયોગ કરી શકાય છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 00:43)

આપણે અસાધારણ રીતે જોવા મળે છે તે રીતે ડિફ સ્ટેટમેન્ટનો ઉપયોગ કરીને કાર્યને વ્યાખ્યાયિત કરીએ છીએ. તેથી વ્યાખ્યા કાર્યના નામને વ્યાખ્યાયિત કરે છે, આ કિસ્સામાં આપણે તેને ફક્ત એક કહેવાય છે, સામાન્ય રીતે આપણે તેને વધુ અર્થપૂર્ણ નામો આપીશું. પછી તે કહે છે કે આ ફંક્શન ત્રણ મૂલ્યોને ઇનપુટ તરીકે લે છે, તેથી તે પરિમાણો અથવા દલીલો કહેવામાં આવે છે. તેથી, સૌપ્રથમ એકને કહેવાય છે, બીજો બી કહેવાય છે, અને ત્રીજાને c કહેવામાં આવે છે, અને કાર્યના એક ભાગમાં, b અને c એ મૂલ્યો નો સંદર્ભ લેશે જે આપેલ કાર્ય માટે આ કાર્યમાં પસાર થાય છે. કોલ કરો એક વિધેયની અંદર અમારી પાસે એક એવું નિવેદન હોઈ શકે છે કે જેમને વળતર કહેવાય છે. વિધેયનો ભાગ ઇન્ડેન્ટેડ છે, જેમ કે, ક્યારે અને માટે, અને પરત સ્ટેટમેન્ટ જો આવી હોય, તો તે કહે છે કે આ બિંદુએ ફંક્શનનું અમલ સમાપ્ત થશે અને તમે જ્યાંથી ફંક્શનને બોલાવ્યું છે ત્યાં પાછા આવશે. નામમાં મૂલ્ય પરત કરવું v. આ કોઈપણ અભિવ્યક્તિ હોઈ શકે છે; અમે ફક્ત સતત વળતર અથવા વી વત્તા એક અથવા જે કંઈપણ પરત કરી શકે છે

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 01:50)

જ્યારે આપણે કોઈ ફંક્શનને બોલાવીએ ત્યારે આપણને દલીલો માટે મૂલ્ય પાસ કરવી પડશે, અને તે વાસ્તવમાં બરાબર એ જ રીતે નામ માટે તે મૂલ્ય તરીકે કાર્ય કરે છે. ધારીએ કે, આપણી પાસે કાર્ય છે જે x લે છે અને તેને પાવર n માં ઉઠાવે છે. ચાલો આપણે આ કાર્યને ફક્ત સમજવા માટે જોઈએ કે કોડ શું કરી રહ્યું છે. અમે ધારીએ છીએ કે જવાબની કિંમત 1 છે. અને હવે ઘણા બધા માટે, કારણ કે ત્યાં 0 થી n ઓછા 1 ની રેન્જમાં છે, આપણે એક્સમાં ગુણાકાર કરીએ છીએ તેથી આપણે x વખત, x વખત, x એક વખત મેળવીએ છીએ. દરેક વખતે આપણે આ લૂપમાંથી પસાર થઈએ છીએ ત્યારે આપણે વધુ એક ગુણાકાર કરીએ છીએ અને છેવટે આપણે જે જવાબ મેળવ્યો છે તે પાછો મેળવે છે. હવે આપણે જે રીતે આ ફંક્શનનો ઉપયોગ આપણા કોડમાં કરીશું તે ફોર્મની અભિવ્યક્તિ લખવાનું છે, પાવર 3, 5 કહે છે, તેથી આનો મતલબ એ કે 3 નો ઉપયોગ કરવા માટે x અને 5 નો ઉપયોગ n નો ઉપયોગ કરવો જોઈએ અને આપણે પછી આ કોડને x બરાબર 3 અને n બરાબર 5 વડે ચલાવો. વાસ્તવમાં, તમે કલ્પના કરી

શકો છો કે જ્યારે આપણે આ કોડ ચલાવીએ છીએ, તે આ પ્રમાણે છે કે આ કોડ અમારા પ્રોગ્રામમાં આ બિંદુએ આ અસાઇનમેન્ટ દ્વારા આગળ છે. તેથી, આ એસાઇનમેન્ટ મૂળભૂત રીતે કહે છે કે નામ x ની મૂલ્ય 3, સેટ n થી 5 દ્વારા પસાર થયેલ મૂલ્ય પર સુયોજિત કરે છે. જ્યારે તમે કોઈ કાર્ય કોલ કરો છો ત્યારે આ સૌપણી અસરકારક રીતે થાય છે. અને તે અસાઇનમેન્ટ હોવાથી, તે નિયમિત કેસમાં અસાઇનમેન્ટની જેમ વર્તે છે.

(સ્લાઇડ સમય જણાવો: 03:32)

ખાસ કરીને તે જ નિયમો અમર્યાદિત અને નિર્વિકાર કિંમતો માટે લાગુ પડે છે. યાદ રાખો, આપણે જ્યારે કહ્યું કે જ્યારે આપણે x બરાબર વાચ લખીએ છીએ, જ્યારે તે વેરિયેબલ હોય તો તે y માં વેલ બદલી શકતો નથી, તો આપણે કિંમતની નકલ કરીએ છીએ અને આપણને x માં નવી નકલ મળે છે, તેથી x અને y માં મૂલ્ય જુદું છે તેથી આ એ છે કે તે અસમર્થનીય છે. અને જો તે પરિવર્તનક્ષમ છે, તો અમે જણાવ્યું હતું કે અમે નકલ નથી, અમે કિંમત શેર; એટલે કે, બંને નામો મૂલ્યની સમાન નકલ તરફ નિર્દેશિત કરશે, તેથી એકમાં ફેરફાર એ અન્યમાં ફેરફાર પણ કરશે; તે સૂચિ જેવી પરિવર્તનીય વસ્તુઓ સાથે થાય છે. અવિરત મૂલ્યો અમારા કેસમાં કોલિંગ બિંદુ પર અસર કરશે નહીં અને પરિવર્તનક્ષમ કિંમતો અસર કરશે. તે એવું છે કે અમે ફોન ફંક્શનમાં અભિવ્યક્તિ અથવા નામનું સૌપણી કરી રહ્યા છીએ, વિધેયમાં નામને બોલાવીને. તેથી, જો કાર્ય તે નામમાં ફેરફાર કરે છે, તો તે નામનું મૂલ્ય; જો તે અમર્યાદ મૂલ્ય છે, તો અહીં કંઈ થશે નહીં, જો તે પરિવર્તનીય મૂલ્ય કંઈક થશે તો.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 04:36)

તો, આ બિંદુને સમજાવવા માટે અહીં એક સરળ કાર્ય છે. આ કાર્યનો ઉદ્દેશ યાદીને અપડેટ કરવાનો છે. તેથી, હું તમને એક સૂચિ આપેલી છે જે આ ફંક્શન 1 માં કહેવામાં આવે છે અને હું તમને એક પોઝિશન આપું છું જે હું છું અને હું શું કરવા માંગુ છું તે નવી મૂલ્ય દ્વારા જે પણ છે તે બદલવું છે. તેથી મને ત્રણ દલીલો મળે છે, 1 એક સૂચિ છે, અને પછી હું પદની અનુક્રમણિકા છું, અને છેવટે v એ બદલવાની કિંમત છે. તો, આપણે શું કરીએ? પ્રથમ તપાસો કે ઇન્ડેક્સ માન્ય ઇન્ડેક્સ છે. અમે તપાસ કરીએ છીએ કે તે 0 અને એલ ઓછા 1 ની વચ્ચે છે. તેથી તે 0 બરાબર કરતા મોટો છે અને તે 1 ની લંબાઈ ઓછી છે. જો એમ હોય તો, આપણે શું કરીએ છીએ તે મૂલ્ય v દ્વારા આપણે i નું સ્થાન બદલ્યું છે જે અમને મળ્યું છે અને અમે એ વાત સાચી ઠરે છે કે આ અપડેટ સફળ થયું છે. હવે જો હું આ શ્રેણીમાં નથી તો અમે કોઈ અપડેટ કરી શકતા નથી. તેથી, આપણે શું કરીશું તે અસરકારક રીતે ખોટી રીતે ફરે છે. આ ફક્ત એવું જ કહે છે કે સુધારો કામ કરતો નથી અને પછી વ્યક્તિ, કોડનો ભાગ જે આને બોલાવી શકે છે તે સમજી શકે છે કે કંઈક ખોટું થયું છે અને કદાચ ખોટું શું થયું છે તે ઇન્ડેક્સ માન્ય શ્રેણીમાં નથી. પરંતુ ફક્ત અસ્પષ્ટ મૂલ્યો સાથે શું થાય છે તે સમજાવવા માટે, આ કિસ્સામાં અમે કોઈ સારા કારણ માટે પણ અપડેટ કરી રહ્યા છીએ કારણ કે મૂલ્ય v એ વત્તા 1 ની હોવું જોઈએ. તેથી, યાદ રાખો કે વીને અહીં મૂકવા માટેની કિંમત પસાર થઈ રહી છે અને અમે સામાન્ય રીતે એમ ધારી રહ્યા છીએ તે વી એક નિર્વિકાર મૂલ્ય હશે. ચાલો ધારીએ કે આપણે તેને હમણાં કહીએ છીએ, તેથી આપણે જે ઉપયોગ કરીએ છીએ તે સંખ્યાઓ 3, 11, 12 ની સૂચિ બનાવી છે અને ત્યારબાદ આપણે આ 12 ને 8 દ્વારા બદલવા માંગીએ છીએ. તેથી, ફક્ત દલીલ ખાતર આપણે સૌપ્રથમ સેટ અપ કર્યું એક નવું નામ z જે 8 કહેવાય છે અને આપણે કહીએ છીએ કે સ્થાને 2 માં સૂચિ અપડેટ કરો, તેથી યાદ રાખો કે સ્થિતિ 0, 1, 2 છે. તેથી, ઝેડની કિંમત દ્વારા સ્થાને 2 માં સૂચિને અપડેટ કરો. અને પછી આપણે કહીએ છીએ કે આ જ સૂચિ સ્થિતિ 4 પર z ની મૂલ્ય દ્વારા અપડેટ કરો. હવે જો આપણે જોયું કે મૂલ્યો 4 બરાબર છે તો આ નિષ્ફળ જશે, તો તે બદલે તે અહીં જશે, તે કામ કરશે નહીં, તેથી તે અહીં આવશે. અને કોડમાં શું થશે તે છે કે v વધે છે, હવે v ને z માંથી નકલ કરવામાં આવી છે. પ્રશ્ન એ છે કે z ને શું થાય છે? તેથી, આ ચાર નિવેદનો અમલ કર્યા પછી તમે અપેક્ષા રાખશો, કારણ કે ઝેડના મૂલ્યના આ સુધારાને લીધે આ યાદીમાં કોપી કરવામાં આવે છે, જે સ્થાને 2 છે અને તેથી મૂલ્ય 12 ની જગ્યાએ આપણે મૂલ્ય 8 મેળવી શકીએ છીએ જે અમે શરૂ કર્યું છે. બીજી તરફ, જો આપણે આ

સ્ટેટમેન્ટ એક્ઝેક્યુટ કરીએ છીએ, તો પછી આ એક અનિવાર્ય મૂલ્ય છે કારણ કે વિધેયની અંદર વીમાં ફેરફાર એ ઝેડને અસર કરતું નથી. જોકે વી 8 થી 9 સુધી વધારી દેવામાં આવ્યો છે, z એ 8 રહે છે. આ સ્પષ્ટ કરવા માટે છે કે જો આપણે કોઈ પેરામીટરને પસાર કરીએ છીએ, તો પેરામીટર દ્વારા તે મૂલ્ય કે જે પરિવર્તનક્ષમ છે તે ફંક્શનમાં અપડેટ થઈ શકે છે અને આને કેટલીક વખત આડઅસર કહેવામાં આવે છે. તેથી કાર્ય અન્ય પ્રોગ્રામમાં મૂલ્યને અસર કરે છે, તેથી તેને આડઅસર કહેવામાં આવે છે. જો મૂલ્ય અસ્પષ્ટ છે તો એક બાજુ અસર થઈ શકે છે, પરંતુ જો મૂલ્ય અસંચિત છે તો મૂલ્ય બદલાતું નથી, અમે કોઈ પ્રોગ્રામમાં શું કરીએ છીએ તે બદલતું નથી. હવે, આ ફંક્શનની નોંધ લેવા માટે અન્ય બિંદુઓ પણ છે; એક એ છે કે આપણી પાસે અહીં બે રીટર્ન સ્ટેટમેન્ટ છે. આ વિચાર એ છે કે તેઓ કોલિંગ ફંક્શનને સૂચવે છે કે નહીં તે અપડેટ સફળ થયું કે નહીં. તેથી આદર્શ રીતે તમારે એવું કહ્યું હોવું જોઈએ કે પરિણામનું પરિણામ બરાબર છે, અને પછી સુધારા પછી તપાસો કે જ્યાં પરિણામ સાચું છે અથવા ખોટું છે. યાદ રાખો કે અપડેટ સૂચિને અપડેટ કરશે અથવા ઇન્ડેક્સ માન્ય છે કે નહીં તેના આધારે સૂચિને અપડેટ કરવામાં આવશે નહીં અને તે સફળ થાય છે કે નહીં તે આધારે તે સાચી અથવા ખોટા આપશે. તેથી, જે કંઈપણ પરત કરવામાં આવે છે તેના મૂલ્યનું પરિક્ષણ કરીને અમે ચકાસી શકીએ કે શું સુધારા માટે અમે ઇરાદો કર્યો છે કે નહીં. આ એવી વસ્તુ છે જે અમે અપેક્ષા રાખીએ છીએ પરંતુ અમે તે કર્યું નથી, તેથી આ માત્ર એટલું જ સમજવું જ રહ્યું છે કે વળતર મૂલ્ય હોઈ શકે છે પરંતુ તે વિચાર હોઈ શકે છે કે વિધેય વાસ્તવમાં કેટલાક ફેરફારયોગ્ય મૂલ્યને અપડેટ કરશે જેથી અમને તેની પર કોઈ અસર થતી નથી. કાર્ય કાર્ય અંદર કરવામાં આવે છે જો વળતર મૂલ્ય હોય તો પણ તમે તેનો ઉપયોગ કરવા માટે બંધાયેલા નથી, તમે એક વિધેયને એક અલગ વિધાન તરીકે બોલાવી શકો છો કારણ કે અમે અહીં કર્યું છે, તેને અસાઇનમેન્ટનો ભાગ હોવો જરૂરી નથી. બીજી વસ્તુ એ છે કે આને લીધે વિધેયો હોઈ શકે છે જે કંઈ પણ ઉપયોગી નથી. એક લાક્ષણિક ઉદાહરણ એક વિધેય હશે જે ફક્ત એક સંદેશ દર્શાવે છે જેમ કે ભૂલ હતી અથવા તમારા કોડ શું કરી રહ્યું છે તે સમજવા માટે તે કેટલીક અન્ય સૂચક વસ્તુઓ દર્શાવે છે. હવે આવા કાર્ય જેમ જકંઈક દર્શાવવા માટે, તેને કંઈ ગણતરી કે પરત કરવાની જરૂર નથી. તેથી, કોઈ રીટર્ન ફંક્શન હોઈ શકતું નથી. તેથી, મૂળભૂત રીતે શું થાય છે તે છે કે ફંક્શન જ્યારે બધું સામેલ હોય ત્યારે ટોચથી તળિયે બાકી રહેલું બધું જ ચલાવે છે. અને હવે જો તમને તે સમયે રીટર્ન સ્ટેટમેન્ટ મળે તો વિધેય એક્ઝેક્યુશન અટકે છે અને તમે પાછા જાઓ છો. બીજી બાજુ જો તમે નિવેદનો ચલાવવા માટે ચલાવો છો, જો તમે છેલ્લા સ્ટેટમેન્ટ પર પહોંચો છો તો ત્યાં વધુ કંઈ નથી અને પછી કાર્ય સમાપ્ત થશે. વાસ્તવમાં વળતરનું વિધાન હોય તેવું કાર્ય માટે કોઈ જવાબદારી નથી. તેથી, વળતરનું નિવેદન ઉપયોગી છે જો વિધેય મૂલ્યની ગણતરી કરે છે અને તમને અમુક પરિણામ આપે છે જે તમે પાછળથી ઉપયોગ કરશો, પરંતુ તમારી પાસે એવા ફંક્શનો હોઈ શકે છે કે જેની પાસે વળતર મૂલ્ય નથી, તે કિસ્સામાં તમે અમુક ખાલી વસ્તુ પરત કરી શકો છો અથવા તમે કશું પાછું લાવી શકે છે અને બધું બરાબર કામ કરશે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 10:07)

પાયથોન માં ફંક્શનો વિશે નોંધવું એ બીજો મુદ્દો એ છે કે ફંક્શનમાંના નામો વિધેયની બહાર નામોથી છૂટા છે. તો ચાલો ફરી એક રમકડું ઉદાહરણ તરીકે જોઈએ જે કરવું ઉપયોગી નથી. અમારી પાસે ફંક્શન છે જે અમે મૂર્ખ કહીએ છીએ જે અનિવાર્યપણે લે છે દલીલ લે છે અને તે પાછું આપે છે, તેથી તે કંઈ કરતું નથી. પરંતુ તે શું કરે છે તેની વચ્ચે કોઈ સારા કારણ માટે જ નામ એન ને 17 હોય છે. હવે ધારો કે અમારી પાસે અમારા પ્રોગ્રામમાં બહાર છે, એક નિવેદન જે નામ 7 ને મુલ્ય 7 આપેલ છે અને પછી આપણે આ ફંક્શનને કોલ કરીએ છીએ. હવે દેખીતી રીતે, જો આપણે 8 ની મૂર્ખ કહીએ તો વી પણ ઇનપુટ હશે, તેથી v 28 થશે. પ્રશ્ન એ છે કે જ્યારે વી 28 ની હકીકતને અમલમાં મૂકે છે; વિધેય આંતરિક રીતે n એ 17 બરાબર સુયોજિત કરે છે. પ્રશ્ન એ છે કે, આપણે 7 થી 7 ને પૂછ્યું છે, તો આપણે કહીએ કે આ ફંક્શન n એ કાર્યરત 17 ની અંદર 17 હવે છે કે નહીં. તેથી જવાબ એ છે કે n હજુ 7 છે અને તે એટલા માટે છે કે n અંદર અને બહાર n એ n ની બે અલગ અલગ નકલો છે. તેથી, કોઈ પણ નામ જે વિધેયની બાજુમાં વપરાય છે તે બહારના નામથી અલગ હોવાનું માનવામાં આવે છે. બહારના નામો અંદર

દેખાતા નથી, અંદરનું નામ બહાર દૃશ્યમાન નથી. હવે આ એવી કોઈ વસ્તુ નથી જે તમે સામાન્ય રીતે કરો કારણ કે તે ફક્ત ગૂંચવણમાં જ છે જો તમે અંદર અને બહાર સમાન નામનો ઉપયોગ કરો છો, પરંતુ ક્યારેક આ અલગતા માટે ઉપયોગી છે કારણ કે ઘણી વખત આપણે સામાન્ય વસ્તુઓનો ઉપયોગ કરીએ છીએ જેમ કે *ijk* સૂચિની જેમ તમે જાણો છો કે રેન્જ અને તે જેવી વસ્તુઓ. અને તે ઉપદ્રવ બનશે જો આપણે તેનો ઉપયોગ કરવો, યાદ રાખવું અને હું બહાર અને જ અંદર ઉપયોગ કરવો અને ખાતરી કરો કે તેઓ ક્રિયાપ્રતિક્રિયા કરતા નથી. કારણ કે તેઓ કોઈપણ રીતે ક્રિયાપ્રતિક્રિયા કરતા નથી કારણ કે અમે ગમે તે સ્થળે મુક્તપણે ઉપયોગ કરી શકીએ છીએ અને આ હકીકત વિશે ચિંતા ન કરીએ છીએ કે અમે પહેલેથી જ કેવી રીતે કોલિંગ ફંક્શનમાં *I* અથવા *J* છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 12:07)

અમે આગળ ઉલ્લેખ કર્યો છે તે એક બાબત એ હતી કે કાર્યને અમલમાં મુકવા પહેલાં વ્યાખ્યાયિત થવું જોઈએ. હવે આ થોડું સૂક્ષ્મ બિંદુ છે, તો ચાલો આપણે તેને થોડી વધુ જોઈએ. યાદ રાખો કે દુભાષિયો દ્વારા પાયથોન પ્રોગ્રામ ઉપરથી નીચે સુધી વાંચવામાં આવે છે. તેથી, જ્યારે પાયથોન પ્રોગ્રામ વાંચવામાં આવે છે ત્યારે તે એક ની વ્યાખ્યા વાંચે છે, પરંતુ તેને અમલમાં મૂકતું નથી, અને નોંધ્યું છે કે એક ની આ વ્યાખ્યા *G* માટે એક અભ્યર્થ છે જે વાસ્તવમાં પછીથી છે. પરંતુ બિંદુ એ છે કે જ્યારે *fig* ની વ્યાખ્યા વાંચતી નથી ત્યારે તે ફક્ત યાદ છે કે આ વિધાન જે કૌંસમાં હોવું જોઈએ, માત્ર સુસંગત હોવું જોઈએ. તેથી, જો હું એક ને કોલ કરું તો આ નિવેદનની ગણતરી કરવી જોઈએ જેથી તે ફક્ત એક ને વ્યાખ્યાયિત કરતી નથી. તેથી, હું એક વ્યાખ્યાયિત કરું છું, પછી હું જી વ્યાખ્યાયિત કરું છું, છેવટે જ્યારે હું આ નિવેદનમાં આવું છું ત્યારે તે કહે છે 77 ની એક છે. તો 77 ની એક અહીં આવશે અને અમે 77 અહીં આવશે. અને તે કહેશે 78 ની જી એ કંઈ પણ 81 નથી, તેથી તે અહીં આવશે. તેથી 81. અને પછી અંતે મને 81 મળશે. તેથી, તે માત્ર ત્યારે જ થાય છે જ્યારે હું નિવેદન એક ચલાવવા માટે તે સમયે જી એક્ઝિક્યુટ થઈ ગયો છે. તેમ છતાં અમે કહીએ છીએ કે ફંક્શનનો ઉપયોગ થતાં પહેલાં તે વ્યાખ્યાયિત થવો જોઈએ, તે હકીકતને નકારી શકતું નથી કે એક ફંક્શન તે ફંક્શનને કોલ કરી શકે છે જે તેના પછી વ્યાખ્યાયિત કરવામાં આવે છે, જો કે તે વ્યાખ્યા પછી જ તમે આ ફંક્શનનો ઉપયોગ કરો છો. તેથી આ ક્રમ દંડ છે. ધારો કે, આપણે આ ક્રમને બીજી રીતમાં ફરીથી લખ્યા છે, તેથી ધારણા કે અમારી પાસે એક ની વ્યાખ્યા હતી તો આ નિવેદન હતું. અમે મૂળભૂત રીતે આ બે નિવેદનો લે છે હવે શું થાય છે કે જ્યારે પાયથોન ઇન્ટરપ્રીટર આ બિંદુ પર આ રેખા નીચે આવે છે ત્યારે તે પ્રયત્ન કરશે અને *f* ને ફોન કરશે, તેથી *f* પ્રયત્ન કરશે અને *g* ને ફોન કરશે અને જી સારી રીતે કહેશે મારી પાસે હજી સુધી કોઈ વ્યાખ્યા નથી, કારણ કે હું હજી ગયો નથી આ વિધાન ભૂતકાળ તેથી જો હું આ સ્ટેટમેન્ટ મૂકું છું, તો હું જી વ્યાખ્યાયિત કરું છું અને *f* માટે *g* ની જરૂર છે તો આ સ્ટેટમેન્ટ એક ભૂલ બનાવશે જ્યારે આ નિવેદન નહીં થાય. તે ખરેખર ઉપયોગી છે જો આપણે બધા ફંક્શનો વ્યાખ્યાયિત કરીએ છીએ કારણ કે વિધેયો વચ્ચેની આંતર નિર્ભરતાને દુભાષિયા દ્વારા યોગ્ય રીતે ઉકેલવામાં આવશે અને અમને તેના વિશે ચિંતા કરવાની જરૂર નથી. જ્યારે આપણે વિધેયો અને નિવેદનો મિશ્રણમાં કરીએ છીએ ત્યારે આપણે સાવચેત રહેવું જોઈએ કે કાર્યો પાછળની વસ્તુઓનો સંદર્ભ આપતા નથી કે જે દુભાષિયા દ્વારા હજી સુધી સ્કેન કરવામાં આવ્યો નથી. શરૂઆતમાં તમારી બધી કાર્યપદ્ધતિઓ વ્યાખ્યાયિત કરવાની આ એક વધુ કારણ છે અને માત્ર પછી તે નિવેદનો છે જે તમે અમલ કરવા માંગો છો.

(સ્લાઈડ ટાઇમ નો ઉલ્લેખ કરો: 14:40)

એક અંતિમ મુદ્દો એ છે કે જ્યારે આપણે પ્રોગ્રામિંગમાં આગળ વધીએ છીએ ત્યારે વધુ રસપ્રદ ઉદાહરણો દ્વારા આપણે પાછળથી પાછા જઈશું. તે એ છે કે એક ફંક્શન પોતે જ સારી રીતે કોલ કરી શકે છે. આ પ્રકારની સૌથી પ્રયોગાત્મક કાર્ય, જેને રિકર્સિવ ફંક્શન્સ કહેવાય છે. કાર્યો જે પોતાને પર આધાર રાખે છે કારણદર્શી કાર્ય છે. જો તમને યાદ કરાય છે કે ફેક્ટોરિયલને *n* માં *n* ઓછા 1 માં *n* ની 2 ની *n* માં 2 ની *n* માં વ્યાખ્યાયિત કરવામાં આવે છે. તેથી તમે *n* નો કરો અને તેને પોતાની સંખ્યા કરતા નાની સંખ્યામાં

ગણી શકો છો અને વ્યાખ્યા દ્વારા 0 ફેક્ટોરિયલને વ્યાખ્યાયિત કરવામાં આવે છે. 1. આ વ્યાખ્યામાં આપણે જે અવલોકન કરીએ છીએ તે એ છે કે, $n-1$ થી 1 નો આ ભાગ વાસ્તવમાં n બાદ 1 ફેક્ટોરિયલ છે. અન્ય શબ્દોમાં n ફેક્ટોરિયલને નાના ફેક્ટોરિયલની દ્રષ્ટિએ વ્યાખ્યાયિત કરી શકાય છે, તે n વખત n ઓછા 1 ફેક્ટોરિયલ છે, તેથી તે આ કાર્ય શોષણ કરે છે. 0 નો આધારભૂત પરિબલ 1 છે અને ત્યારથી નકારાત્મક સંખ્યાના કારણદર્શી વ્યાખ્યાયિત નથી અને અમે સુરક્ષિત રહેવા માંગીએ છીએ, જો આપણે કહી શકીએ કે જો n બરાબર 0 અથવા n બરાબર છે તો 0 ની બરાબર આપણે 1 પરત કરીશું. તેથી આ અમે સામાન્ય રીતે બેઝ કેસને કોલ કરીએ છીએ. આ કિસ્સામાં ફેક્ટોરિયલ સંપૂર્ણપણે આગળ કોઈ કાર્ય કર્યા વગર વ્યાખ્યાયિત થયેલ છે. હવે જો n બરાબર 0 ની બરાબર ન હોય તો n એ 0 કરતા મોટો છે. જો n એ 0 કરતા મોટો છે તો આપણે વર્તમાન સંખ્યા લઈએ છીએ અને આપણે તે નાના ફેક્ટોરિયલ દ્વારા વધારીએ છીએ જે બરાબર ઉપર દર્શાવેલ વ્યાખ્યા છે. તેથી જો હું 3 નો ફેક્ટોરિયલ કહું, તો આ 3 ગુણ્યા 3 વખત પરિણમશે, જેથી તે આ ફંક્શનને ફરીથી બોલાવશે અને આ મને 2 વખત 2 થી ફેક્ટોરિયલ આપશે અને આ રીતે. 1 નો ફેક્ટોરિયલ મને 1 વખત ફેક્ટોરિયલ આપશે અને બિંદુ એ છે કે ફેક્ટોરિયલ 0 હવે સમાપ્ત થશે અને તે મને 1 આપશે, કારણ કે તે કહે છે કે દલીલ 0 રીટર્ન 1 થી ઓછી અથવા બરાબર છે. આ 1 હવે પાછો આવશે. અને અહીં ગુણાકાર કરો, એટલે તમને 1 વાર મળે 1, તો 1 ગુણ્યા 1 અહીં આવશે અને અહીં આવશે, તો પછી આ 2 ને પાછો લાવશે અને પછી 3 ગુણ્યા 2 આ બનશે 6. આ રીતે કાર્ય આપણે ચલાવશે આ વિશે વધુ પછીથી ચર્ચા કરો, પરંતુ માત્ર તે સમજાવવા માટે કે કાર્યો પોતાને સારી રીતે કોલ કરી શકે છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 17:03)

સારાંશ માટે, કાર્યો તમારા કોડને લોજીકલ હિસ્સામાં ગોઠવવાનો સારો માર્ગ છે. તેથી જો તમારી ગણતરીના એકમ છે જે વારંવાર કરવામાં આવે છે અને ઘણી વખત વિવિધ શક્ય પ્રારંભિક મૂલ્યો સાથે કરવામાં આવે છે તો તમારે તેને એક કાર્યમાં એકસાથે ખસેડવું જોઈએ. જો તમે તમારા કોડને નાના કાર્યોમાં વિભાજીત કરો છો, તો તેને સમજવું, વાંચવા અને જાળવવાનું ખૂબ સરળ છે. જ્યારે આપણે ફંક્શનમાં દલીલો પસાર કરીએ છીએ ત્યારે તે બરાબર એક નામના મૂલ્ય અસાધન કરવા જેવું છે. તેથી, પસાર થનારા મૂલ્યો ફંક્શનમાં માત્ર સુધારિત થઈ શકે છે જો તે પરિવર્તનક્ષમ હોય, જો તેઓ નિર્વિવાદ હોય તો કોઈ કાર્યમાં કોઈ ફેરફાર થતો દલીલ બહાર નથી કરતું. પણ જો આપણે કોઈ કાર્યની અંદર જ નામનો ઉપયોગ કરીએ છીએ, જેમ કે ફંક્શનની બહાર જોવા મળે છે તો ફંક્શનમાંનું નામ કોઈ પણ રીતે બહારના નામ પર અસર કરતું નથી. તેથી, કાર્યોને આપણે શું કહીએ છીએ તેની સ્થાનિક કલ્પના છે. એક નામનું અવકાશ છે જ્યાં એક નામ સમજી શકાય છે, તેથી કાર્યમાંનું નામ બહારથી અસ્તિત્વમાં નથી અને ઊલટું. પણ વિધેયોને તેનો ઉપયોગ કરવામાં આવે તે પહેલાં વ્યાખ્યાયિત થવો જોઈએ અને તમારા પ્રોગ્રામની શરૂઆતમાં તમારી બધી ફંક્શનની વ્યાખ્યાઓ દબાણ કરવા માટે આ એક સાફ કારણ છે, જેથી વાસ્તવમાં બોલાવવામાં આવે તે પહેલાં પાયથોન ઈન્ટરપ્રીટર તેમને બધાને ડાયજેસ્ટ કરશે. તેથી જો ત્યાં પરસ્પર નિર્ભરતા હોય તો આપણી સમસ્યા નથી. છેલ્લે, આપણે જોયું કે આપણે રસપ્રદ કાર્યો લખી શકીએ છીએ જે પોતાને કહે છે અને આવનારા અઠવાડિયામાં આપણે આનાં ઘણા બધા ઉદાહરણો જોશું.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રોફેસર. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
અઠવાડિયું - 02 લેક્ચર - 06

આ અઠવાડિયાની આસપાસના ઉદાહરણો, ચાલો આપણે કેટલાક ઉદાહરણો પર વિચાર કરીએ જે કેટલાક વિભાવનાઓને સમજાવે છે. અત્યાર સુધી જોયું છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 00:11)

આપણે પહેલાથી જ એક કાર્ય જોયું છે જે સંખ્યા n ના પરિબલોને ગણતરી કરે છે. તેથી, અમે જોયું કે તમામ પરિબલો 1 અને n વચ્ચે આવેલા હોવા જોઈએ. આ કંઈક છે જે આપણે લૂપ માટે કુદરતી રીતે ગણતરી કરી શકીએ છીએ. અમે નીચે પ્રમાણે n ના પરિબલો વ્યાખ્યાયિત કરે છે; અમે માનીએ છીએ કે પરિબલોની સૂચિ ખાલી છે, અને રેંજ 1 થી n માં દરેક નંબર માટે જો તે સંખ્યા વિભાજક છે, તો તે n નો પરિબલ છે અને તે આપણે પરિબલોની સૂચિમાં જોડીએ છીએ અને આખરે અમે આ સૂચિને પરત કરીએ છીએ. આ એક સરળ કાર્ય છે જે ફક્ત પરિબલોની સૂચિ લે છે, તે ઇનપુટ n ના પરિબલોની સૂચિ આપે છે.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 00:47)

એક પ્રાથમ નંબર એ એક છે જે 1 અને તેના બદલે અન્ય કોઈ અન્ય સંખ્યા દ્વારા વિભાજીત છે. બીજા શબ્દોમાં કહીએ તો, n નો એકમાત્ર પરિબલો 1 અને n હોવો જોઈએ, જો n એ મુખ્ય છે તેથી, ઉદાહરણ તરીકે, 17, જે એક અપૂર્ણ સંખ્યા છે, તેમાં ફક્ત બે પરિબલો 1 અને 17 છે. જ્યારે 18 જે કોઈ મુખ્ય નથી તે ઘણી વધારે પરિબલો છે, તે 2 ગુણ્યા 9 અને 3 ગુણ્યા 6 છે. તેથી 18 ના પરિબલોની સૂચિ એક લાંબી યાદી છે અને ફક્ત 1 અલ્પવિરામ 18 છે. આ આપણને પહેલેથી જ જોવામાં આવેલા પરિબલો પર આધારિત પ્રાથમની ખૂબ જ સરળ વ્યાખ્યા લખવાની મંજૂરી આપે છે. સંખ્યા મુખ્ય છે, જો પરિબલોની સૂચિ બરાબર છે 1 અલ્પવિરામ n . તો આપણે શું કરીએ તે આપણે કાર્ય પરિબલોને બોલાવીએ છીએ અને તપાસ કરીએ છીએ કે તે શું આપે છે અને જુઓ કે તે સૂચિ 1 કોમા n એ બરાબર છે. આ હકીકતનું બીજું દૃષ્ટાંત એ છે કે જો આપણે આપણા કોડને વિધેયમાં વિભાજીત કરીએ તો આપણે કાર્યોને અન્યની અંદર વાપરી શકીએ છીએ અને આપણી સમસ્યાને નાના એકમોમાં વિભાજીત કરી શકીએ છીએ જે પચાવી શકાય તેવું સરળ છે અને સમજવા માટે. અહીં, અમે કહ્યું હતું કે મુખ્ય સંખ્યામાં ફક્ત બે પરિબલો 1 અને પોતે જ છે. અમે અલગથી પરિબલોની સૂચિની ગણતરી કરવા માટે એક માર્ગ લખ્યો છે, તેથી અમે તે સૂચિ લઈ શકીએ છીએ અને આપેલ નંબર મુખ્ય છે કે નહીં તે તપાસ કરી શકીએ છીએ. જ્યારે આપણે મુખ્ય નંબરો સાથે વ્યવહાર કરી રહ્યા છીએ ત્યારે એ જાણવું એક નાની બાબત છે કે આપણે આકસ્મિક રીતે 1 ને પ્રાથમ તરીકે વ્યાખ્યાયિત ન કરવો જોઈએ, કારણ કે જો તમે ફક્ત આ વ્યાખ્યાને જોશો કે ફક્ત એક જ પરિબલો 1 અને પોતે છે, તે થોડી અસ્પષ્ટ છે કારણ કે 1 એક પરિબલ છે અને પોતે 1 પણ એક પરિબલ છે. અમે નિખાલસપણે આને એમ કહીને વાંચી શકીએ છીએ કે 1 એક મુખ્ય છે, પરંતુ પરંપરા 1 દ્વારા કોઈ મુખ્ય નથી. તેથી, બે પરિબલો 1 અલગથી અને પોતે જ હોવા જોઈએ. સદભાગ્યે આપણે આપણા કાર્ય પરિબલોને કહીએ છીએ તો 1 નાં પરિબલો એક જ યાદી તરીકે પરત કરશે- જેમાં 1 મૂલ્ય ધરાવતી સિંગલન સૂચિ છે, કારણ કે તે કોડથી ચાલશે 1 થી 1 સુધી ચાલશે, તેથી તેને માત્ર એક વાર જ મળશે જ્યારે આ રીટર્ન સ્ટેટમેન્ટ સાચું હોત, તો હું ઈચ્છીએ છીએ કે વાસ્તવિક મૂલ્ય 1 અલ્પવિરામ 1, 1 અલ્પવિરામ નો છે, તેથી n એ 1 છે. તેથી સદભાગ્યે, જે રીતે અમે લખ્યું છે, તે યોગ્ય

રીતે એ 1 એક મુખ્ય નહીં, પરંતુ આ પ્રકારની સરહદ પરિસ્થિતિઓ છે કે જ્યારે કોઈ પાયથોન અથવા અન્ય કોઈપણ પ્રોગ્રામિંગ ભાષામાં વિધેયો લખે ત્યારે તપાસ કરવી જોઈએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 03:09)

જો તમે આપેલા નંબર n ની નીચે રહેલા બધા મુખ્ય નંબરોની સૂચિબદ્ધ કરવા માંગો છો. તેથી, આપણે ફક્ત દરેક સંખ્યાને 1 થી n ની તપાસ કરવી પડશે, પછી ભલે તે મુખ્ય છે કે નહીં. અમે પહેલેથી જ જાણીએ છીએ કે કઈ કોઈ વસ્તુ મુખ્ય છે કે નહીં. ફરી એક વાર, આપણે ફંક્શન `primesup` લખી શકીએ છીએ જે દલીલ n ને લે છે. તેથી, શરૂઆતમાં અમે કહીએ છીએ કે ત્યાં સુધી કોઈ પ્રાઈમ નથી. હવે રેન્જ 1 થી n વત્તા 1 માં તમામ નંબરો માટે, જેનો અર્થ 1, 2 થી n સુધી થાય છે. અમે તપાસ કરીએ છીએ કે તે સંખ્યા હું એક મુખ્ય છે, હવે આ એક કાર્ય છે જે આપણે પહેલેથી લખ્યું છે. જો તે મુખ્ય છે તો આપણે તેને અમારી સૂચિ જોડીએ છીએ, જો આપણે આગામી એક પર ન જઈએ અને છેલ્લે આપણે જોયેલા પ્રાઈમની યાદી પરત કરીશું. ફરી એક વાર, હવે આપણે એ ફંક્શનનો ઉપયોગ કર્યો છે જે આપણે એસપ્રાઈમ પહેલાં લખ્યો છે. ઇસપ્રાઈમ એ ફંક્શન્સ તરીકે ઓળખાતી કાર્યનો ઉપયોગ કરે છે જેને આપણે અહીં જોતા નથી હવે અમારી પાસે ત્રણ સ્તરો છે, `primesupto` જે કોલ્સ `isprime` છે, જે પરિબલોને કોલ કરે છે. આ સરસ લાક્ષણિક રીત છે કે તમે સરસ કાર્યક્રમો લખો જ્યાં તમે તમારા કાર્યને નાના કાર્યોમાં વિભાજિત કરી શકો છો. હવે તમારા કાર્યને નાના કાર્યોમાં તોડી નાખવાનો ફાયદો એ છે કે જો તમે વધુ કાર્યક્ષમતાને ઓપ્ટિમાઇઝ અથવા વધુ કરવા માંગો છો, તો તમે ફંક્શનને અમલ કરવા માટે સૌથી વધુ સ્પષ્ટ અથવા નિષ્કપટ માર્ગ લખી શકો છો જેથી કરીને તમે તપાસી શકો કે તમારું એકંદર કોડ તે કરે છે આમ કરવાથી તમે અલગથી દરેક કાર્યમાં જઈ શકો છો અને પછી વધુ કાર્યક્ષમ બનાવવા માટે તેને અપડેટ અથવા ઓપ્ટિમાઇઝ કરી શકો છો. તેથી, વિધેયોમાં તમારો કોડ તોડીને તમારા કોડને અપડેટ કરવાનું અને તેને જાળવવાનું સરળ બનાવે છે, અને બાકીનાને અસર કર્યા વગર તેના ભાગોને બદલવામાં સરળ બનાવે છે N સુધી પ્રાઈમ્સ, અમે અગાઉથી જાણતા હતા કે અમને બધી સંખ્યાઓ 1 થી n ની તપાસ કરવી પડશે, તેથી અમે તેનો ઉપયોગ કરી શકીએ છીએ. જો આપણે આપણી આવશ્યકતાઓને બદલીને કહીએ કે અમે n સુધી પ્રાઈમ અપ નથી માંગતા, પરંતુ અમે પ્રથમ n primes જોઈએ છીએ હવે, જો આપણે પ્રથમ n primes જોઈએ, તો અમને ખબર નથી કે કેટલા સ્કેન સ્કેન કરે છે. અમે જ્યાં n એ વડા આવશે આવે છે ખબર નથી. જો n એ નાનું છે, તો આપણે તેને જોઈને માત્ર આકૃતિ કરી શકીએ છીએ, પરંતુ જો એ મોટી સંખ્યા છે, જો આપણે પ્રથમ હજાર પ્રાઈમ્સ માટે કહીએ તો તે અંદાજ લગાવવાનું ખૂબ જ મુશ્કેલ છે કે હજારમો મુખ્ય કેટલું મોટું હશે. તેથી, આપણે ત્યાં સુધી જતા રહેવું પડશે જ્યાં સુધી આપણે હજાર પ્રાઈમ શોધતા નથી અને અમને અગાઉથી ખબર નથી. આ અન્ય પ્રકારની લૂપનો ઉપયોગ કરવા માટેનો એક સારો કેસ છે જ્યારે કે તેથી, અમારે શું કરવાની જરૂર છે તે આપણે કેટલા પ્રાઈમ જોઈ છે તેની ગણતરી રાખવી જરૂરી છે. તપાસ કરવા માટે એક જ સમયે તમામ નંબરો પસાર કરવાની જરૂર છે, અને જો આપણે અત્યાર સુધી જોઈ હોય તે તમામ પ્રાઈમની સૂચિની જરૂર છે. માત્ર બીજા બિંદુને સમજાવવા માટે કે જે આપણે જોયું છે તે એકસાથે સોંપણી છે જે કહે છે ઠીક છે, શરૂઆતમાં આપણે 0 પ્રાઈમ જોયું છે, તેથી ગણતરી 0 છે. અમે પ્રથમ નંબર સાથે શરૂ કરીએ છીએ, આ પહેલું નંબર છે કે આપણે તપાસીએ કે તે મુખ્ય છે કે નથી, અને શરૂઆતમાં primes ની સૂચિ ખાલી છે. તેથી આ કહે છે, આ ત્રણ મૂલ્યો લે છે, આ ત્રણ નામો લે અને તેમને આ ત્રણ મૂલ્યો આપો. સોંપો આ સમાન છે, તે જ ગણતરી 0 બરાબર છે; હું 1 બરાબર છે; અને `plist` ખાલી છે. તેથી, આનો ચોક્કસ કાર્ય ચોક્કસ કાર્ય છે. હવે જ્યાં સુધી આપણે n primes ન જોઈ હોય ત્યાં સુધી, જ્યારે સંખ્યા n કરતા ઓછી છે, ત્યારે આપણે તપાસવું પડશે કે વર્તમાનમાં હું એક મુખ્ય છે અને જો તે હોય, તો તેને ઉમેરો. જો પ્રથમ, જો હું જે મૂલ્ય જોઈ રહ્યો છું તે એક મુખ્ય છે, તો સૌ પ્રથમ તો આપણે વધુ એક પ્રાઈમ મળી છે, તેથી આપણે ગણતરીની વેલ્યુ વધારીએ છીએ. અને અમે એક પ્રાઈમ મળી છે, તેથી આપણે તેને primes ની સૂચિમાં ઉમેરીએ. જો આ કોઈ મુખ્ય નથી, તો આપણે આગલા નંબર પર જવું જોઈએ; કોઈ પણ સંજોગોમાં, આપણે આગામી નંબર પર જવાની જરૂર છે અને જ્યાં સુધી અમે n નો ગણતરી નહીં કરીએ. તો, જો આપણે બહાર નીકળીએ, તો આ બિનશરતી છે, આપણે હંમેશા અપડેટ કરીએ છીએ. તેથી, દરેક

માટે હું પહેલી વખત તપાસ કરું છું જો તે મુખ્ય છે, જો તે મુખ્ય છે તો અમે ગણતરીને અપડેટ કરી શકીએ છીએ અને અમે મને પ્લાસ્ટમાં જોડીએ છીએ. અને તે અવિભાજ્ય છે કે નહીં, અમે વધારો કરીએ છીએ; અને દરેક વખતે, અમે ગણતરીમાં વધારો કરીએ છીએ અમે આ તરફ આગળ વધારી રહ્યા છીએ. યાદ રાખો કે આપણી નોકરી છે તે સુનિશ્ચિત કરવા માટે કે જ્યારે આખરે બહાર આવશે ત્યારે આ શરત ખોટી બની જશે. જ્યારે આપણે એક મુખ્ય સંખ્યા જોશું, ત્યારે ગણતરી 1 વત્તા બનશે. આપણે 0 બરાબર ગણતા થી શરૂ કરીએ છીએ, તેથી છેવટે તે n એ પાર કરી રહ્યું છે. અલબત્ત, આપણે આ હકીકતનો ઉપયોગ કરી રહ્યા છીએ, હકીકતમાં આપણે જાણીએ છીએ કે અનંત સંખ્યાના પ્રાઈમ છે. અમે કોઈપણ n માટે હંમેશા હતા, અમે પ્રથમ n primes શોધવા માટે સક્ષમ. તેથી, જ્યારે આપણે n primes શોધીએ છીએ, જ્યારે આપણે 0 થી n ની 1 ઘાત ચાલ્યા ગયા છીએ, અને અમે ગણતરી n સુધી પહોંચી ગયા છીએ, ત્યારે આપણે એ પ્રાઈમ જોયું છે, તો આપણે તે સૂચિ પરત કરીએ છીએ જે આપણે અત્યાર સુધી મળી છે અને આ પ્લાસ્ટ છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 07:31)

આ બે ઉદાહરણો, n અને n primes સુધીના primes એ અને જ્યારે માટે ઉપયોગો વચ્ચેનો તફાવત સમજાવે છે. N સુધીના સમય સુધી, અમે જાણીએ છીએ કે અમે બધી સંખ્યાઓ 1 થી n સુધી સ્કેન કરવી જોઈએ, તેથી for ; નો ઉપયોગ કરવો સરળ છે. N primes માં, આપણે કેટલા પ્રાઈમ નથી જાણતા, કેટલાંક સંખ્યામાં આપણને n primes શોધવા માટે સ્કેન કરાવવું પડે છે, તેથી અમે થોડા સમયનો ઉપયોગ કરીએ છીએ અને અમે ગણતરી કરીએ છીએ અને સંખ્યાને પાર કરવા માટે અમે ગણતરી કરીએ છીએ જે આપણે શોધી રહ્યા છીએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 07:58)

તેથી, તે તારણ આપે છે કે તમને વાસ્તવમાં માટે ; જરૂર નથી, તમે હંમેશાં માટે ; અનુકરણ કરી શકો છો. ચાલો આપણે બે ખાસ પ્રકારો જોઈએ છીએ જેમાં આપણે લખીએ છીએ. પ્રથમ રસ્તો આ શ્રેણી માટે છે, તેથી અમે શ્રેણી i અને j માં n માટે કહીએ છીએ. અમે હું શરૂ કરું છું અને આપણે અનુક્રમથી i , i વત્તા 1 અપ જ મા ઓછા 1; અને n ની દરેક વેલ્યુ માટે, આપણે આ સ્ટેટમેન્ટ એક્ઝિક્યુટ કરીએ છીએ અથવા અહીં વધુ સ્ટેટમેન્ટ્સ હોઈ શકે છે. આ લૂપની અંદરનો એક બ્લોક છે જે અમે એક્ઝિક્યુટ કરીશું. જો તમે આના માટે વિના કરવા માંગો છો, તો અમે નીચે પ્રમાણે આ પ્રમાણે કરી શકીએ છીએ. તેથી અમે આરંભ, તેથી n ની પ્રથમ કિંમત સાથે કહે છે i છે; અને જ્યારે n એ પાર નથી કરતો, તો તમે આ નિવેદન ચલાવો - તે જ નિવેદન, અને તમે એન નો વધારો કરો. તેથી, તે અહીં પાછો આવે છે અને તમે તપાસો છો કે હું તમને વત્તા 1, હું વત્તા 2 મળે છે, અને પછી જ્યારે તમે જમાસ 1 પર પહોંચો છો, તો આગળના સમયે તે j હશે અને તે બહાર નીકળી જશે. અમારી પાસે માટે ; માં એક શ્રેણી છે, અમે ફક્ત એક કાઉન્ટર સેટ કરી શકીએ છીએ અને કાઉન્ટરને મેન્યુઅલી વધારીએ છીએ અને તે વખતે ઉપલા બાંધો સામે કાઉન્ટર મૂલ્યને તપાસ કરી શકીએ છીએ. અમે જે રીતે ઉપયોગ કરીએ છીએ તે એક યાદીના તત્વો દ્વારા ફરી વળવું છે. તેથી, n હવે જો અલ $x_1, x_2, \dots, x_k$ સુધીના મૂલ્યોની સૂચિ છે, n આ સૂચિમાં દરેક મૂલ્ય લેશે. અહીં, આપણે હવે દરેક સ્થિતિ પરની સૂચિ અને પિકઅપની સૂચિમાંથી ચાલવા માટે સ્થિતિ ગોઠવી શકીએ છીએ. અમે કહીએ છીએ કે અમે 0 મા સ્થાનેથી શરૂ કરીએ છીએ; અને જ્યાં સુધી સ્થિતિની દ્રષ્ટિએ સૂચિના અંત સુધી પહોંચતા ન હોય ત્યાં સુધી, આપણે n એ સ્થાને મૂલ્ય તરીકે સુયોજિત કરીએ છીએ અને પછી અમે આ નિવેદન ચલાવો અને આપણી સ્થિતિને વધારીએ છીએ. તેથી, માટે ; બંને સ્વરૂપો જ્યારે લખી શકાય છે, પરંતુ નોંધવું છે કે જમણા હાથ બાજુની બાજુ કરતા નોંધપાત્ર રીતે વધુ નીચ અને જટીલ છે.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 09:40)

જ્યારે આપણે તેનો ઉપયોગ કરી શકીએ છીએ, જ્યારે 'સ્ટેટમેન્ટ માટે' અનુમતિ આપતી વખતે સ્ટેટમેન્ટ એ 'ફોર' નો ઉપયોગ કરવા માટે વધુ સારું છે, જ્યાં તે પ્રાકૃતિક છે જ્યાં આપણે અગાઉથી જાણીએ છીએ કે આપણે શું પુનરાવર્તન કરીએ છીએ ઓવર આ વધુ વાંચનીય કોડ માટે બનાવે છે. અને સામાન્ય રીતે વધુ વાંચનીય કોડ વધુ સારા પ્રોગ્રામ માટે બનાવે છે. તો, એક સારા કાર્યક્રમ લખવા માટે તમારે શું કરવાની જરૂર છે? વેલ સૌ પ્રથમ તો તમારે તે કરવું જોઈએ તેવું જ કરવું જોઈએ, તેથી તે યોગ્ય હોવું જોઈએ. અને બીજું, તે શક્ય તેટલી કાર્યક્ષમ રીતે કરવું જોઈએ; અમે અત્યાર સુધી કાર્યક્ષમતામાં નથી જોયા છે, અમે આ બાબતે વધુ ધ્યાન આપીશું. પરંતુ અમે વસ્તુઓને ઝડપથી કાર્ય કરવા માંગીએ છીએ જે આપણે કાર્યક્ષમ રીતે સૌથી વધુ કરવા માંગીએ છીએ; અને આ તે છે જ્યાં અલ્ગોરિધમ આવે છે, તમે કેવી રીતે કંઈક કરો છો તે અલ્ગોરિધમનો તમને શું કહેશે. અને બીજો ભાગ એ છે કે તમારે તમારી સૂચનાઓને એવી રીતે લખવી જોઈએ કે જે તમારા માટે યોગ્ય છે તે માન્ય છે, અને કોઈ બીજાને સમજવા માટે અને જો જરૂરી હોય તો અપડેટ કરો. હવે અમે જાળવણી આ હકીકત પર ભાર મૂકે છે કે તમે આજે કંઈક લખી ન જોઈએ, જે ચોક્કસ કાર્ય કરે છે ઘણીવાર કોઈકને પાછળથી પછી ક્યાં તો તે વધુ કાર્યક્ષમ બનાવવા અથવા કાર્યક્ષમતામાં ફેરફાર કરે છે તમારા કાર્યક્રમ કરે છે તે વસ્તુઓ શ્રેણી વધારો અને તે કિસ્સામાં ખૂબ જ વારંવાર, તે વ્યક્તિ જે કોડ લખે છે પરંતુ કોઈ અન્ય વ્યક્તિને કોડ સમજવા અને અપડેટ કરવાની નોકરી હોય તે નથી. તેથી, દરેક તબક્કે જો કોઈ વ્યક્તિ વ્યક્તિથી ફેરફાર કરવાનું શરૂ કરી દે છે, જે કોડ લખે છે તે શરૂઆતમાં તે સ્પષ્ટ અને વાંચનીય શૈલીમાં લખે છે તો તે કોડને મજબૂત રીતે જાળવી રાખવામાં સરળ બનાવે છે કારણ કે તે બદલાય છે. સારાંશ માટે, પ્રોગ્રામિંગના બે ભાગો છે; પ્રથમ એ છે કે તમે શું કહેવા માગો છો કે જે અલ્ગોરિધમનો છે, બીજા ભાગમાં તમે કેવી રીતે કહી શકો છો કે શૈલી શું છે. તેથી, દરેક પ્રોગ્રામિંગ ભાષા માં સ્ટાઇલનો ઉપયોગ તમે સૌથી વધુ અસરકારક અને વાંચનીય કોડ બનાવવા માટે કરી શકો છો; જો તમારી પાસે 'માટે' હોય તો - તેનો ઉપયોગ કરો, જેમ કે મેં થોડો સમય ઉપયોગ કરવાનું કહ્યું અને તેથી આગળ વધવું નહીં.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન,
પ્રોફેસર માધવન મુકુન્દ,
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 03 લેકચર - 01 રેજ વિશે વધુ

(સ્લાઇડનો સંદર્ભ લો: 00:02)

આપણે રેજ ફંક્શન જોયું છે જે મૂલ્યોનો ક્રમ પેદા કરે છે. સામાન્ય રીતે, જો આપણે રેન્જ આઇ કોમા જ લખીએ છીએ તો, હું ક્રમ, i વત્તા 1 અપ એન્ડ જ ઓછા 1 મળે છે. ઘણી વાર આપણે 0 થી શરૂ કરવા ઇચ્છીએ છીએ. તેથી, જો આપણે ફક્ત એક જ દલીલ આપીએ તો જો આપણે શ્રેણી લખીએ, આ ઉપલી બાઉન્ડ અને નીચલા બાઉન્ડ 0 ની જેમ જોવામાં આવે છે. આ એક સ્લાઇડની જેમ છે, જ્યાં તમે સ્લાઇડનો પહેલો દલીલ ન લખો, જો આપણે દાખલા તરીકે લખીએ, તો તે 0 થી n સુધી ચાલશે. ઓછા 1. જો આપણે ફક્ત શ્રેણી જ લખીશું તો, આપમેળે 0, 1 થી ઓછો થશે. 1 ઘણીવાર આપણે ક્રમ બનાવવાની ઇચ્છા રાખી શકીએ છીએ જ્યાં આપણે 1 કરતાં અન્ય મૂલ્યથી અવગણીએ છીએ. આપણે એક પ્રકારનું અંકગણિત પ્રગતિ જો તમે તેનાથી પરિચિત છો. તેથી, અમે, હું વત્તા કે, હું વત્તા 2 k અને તેથી વધુ, અમે ત્રીજા દલીલ આપીને આ કરવા માંગો છો. ત્રીજા દલીલ જો આપણે તેને રેન્જમાં આપીએ તો રેન્જ ફંક્શનને દરેક કી વસ્તુને અવગણવા માટે કહે છે. તેથી, આપણી પાસે તો હું સીધું i વત્તા k પર જઈશ. તેથી, અસ્પષ્ટ રીતે જો આપણે કંઈપણ ન બોલીએ તો એવું જ છે કે આપણે અહીં 1 ને સાચવી. તેથી, મૂળભૂત મૂલ્ય 1 i , i વત્તા 1 અને તેથી વધુ છે. હવે, આ કેવી રીતે જાય છે? ઠીક છે, આપણે ત્યાં સુધી જવું છે જ્યાં સુધી આપણે સામાન્ય રીતે જ ઓછા 1 સુધી પહોંચતા નથી. સામાન્ય રીતે, આપણે j તો, આપણે શું કરીશું, આપણે સૌથી મોટી એન માટે હું વત્તા એનકે મેળવીશ, જેમ કે હું વત્તા એન.કે એ જે કરતા નાની છે, પણ જો હું એક વધુ પગલું જાઉં, જો હું i વત્તા એન વત્તા 1 વખત કું જશે તો હું પાર કરીશ જમણી તેથી, જો અમારી પાસે એક પગલું હોય તો અમે ત્યાં સુધી જતા રહીશું જ્યાં સુધી આપણે પાર નહીં કરીએ અને અમે જો છેલ્લા સમયથી બંધ થઈશું જે પહેલા જ છે. એક પગલા લેવાથી અમને નીચે ગણતરી કરવામાં મદદ મળે છે અમારે બધાએ પગલું નકારાત્મક રાખ્યું છે. તેથી, જો આપણે કહીશું કે અલ્પવિરામ અને અલ્પવિરામ બાદ 1 તો પછી આપણે તે મૂલ્યથી શરૂ કરીએ છીએ જે અંતિમ મૂલ્ય કરતાં મોટું છે, આપણે તેના ઉત્પાદનથી શરૂ કરીશું, હું બાદ 1, હું ઓછા 2 અને આ રીતે જ કરીશું અને આપણે જ વ્યુ સાથે બંધ કરીશું. 1.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 02:07)

રેજ ફંક્શન માટેનો સામાન્ય નિયમ એ છે કે તમે i થી શરૂ કરો છો અને તમે કેવળના પગલામાં જો કે નકારાત્મક હોય તો તમે વધારો અથવા ઘટાડો, જેમ કે k ના પગલામાં કે તમે જ્યાં સુધી ખાસ કરીને, આનો અર્થ શું છે કે જો તમે આગળ વધી રહ્યા છો, જો તમે પાર કરી રહ્યા છો, જો તમારી પાસે પોઝિટિવ છે, જો તમારી ઇન્ફીમેન્ટ પોઝિટિવ છે તો જો તમે મૂલ્ય સાથે શરૂ કરો જે ખૂબ મોટો હોય તો તમે ખાલી ક્રમ બનાવશો કારણ કે તમે પણ મને જનરેટ કરે છે કારણ કે હું પોતે જ કે જેના બરાબર જે કરતા વધારે મોટો છે તે પછી તે મંજૂરી આપતું નથી. ઊલટાનું નકારાત્મક દિશામાં શું થાય છે એ છે કે જો આપણે મૂલ્ય સાથે શરૂ કરીએ જે લક્ષ્ય મૂલ્ય કરતાં નાનું હોય, તો આપણે પહેલાથી જ નીચે જઈએ છીએ અને તેથી અમે આગળ વધારી શકતા નથી, અમને એક ખાલી ક્રમ મળે છે. આ ખરડાને પાર ન કરી શકાય તેવો વિચાર એ જ કરતાં નાની અટકે છે એવું કહીને જ નથી, કારણ કે જો તમે નકારાત્મક દિશામાં જઈ રહ્યા છો, તો તમે j કરતા વધારે મૂલ્ય પર રોકવા માંગો છો. તેથી, તમે પહેલાં અને પહેલાની જેમ વિચારી શકો છો કે તમે આગળ અથવા પાછળની તરફ જઈ રહ્યા છો તેના આધારે અલગ વસ્તુઓ છે.

ઉદાહરણ તરીકે જોવા માટે, ધારવું કે આપણે 12 થી શરૂ થતી શ્રેણી જોઈએ છીએ અને જેની મર્યાદા 1 છે, પરંતુ ઇન્ક્રીમેન્ટ 3 ઓછા છે. તેથી, આપણે 12 થી શરૂ કરીશું અને પછી આપણે 9 પર જઈશું, તો પછી આપણે 6, તો પછી આપણે 3 પર જઈશું અને જો આપણે એક વધુ પગલા લઈશું તો તમે 0 પર જાઓ છો, પરંતુ 0 થી નકારાત્મક દિશામાં 1 વટાણાની દિશામાં આપણે 3 પર અટકીશું, આપણે 0 થી આગળ વધીએ નહીં. સમય: 03:42)

તે ઘણી વાર નવા પ્રોગ્રામરોને ગૂંચવણમાં મૂકે છે જે આઈ કોમિયા જ રમાતા હોય છે જે વાસ્તવમાં i થી j ઓછા 1 છે અને પોતે જ નહીં. તેથી, આ રીત શા માટે હોવી જોઈએ તે કોઈ વાસ્તવિક કારણ નથી, તે માત્ર સગવડ છે અને મુખ્ય સગવડ એ છે કે તે સૂચિ પર પ્રક્રિયા કરવાનું સરળ બનાવે છે. યાદ રાખો કે જો અમારી પાસે લંબાઈની સૂચિ છે, તો સૂચિમાંની પદવી 0, 1 થી n ની 1 ઘાત ગણવામાં આવે છે. તેથી, ઘણી વાર આપણે શું કરવા ઈચ્છીએ છીએ તે 0 થી n ની 1 ઘાતમાં છે અને જો આપણે અગાઉથી n ને ખબર નથી, તો આપણે લંબાઈ કાર્યનો ઉપયોગ કરીને તેને મેળવીએ છીએ. અમે 0 થી લઈને સૂચિની લંબાઈને ગમશે. જો શ્રેણીને હવે તે વ્યાખ્યાયિત કરવામાં આવે છે, જ્યાં વાસ્તવિક મૂલ્ય ઉપલી મર્યાદા કરતાં એકને અટકે છે, તો શ્રેણીની 0 અલ્પવિરામની લંબાઈ માન્ય સૂચકાંકોની વર્તમાન શ્રેણી રજૂ કરશે. જો બીજી તરફ તે કર્યું, આપણે શું વિચારીશું કે તે વધુ કુદરતી છે અને તે હું, હું 1 વત્તા અપ કરીશ, જો આ તો કેસ હોત તો દરેક વખતે જ્યારે આપણે ખરેખર યાદીમાં હોદ્દો પર પહોંચવા માંગતા હો, તો અમે 0 થી લંબાઈની લંબાઈ 1 થી જવું પડશે. તેથી, અમારે પોતાને યાદ અપાવવા માટે દરરોજ 1 ઓછા કહેવું પડશે કે પોઝિશન એક ટૂંકા ગાળો આપે છે. મુખ્યત્વે આ સગવડ માટે અમે ટી તરીકે યાદીની લંબાઈનો મુક્તપણે ઉપયોગ કરી શકીએ છીએ તે ઉચ્ચ બંધાયેલ છે કે જે સૂચિ બંધ કરે છે, તે શ્રેણી જ બંધ કરવામાં આવે છે 1. જો મેં કહ્યું કે આ નથી, તો મારો મતલબ છે કે તમે સરળતાથી રેન્જ ફંક્શનને વ્યાખ્યાયિત કરી શકો છો જે કુદરતી વસ્તુ છે જે હું, હું 1 વત્તા 1, પરંતુ તે પછી તમને યાદ રાખવું પડશે કે જ્યારે તમે ઇચ્છો કે યોગ્ય સ્થાનેથી ઇન્ડેક્સ રોકવું.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 05:23)

શ્રેણી એક ક્રમ છે અને તે રેન્જને યાદી તરીકે વિચારે છે. અમે જોયું કે માટે 2 ફલેવર્સમાં આવે છે, અમે ક્યાં તો એક સૂચિમાં કહી શકીએ છીએ અથવા અમે શ્રેણીની શ્રેણીમાં હું કહી શકીએ છીએ. તેથી, શ્રેણી 0 થી 10 શ્રેણી 0, 1 થી 9 સુધી ઉત્પન્ન કરે છે. તેથી, આ સૂચિ 0, 1, 2, 3, 4 થી 9 સુધી હું તે માટે કહી રહ્યો છું. બીજા શબ્દોમાં, જો આપણે પાયથોન નીચે આપેલી સરખામણી, શ્રેણી 0 અલ્પવિરામ 10, 0, 1, 2, 3 9 સુધીના સૂચિની સમાન હોય છે, પછી પ્રશ્ન આ સરખામણીના સાચા અથવા ખોટા પરિણામ છે. અહીં, અમે પાયથોન 2 અને પાયથોન 3 વચ્ચેનો તફાવત અનુભવીએ છીએ. પાયથોન 2 માં, શ્રેણીનું આઉટપુટ હકીકતમાં છે, સૂચિ. તેથી, જો તમે આ સમાનતા ચેક ચલાવો છો, તો પાયથોન 2 માં જવાબ સાચું હશે, પરંતુ પાયથોન 3 શ્રેણીમાં અમારા માટે સૂચિ નથી. તેથી, રેન્જ એવી વસ્તુઓ પેદા કરે છે જે મૂલ્યોનો ક્રમ છે જે 'ફોર' જેવા સંદર્ભમાં ઉપયોગમાં લઈ શકાય છે, પરંતુ તકનીકી રૂપે શ્રેણી કોઈ સૂચિ નથી, અમે શ્રેણીના આઉટપુટને મેનિપ્યુલેટ કરી શકતા નથી જે રીતે અમે સૂચિબદ્ધ કરીએ છીએ.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 06:37)

હવે, કાર્ય સૂચિનો ઉપયોગ કરીને સૂચિ બનાવવા માટે શ્રેણીનો ઉપયોગ કરવો શક્ય છે. તેથી, કાર્યનું નામ ખરેખર યાદી છે. આપણે શું કરીએ, ઉદાહરણ તરીકે, રેન્જને યાદીના દલીલ તરીકે આપે છે. તેથી, શ્રેણી દ્વારા નિર્માણ કરાયેલા ક્રમને સૂચિમાં રૂપાંતરિત કરવામાં આવશે. જો હું સૂચિ 0 થી 4 ની ઈચ્છું છું, તો હું કહી શકું છું કે 0 થી 5 ની રેન્જ મને આપો. યાદ રાખો, શ્રેણી 4 પર બંધ કરવામાં આવી હતી કારણ કે 5 ઉચ્ચ બાઉન્ડ છે અને આ ક્રમ કાર્ય સૂચિ દ્વારા યાદીમાં રૂપાંતરિત કરવામાં આવશે. આ પ્રકારના રૂપાંતરણનું ઉદાહરણ છે; અમે રેન્જ દ્વારા પેદા કરાયેલા ક્રમને રૂપાંતરિત કરી રહ્યા છીએ, જો આપણે કહ્યું કે સૂચિમાં સૂચિ નથી, તો તેને સૂચિ બનાવો. તેથી, કાર્ય સૂચિ કંઈક લે છે અને તે યાદી બનાવે છે જો તે

શક્ય છે તે યાદી બનાવવા માટે. જો તે એવી વસ્તુ છે કે જે અર્થમાં નથી કરતી તો તે તમને માન્ય મૂલ્ય આપશે નહીં, હવે આ એક સામાન્ય સુવિધા છે. તેથી, આપણે નામનો ઉપયોગ કરી શકીએ છીએ જે પાયથોન આંતરિક પ્રકારો માટે પણ ઉપયોગ કરે છે, જેમ કે એક પ્રકારને બીજામાં રૂપાંતરિત કરવા માટે કાર્ય કરે છે, ઉદાહરણ તરીકે, જો અમારી પાસે 78 નંબર હોય અને આપણે તેને સ્ટ્રિંગ તરીકે વિચારીએ તો કાર્ય સત્ર તેની દલીલ લેશે અને તેને શબ્દમાળામાં રૂપાંતરિત કરો. તેથી, str કોઈપણ મૂલ્ય લેશે અને તે મૂલ્ય રજૂ કરતી સ્ટ્રિંગમાં તેને રૂપાંતરિત કરશે. આ ઉદાહરણ માટે અસ્પષ્ટ રીતે થાય છે, કારણ કે આપણે જોશું કે પ્રિન્ટ ફંક્શનનો ઉપયોગ કરીને મુલ્ય ક્યારે દર્શાવવી છે. તો, પ્રિન્ટ શું કરે છે તે મૂલ્ય લે છે, તેને સ્ટ્રિંગમાં કન્વર્ટ કરો અને ફક્ત સ્ટ્રીંગ્સ વાસ્તવમાં પ્રદર્શિત થઈ શકે છે, કારણ કે શબ્દમાળાઓ સ્ક્રીન પર જે દેખાય છે તે ટેક્સ્ટ છે અથવા જ્યારે આપણે કંઈક છાપે છે તો ટેક્સ્ટ છે. તેથી, str ખૂબ જ વારંવાર વપરાય છે ક્યારેક અમે રિવર્સને કરવા માગીએ છીએ કે આપણે તેને સ્ટ્રિંગ લેવા અને તેને મૂલ્યમાં રૂપાંતરિત કરવા માગીએ છીએ. તેથી, દાખલા તરીકે જો શબ્દમાળા માત્ર અંકો ધરાવે છે તો આપણે તે સ્ટ્રિંગને અનુરૂપ મૂલ્ય મેળવવું જોઈએ. જો આપણે તેને 324 શબ્દ આપીએ તો તે આપણને પૂર્ણાંક 321 આપવો જોઈએ. તેથી, મૂલ્ય, યાદ રાખો કે કાર્યનું નામ એ પ્રકારનું નામ છે જે તમે રૂપાંતરણ કરવા માંગો છો. તેથી, આપણે એક સ્ટ્રિંગ લેવા અને તેને પૂર્ણાંકમાં બનાવવા માગીએ છીએ, આપણે નામ પૂર્ણાંકનો ઉપયોગ કરીએ છીએ. હવે, આ બધી બાબતોમાં જો તે આવું ન કરી શકે તો વિધેય એક યોગ્ય મૂલ્ય પેદા કરશે નહીં. જો હું ફંક્શન ફંક્શનને એક સ્ટ્રિંગ આપીશ જે નંબરનું પ્રતિનિધિત્વ કરતું નથી તો તે માત્ર મને ભૂલ આપશે તેથી, લાંબા સમય સુધી રૂપાંતરણ શક્ય છે, તે તે કરશે, જો તે ભૂલ ન આપે તો. અમે પાછળથી જોશું કે હકીકત એ છે કે તમને કોઈ ભૂલ મળે છે એ આપત્તિ નથી, કોઈ ભૂલમાંથી પુનઃપ્રાપ્ત કરવા અથવા તે કઈ ભૂલ છે તે ચકાસવા અને તે મુજબ આગળ વધવા માટે પાયથોન માંના રસ્તા છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 09:16)

આપણે જે જોયું છે તે સારાંશ આપવા માટે છે કે આઈ અને જેના શ્રેણીની અમારી સરળ કલ્પનામાં કેટલાક સ્વરૂપો છે. ખાસ કરીને, જો આપણે તેને એક પ્રારંભિક બિંદુ આપતા નથી, તો આપણે તેને એક મૂલ્ય આપીશું, તે ઉચ્ચ બાઉન્ડ તરીકે સમજવામાં આવે છે. તેથી, રેંજ n એ 0, 1, 2, n ની 1 ઘાત સુધી લખવાની એક ટૂંકી રીત છે. આપણે ત્રીજા આર્ગ્યુમેન્ટનો ઉપયોગ કરી શકીએ છીએ જે અનુક્રમ પેદા કરવા માટે એક પગલું છે જે પગલાં, i, k, i વત્તા 2 માં આગળ વધે છે. K અને તેથી પર. ખાસ કરીને, જો k એ નકારાત્મક પગલું છે, તો આપણે ઘટતી સિક્વન્સ ઉત્પન્ન કરી શકીએ છીએ, અને યાદ રાખવા માટેની છેલ્લી વસ્તુ રેન્જ શ્રેણીને ઉત્પન્ન કરે છે, જેનો ઉપયોગ પાયથોન 3 માં જેવી સૂચિમાં બરાબર થાય છે, તે શ્રેણીમાં નથી. જો આપણે રેંજ આઉટપુટમાંથી સૂચિ મેળવવા માંગીએ છીએ તો આ પ્રકારના રૂપાંતરણને યાદી તરીકે ઉપયોગ કરવો જોઈએ. સામાન્ય રીતે, આપણે એક પ્રકારથી બીજા પ્રકારમાં કન્વર્ટ કરવા માટે ટાઇપ નામોનો ઉપયોગ કરી શકીએ છીએ, જો તે કિંમત જે અમે રૂપાંતરિત કરી રહ્યા છીએ તે પ્રકારને અનુરૂપ છે જે અમે રૂપાંતરિત કરવાનો પ્રયાસ કરી રહ્યા છીએ.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રો. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટીટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 03 લેક્ચર - 02 મેનિપ્યુલેટીંગ લિસ્ટ્સ

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 00:02)

તો ચાલો હવે યાદીઓ પર નજર કરીએ. અમે જણાવ્યું હતું કે યાદીઓ ફેરફારપાત્ર પદાર્થો છે તેથી, જો અમારી પાસે યાદી 1 ની સૂચિ છે, જેની મૂલ્યો [1, 3, 5, 6] છે, અને પછી અમે યાદી 2 ને યાદી 2 ની યાદીમાં આપીએ છીએ તો અમે આ કિસ્સામાં યાદી 1 અને યાદી 2 બંનેએ કહ્યું છે કારણ કે સૂચિમાં ફેરફાર યોગ્ય છે એ જ સૂચિ [1, 3, 5, 6] હવે જો હું પોઝિશન 2 લેઉં જે આ પોઝિશન છે અને તેને વેલ્યુ 7 વડે બદલી નાંખે તો તે સ્પષ્ટ રીતે સૂચિ 1 છે [1, 3, 7, 6], પરંતુ કારણ કે યાદી 2 અને યાદી 1 એ જ ઓબ્જેક્ટ તરફ ઇશારો કરતા હતા, તો તે યાદી 2 પાસે પણ છે સમાન મૂલ્ય [1, 3, 7, 6] બીજી તરફ, જો આપણે આ ફેરફારને વધુ રસ્તામાં બનાવ્યું છે. તેથી આપણે શું કર્યું, અમે આ સૂચિ લીધી અને ત્યારબાદ આપણે પ્રથમ 1, 3 0 થી લઇને પોઝિશન 1 નો 2 પર લીધો એટલે મને 1, 3 મળશે. પછી હું 7 દાખલ કરું છું અને પછી વોર્ડ્સ પર પોઝિશન 3 લઇશ. જે 6 છે. પછી મને યાદીમાં [1, 3, 7, 6] મળે છે. પરંતુ બીજી બાજુ, કારણ કે મેં પ્લસનો ઉપયોગ કર્યો છે, મેં જે કર્યું છે તે મેં નવી યાદી બનાવ્યું છે અને તેથી યાદી 2 બદલાઈ નથી, આ કિસ્સામાં યાદી 2 remains [1, 3, 5, 6]; બીજા શબ્દોમાં કહીએ તો, એક નવી સૂચિ ઉત્પન્ન કરવામાં વત્તા પરિણામોનો ઉપયોગ કરીને જોડાણ કરવું.

(સ્લાઇડ સમયનો સંદર્ભ લો: 01:38)

ચાલો આ પાયથોન ના દુભાષિયામાં તપાસીએ. તેથી, જો હું 1, 3, 5, 6 માં ઉદાહરણ તરીકે 1, 3, 5, 6 કહું છું અને હું કહું છું કે યાદી 2 એ યાદી 1 બરાબર છે અને પછી હું ફક્ત યાદી 1 નું સ્થાન 2 બદલી અને પછી યાદી 1 અને યાદી 2 એ 1, 3, 7, 6. પર બીજી બાજુ જો હું કહું છું કે યાદી 1 એ 1, 3, 5, 6 ની બરાબર છે અને યાદી 2 એ યાદી 1 ની બરાબર છે અને હવે હું આ સ્લાઇડ્સ વત્તા એકસાથે રસ્તાની યાદી બદલીશ, તેથી જો હું કહું કે પ્રથમ બે સ્થાનો લો તો 7 અને પછી બાકીની યાદી 1 લે છે. હવે, યાદી 1 ફરી 1, 3, 7, 6 છે, પરંતુ યાદી 2 જે યાદી 1 તરફ ઇશારો કરતું હતું તે હવે સૂચિ 1 તરફ ઇશારો કરતું નથી કારણ કે વત્તાએ નવી સૂચિ બનાવી છે અને તેથી નવી સૂચિ તમારી જૂની યાદીની જેમ નથી, તેથી યાદી 2 જૂની સૂચિ પર નિર્દેશ કરવાનું ચાલુ રાખ્યું છે તેથી તે 1, 3, 5, 6 છે. આ એક મહત્વનો મુદ્દો છે કે જે પરિવર્તનક્ષમતા અંગે ધ્યાનમાં રાખવું જોઈએ. જો આપણે સૂચિને ફરીથી ઉપયોગમાં લઇએ છીએ તો અમે નવી સૂચિ મેળવી શકીએ છીએ, જ્યારે તે કાર્યમાં કાર્ય કરે છે ત્યારે તે પણ લાગુ પડે છે. જો કોઈ વિધેયની અંદર આપણે ફંક્શનની સૂચિને અપડેટ કરવી હોય તો તેટલા લાંબા સમય સુધી આપણે તેને ફરીથી સોંપીશું નહીં કારણ કે અમે ઠીક છીએ, પરંતુ જો આપણે વત્તા ઉપયોગ કરીને ફરીથી સોંપણી મૂકીએ તો તે સૂચિ, જે ફંક્શનની અંદર અપડેટ કરવામાં આવી છે તે ફંક્શનની બહાર દેખાશે નહીં. તેથી, આપણે હંમેશા આ વિશે ખૂબ કાળજી રાખવી જોઈએ.

(સ્લાઇડ સમય જણાવો: 03:13)

હવે, સૂચિને વિસ્તરેલ કેવી રીતે થશે? ધારો કે, અમે સૂચિના અંતે નવા મૂલ્ય 22 ને વળગી રહેવું છે; એમ કરવાની એક રીત એ છે કે એલ એ એલ વત્તા 22 છે. પરંતુ આપણે જોયું તેમ, આ વત્તા ઓપરેટર નવી

સૂચિ બનાવશે, તેથી જો તમે સૂચિમાં મૂલ્ય ઉમેરવાની અને તે જ સૂચિ જાળવવા ઇચ્છતા હોવ, જેથી જો તે વિધેયની અંદર છે, આપણે દલીલ વચ્ચેના સંબંધને ગુમાવતા નથી અને આ કાર્યની અંદર મૂલ્યની હેરફેર કરવામાં આવશે નહીં. અમે આ ફંક્શનને પ્રથમ જ સપ્તાહમાં જીસીડી કર્યું ત્યારે પસાર થવા સાથે જોડાયું હતું. એડેડ એ કાર્ય છે જે યાદી લેશે અને તેની કિંમત ઉમેરશે. તો અહીં આપણે કહ્યું છે કે યાદી 1 એ 1, 3, 5, 6 છે, જેમ કે અગાઉના ઉદાહરણો. યાદી 2 એ યાદી 1 છે અને હવે આપણે કહ્યું છે કે યાદી 1 અને 12 જોડવું. તેથી, યાદી 1 જે રીતે આપણે લખ્યું છે તે યાદી 1 ડોટ એપેન્ડ છે અને ઉમેરામાં આપણે દલીલને ઉમેરવાની નવી કિંમત આપીએ છીએ. તેથી આ શું કરે છે, અલબત્ત તે યાદી 1 બનાવશે જે હવે મૂળ 1, 3, 5, 6 અને એક નવો મૂલ્ય 12 સાથે પાંચ ઘટકોની સૂચિ બનાવશે, પરંતુ અગત્યની રીતે આ જૂની સૂચિ છે 1 તે નવી સૂચિ નથી તે અર્થમાં તેથી, યાદી 2 પણ બદલાઈ ગયો છે. વાસ્તવમાં મૂલ્ય ઉમેરે છે યાદી 1 અને યાદી 2 બિંદુ, અંતે 12 સાથે નવી સૂચિ છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 04:37)

બીજી બાજુ, જો આપણે તે કર્યું હોત તો મેં વત્તા ઓપરેટરનો ઉપયોગ કરીને ઉલ્લેખ કર્યો છે, તો આપણે તે યાદી 1 ફેરફારો શોધી શકીએ છીએ, પરંતુ યાદી 2 એ નથી કારણ કે જેમ આપણે જોયું તેમ કન્સેટેનમેશન નવી યાદી બનાવે છે તેથી, એપેન્ડ એક કાર્ય છે જે તેને બદલીને નવો મૂલ્ય ધરાવતી સૂચિને વિસ્તૃત કરે છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 04:48)

તેથી, ઉમેરોમાં એક જ મૂલ્ય લે છે. હવે, જો આપણે કોઈ એક મૂલ્ય ઉમેરવું ન હોત, પરંતુ મૂલ્યોની સૂચિ; અમે વાસ્તવમાં એક સૂચિ લેવા માંગીએ છીએ અને અંતે તેને એક સૂચિ ઉમેરીને વિસ્તૃત કરવા માંગીએ છીએ, અમે 1, 3, 5 કહેતા હતા અને અમે 6, 8, 10 મુકવા માંગીએ છીએ. તેથી, આપણે 1, 3, 5 લેવા માંગીએ છીએ અને અમે આને વધુ ત્રણ મૂલ્યો ધરાવતા હોવાનું વિસ્તૃત કરવા માગે છે, અલબત્ત અમે આમાંના દરેક મૂલ્યને એક સમયે એક ઉમેરી શકીએ છીએ. પરંતુ એક કાર્ય છે જે આપવામાં આવે છે જે એપેન્ડની યાદીને વિસ્તરે છે, પરંતુ અહીં આ સૂચિ પોતે જ હોવી જોઈએ. તેથી વિસ્તૃત યાદીને દલીલ તરીકે લે છે, જોડણી દલીલ તરીકે મૂલ્ય લે છે તેથી, યાદી 1 વિસ્તૃત યાદી 2 એ કહીને સમકક્ષ છે યાદી 1 1 યાદી સમાન છે પ્લસ યાદી 2, પરંતુ યાદ રાખો કે આ એક યાદી હોવી જ જોઈએ, તે એક મૂલ્ય નથી, તે મૂલ્યનો અનુક્રમ નથી, તે એક સૂચિ છે, તેથી તેને આપેલું હોવું જોઈએ ચોરસ કૌંસમાં તમારે 6, 8, 10 ને વિસ્તૃત કાર્ય માટે દલીલ તરીકે આપવું આવશ્યક છે. હવે, આ સૂચિમાં તત્વો ઉમેરવાનું છે, સૂચિમાંથી એક ઘટક દૂર કરવા માટે એક માર્ગ પણ છે. તેથી, આ કિંમતને સ્પષ્ટ કરીને તેને દૂર કરવાનો એક માર્ગ છે. આપણે કોઈ ચોક્કસ પદ પર નજર રાખતા નથી જે આપણે મૂલ્ય x અને યાદી 1 શોધી રહ્યા છીએ તે યાદીમાં x ની પ્રથમ ઘટનાને દૂર કરે છે. હવે, તમે પૂછશો કે શું થાય છે જો સૂચિમાં કોઈ ઘટના નથી. ઠીક છે, હકીકતમાં આ અમને એક ભૂલ આપશે જેથી તમારે ફક્ત દૂર કરવા માટે સાવચેત રહેવું જોઈએ જો તમને ખબર હોય કે ત્યાં ઓછામાં ઓછી એક એક્સની નકલ છે અને યાદ રાખો કે તે માત્ર ત્યારે જ પ્રથમ ઘટના દૂર કરે છે, બધી ઘટનાઓને દૂર કરી નથી તેથી, જો x ની દસ ઘટનાઓ યાદી 1 માં હોય તો માત્ર પ્રથમ જ દૂર કરવામાં આવશે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 06:30)

ચાલો આપણે આ બાબતોની તપાસ કરીએ. ચાલો આપણે સૂચિ 1 સાથે શરૂ કરીએ, તેથી અગાઉના લેક્ચરમાંથી યાદ રાખો કે અમે શ્રેણી લઈ શકીએ અને સૂચિ બનાવી શકીએ. હવે જો હું આ કરું તો મારી પાસે સૂચિ 1 છે 0, 1, 2 થી 9, હવે જો હું કહું કે યાદી 1 ડોટ એડેડ 12, તો પછી યાદી 1 એ 12 થી ઉમેરાયું છે. હવે જો હું કહું તો યાદી 1 ડોટ 13, 14 કહે છે, તો યાદી 1 હવે 13 છે, 14 દેખાય છે. તેથી આ કેવી રીતે ઉમેરે છે અને કામ વિસ્તરે છે હવે ધારણા છે, માત્ર દલીલ ખાતર હું યાદી 2 લઈ છું અને હું યાદી 1 ની બે

નકલો બનાવે છે. હવે, યાદી 2 એ 0 થી 14 થી 10 થી 11 ની વચ્ચે 10 થી 11 ની વચ્ચે છે, ફરીથી 0 થી 14 માં. હવે જો હું કહું છું કે યાદી 2 dot remove કહે છે 5, હવે ત્યાં 5 ની બે નકલો છે જે અહીંની પ્રથમ નકલ છે જે અહીં છે. શરૂઆતમાં અને બીજી નકલ જે પાછળથી છે, તેથી આ પ્રથમ નકલ દૂર કરશે હવે, જો હું સૂચિ 2 ને જોઉં તો પ્રથમ એક 4 થી 6 સુધી અવગણો, પણ બીજી નકલ હજુ પણ ત્યાં છે. જો હું તેને ફરીથી કહું તો બન્ને નકલો નીકળી ગયા છે, કારણ કે મારી પાસે આ 4, 6 નથી, અને મારી પાસે 5 પણ નથી, અહીં ફરીથી 4, 6. હવે શું થાય છે જો હું તેને ત્રીજી વખત હટાવી દઈશ, હવે મને મળશે એક્સ કહેતી ભૂલ યાદીમાં નથી. યાદ રાખો કે દૂર કરે તો જ કામ કરે છે જો x એ સૂચિમાં હોય, જો તે સૂચિમાં ન હોય તો તમને ભૂલ મળે છે હવે તે અગત્યનું છે અમે પછીથી જોશું કે અમને ભૂલ મળે છે તેનું નામ પણ છે, તેથી તે મૂલ્યની ભૂલ કહે છે. આ ઉપયોગી બનશે કારણ કે પાછળથી આપણે શોધીશું કે પાચથોનની અંદર આપણે ખરેખર ભૂલોનું પરીક્ષણ કરી શકીએ છીએ અને કોઈ ભૂલ આવી હોય તો વૈકલ્પિક ક્રિયા કરી શકીએ છીએ અને અમે ભૂલ દર્શાવી શકીએ છીએ કે ભૂલ કેટલી વાર થાય છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 08:18)

એન્ડેન્ડ ફંક્શન આપણે અત્યાર સુધી જોયેલી અન્ય ફંક્શનથી થોડો અલગ જુએ છે. અમે સામાન્ય રીતે એવી અપેક્ષા રાખીએ છીએ કે કાર્ય બે દલીલો લે છે; સૂચિ અને મૂલ્ય જોડવા માટે. તેથી આપણે વિચારીએ કે યોગ્ય રીતે અથવા અપડે લખવાનો પ્રાકૃતિક રસ્તો એ છે કે યાદી -1 માં ઉમેરવાનું મૂલ્ય x બીજી બાજુ, આપણી પાસે જે છે તે આ રમુજી સંકેતલિપી છે જે કહે છે કે યાદી 1 એ મૂલ્ય x સાથે જોડાયેલા ફંક્શનને લાગુ કરો. એક પાચથોન પરિભાષા સૂચિ 1 માં એક ઓબ્જેક્ટ છે અને ઓડેડને અપડેટ કરવા માટે કાર્ય છે અને એક્સ ફંક્શન એપ્ડને દલીલ તરીકે માનવામાં આવે છે. આવી પરિસ્થિતિમાં અમારી પાસે એક ઓબ્જેક્ટ છે અને પછી આપણે તેના માટે એક ફંક્શન લાગુ કરીએ છીએ, તેથી આપણે ઓબ્જેક્ટને ઓબ્જેક્ટ ફાળવવાને બદલે ડોટ નોટેશનનો ઉપયોગ કરીને ઓબ્જેક્ટ સાથે જોડાયેલા ત્રણ વિધેયોનો ઉપયોગ કરીએ છીએ જે વધુ સામાન્ય રીત છે જે તમને લાગે છે વિધેયો અમે પછી આ બિંદુ પર પાછા આવીશું અને હવેથી બે - ત્રણ અઠવાડિયા હોઈ શકે છે અને અમે પાચથોનની અંદર ઓબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગ તરીકે ઓળખાય છે.

(સ્લાઈડ સમયનો ઉલ્લેખ કરો: 09:28)

લિસ્ટ અને સ્થાનને વિસ્તૃત કરવા અને કોન્ટ્રેક્ટ કરવાનો એક બીજો રસ્તો છે, આ એક સ્લાઇડમાં સીધા જ નવા મૂલ્યો આપવાનું છે. તેથી, અમે અમારા જૂના ઉદાહરણ પર પાછા જઈએ છીએ: યાદી 1 એ 1, 3, 5, 6 અને યાદી 2 એ યાદી 1 છે. હવે આપણે શું કહીએ છીએ તે છે કે ટુકડાને વોર્ડ્સ પર પોઝિશન 2 થી લઈને તેને 7, 8 નું મૂલ્ય આપો. તેથી યાદ રાખો કે સ્થિતિ 0, 1, 2, 3 છે. તેથી આ શું કહે છે, આ સ્લાઇડ 5,6 અને તેને 7, 8 વડે બદલો. આપણે શું મેળવીએ છીએ તે અલબત્ત યાદી 1 એ સ્લાઇડ 5, 6 ને 7, 8 ની જગ્યાએ લીધું છે, પરંતુ આ સ્લાઇડ રિપ્લેસમેન્ટ સ્થાને થાય છે. આપેલ પદ પર નવો મૂલ્ય આપવા જેવી થોડી છે. જો હું કહું કે લિસ્ટ 2 બરાબર 7 છે, તો આપણે કહીએ કે પોઝિશન બે 7 ની બની જાય છે. એ જ રીતે જો હું કહું કે યાદી 1 થી સ્લાઇડ -2 ના અંત સુધી 7, 8 તે 5, 6 થી 7, 8 બંનેને યાદી 1 માં બદલાય છે, પરંતુ તે પણ તે નિર્દેશ કરતી નથી જ્યાં તે નિર્દેશ કરે છે, તેથી યાદી 2 પણ પ્રભાવિત થાય છે તેથી તે બન્ને હવે 1, 3, 7, 8. કહે છે, અહીં આપણે બે લંબાઈનો ટુકડો મુક્યો અને તેને લીટીના બેની નવી સૂચિ દ્વારા બદલી. તેથી, અમે સ્થાનોની સંખ્યાના આધારે સૂચિનું માળખું સાચવી રાખ્યું છે આ આવશ્યક નથી, પાચથોન તમને સ્લાઇડ વિસ્તૃત અને સંકોચો બંને કરવાની પરવાનગી આપે છે. દાખલા તરીકે, તમે તે સૂચિ લઈ શક્યા હોત, હવે ચાલો કહીએ કે આ 1, 3, 7, 8 છે અને ફરીથી અમે સ્લાઇડ 2 આગળ લઈ જઈએ છીએ, જે બે પોઝિશન્સ ધરાવે છે અને આપણે કહી શકીએ છીએ તેને ત્રણ મૂલ્યો સાથે સૂચિ દ્વારા બદલો. અમે કહીએ છીએ કે આ સૂચિ 2 થી 3 ની છેલ્લી બે હોદ્દાને લઈએ અને તેને ત્રણ મૂલ્યો સાથે બદલો અને આપણે જે મેળવીએ છીએ તે જૂની 1, 3 છે અને આ સ્લાઇડ હવે 9, 10, 11. બની ગયું છે. ચાર તત્વની સૂચિ પાંચ તત્વોની યાદી બની જાય છે. સ્લાઇડનો ઉપયોગ કરીને સૂચિને વિસ્તૃત

કરવાની આ એક રીત છે. બીજી વસ્તુ જે આપણે કરી શકીએ તે એક યાદી સંકોચો છે; અમે નાની વસ્તુ મૂકી શકીએ છીએ ધારો કે, અમે સૂચિની 1 પર 0 માં ફક્ત એક મૂલ્ય ધરાવીએ છીએ, તેથી અમે સ્લાઇસ 0 થી 2 લઈએ છીએ જે અમને આ બે પોઝિશન્સ આપશે તેથી હવે તમારી પાસે બે લંબાઈનો ટુકડો છે, પરંતુ અમે તેને એક યાદી સોંપી લંબાઈ એક તેથી, આ 1, 3 ને માત્ર એક જ 7 દ્વારા બદલવામાં આવે છે. આ અગાઉના સંક્ષેપો પછી હવે અમારી પાસે લંબાઈ પાંચની યાદી હતી, જે હવે આ સંકોચન પછી ચાર વખતની સૂચિ બની છે. કાપી નાંખવાની સાથે તમે સ્થાને સ્લાઇસને બદલી શકો છો, આ તમે જે લખ્યું છે તેના આધારે મોટી સૂચિ અથવા નાની સૂચિ બનાવી શકો છો, પરંતુ તમે કલ્પના કરી શકો છો કે આ ખૂબ ગૂંચવણભરી હોઈ શકે છે. તેથી, તમારે ખૂબ કાળજી રાખવી જોઈએ કે તમે જાણો છો કે તમે શું કરી રહ્યા છો જો તમે લિસ્ટમાં સ્લાઇસેસને સીધા અપડેટ કરવાનો પ્રયાસ કરી રહ્યા છો.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 12:07)

એક સૂચિ વિશે આપણે જાણવા માગીએ છીએ તે એક સામાન્ય વસ્તુ છે કે શું કોઈ યાદીમાં મૂલ્ય અસ્તિત્વમાં છે કે નહીં. તેથી, પાચથોન એ ખૂબ જ સરળ અભિવ્યક્તિ ધરાવે છે જેનું નામ x છે. તો, 1 માં x સાચું આવે છે જો મૂલ્ય x એ સૂચિમાં જોવા મળે છે. હવે અમે આનો ઉપયોગ સલામત કામગીરીને દૂર કરવા માટે કરી શકીએ છીએ; આપણે 1 નો ઉકેલ લાવો તે પહેલા અમે પ્રથમ તપાસીએ છીએ કે x ખરેખર 1 માં છે. તેથી, જો x 1 માં હોય તો, શરત સાચી હશે અને માત્ર પછી આપણે તેને દૂર કરવાનો પ્રયત્ન કરીશું, જો x એ 1 ન હોય તો આપણે x ને કાઢી નાંખીશું. આ કિસ્સામાં અમે ગેરેંટી આપીએ છીએ કે 1 dot remove એ ભૂલ પ્રક્ષેપણ સંદર્ભમાં કહી શકાશે નહીં જ્યાં તે કહેશે કે 1 માં કોઈ x નથી. પણ યાદ છે કે દૂર દૂર માત્ર પ્રથમ તત્વ દૂર કરે છે. અમે આને બદલીને થોડા સમય સુધી બદલી શકીએ છીએ અને કહી શકીએ છીએ કે લાંબા સમય સુધી મૂલ્ય x છે, જ્યાં હું અરજી લાગુ કરું છું. આ એક જ ટૂંકમાં જ તમામ x એ '1' માં દૂર કરશે કારણ કે દરેક સમયે અમે x દૂર કરીએ છીએ અને પાછા જઈએ છીએ કે જો ત્યાં x માં 1 હોય તો 1 માં x પર હજી આપણે તેને દૂર કરીએ છીએ, એટલે ડાબેથી જમણે આ લૂપ એ તમામ x ને એલ માં દૂર કરશે.

(સ્લાઈડ ટાઇમ નો ઉલ્લેખ કરો: 13:14)

હવે સૂચિ માટે વ્યાખ્યાયિત અન્ય કાર્યો છે, દાખલા તરીકે, 1 નો ડોટ વિપરીત સૂચિને બદલે બદલાશે, 1 ડોટ સોર્ટ ચઢતા ક્રમમાં સૂચિને સોર્ટ કરશે. તમે અન્ય ઓર્ડરમાં પણ તેને સોર્ટ કરી શકો છો કે જે તમે જોઈ શકો છો અને તે કેવી રીતે કરવું તે જુઓ. જો આપણે ફક્ત એ જાણીએ કે એક તત્વ એલમાં છે કે નહીં તે આપણે કહીએ છીએ કે આપણે x માં એલ લખીએ છીએ, પરંતુ જો આપણે જાણવું હોય કે તે ક્યાંથી આવે છે તો આપણે ઇન્ડેક્સનો ઉપયોગ કરીએ છીએ તે ડાબી બાજુની સ્થિતિને શોધી કાઢશે, પરંતુ ફરીથી તે આપણને કોઈ ભૂલ આપશે x યાદીમાં છે, તેથી આપણે જો પ્રથમ 1 માં x તપાસવું જોઈએ અને પછી ડાબી બાજુની સ્થિતિનું ઇન્ડેક્સ શોધીએ. હવે તમે ડાબીબાજુ અથવા જમણી બાજુની સ્થિતિ ન જોઈએ જેથી ત્યાં એક આર ઇન્ડેક્સ હોય અને ત્યાં અન્ય કાર્યોનો યજમાન હોય અને તમારે પાચથોન દસ્તાવેજો જોવાની જરૂર નથી, આ કોર્સ કે કોઈ પણ કોર્સ દરેક ફક્શનને આવરી શકે છે જે વ્યાખ્યાયિત છે દરેક પ્રકાર માટે પાચથોનમાં. તો તમારે દસ્તાવેજો જોવાની જરૂર છે અને જો તમને લાગે કે કાર્ય હોવું જોઈએ જે કુદરતી રીતે કંઈક કરે છે તો ઘણી વાર હશે. તેથી, પ્રયાસ કરો અને જુઓ અને તે કેવી રીતે કાર્ય કરે છે તે જુઓ અને તેનો ઉપયોગ કરવાનો પ્રયાસ કરો. જો તમારી પાસે કોઈ પ્રશ્ન છે કે જો હું આ સારી કરું છું તો શું થશે? પાચથોન એક ઇન્ટરેક્ટિવ ભાષા છે જો હું આવું કરું તો શું થાય? ફક્ત તેને અજમાવી જુઓ અને જુઓ અને દુભાષિયોમાં તમે જે જુઓ છો તેમાંથી બહાર કાઢવાનો પ્રયાસ કરો, કાર્ય કેવી રીતે કાર્ય કરે છે, જો ત્યાં દસ્તાવેજમાં કેટલીક અનિશ્ચિતતા હોય તેમ લાગે છે. પરંતુ બધા ઉપર, ફક્ત દસ્તાવેજોને જોઈને દસ્તાવેજમાં જોવાથી ડરશો નહીં, જે તમને જરૂરી વિષયો શીખવા માટે સક્ષમ હશે કારણ કે તે ખૂબ જ મુશ્કેલ છે કારણ કે મેં કહ્યું છે કે ત્યાં દરેક સંભવિત કાર્ય છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 14:48)

યાદીમાં અંતિમ મુદ્દો એ છે કે આપણે પસાર થવાની વાત કરી છે, કેમ કે નામોમાં પાયથોન માં પ્રકારો નથી, તેથી આપણે નામ જાહેર કરવાની જરૂર નથી, ફક્ત કોડસની જેમ જ પ્રોગ્રેસ થાય છે. તેથી જ્યારે કોઈ નામ આપણને પોપ કરે છે ત્યારે તે જાણવા માટે પાયથોન ને તે શું મૂલ્ય છે તે જાણવાની જરૂર છે. ખાસ કરીને પ્રથમ વખત જ્યારે આપણે કોઈ નામનો ઉપયોગ કરીએ છીએ ત્યારે આપણે તેને અસાઇનમેન્ટના ભાગ રૂપે મુકવો પડશે, આપણે તેને મૂલ્ય આપવું પડશે અને તે મૂલ્ય કંઈક હોવું જોઈએ જે વર્તમાન નામોને ધ્યાનમાં રાખીને આપવામાં આવે છે. તેથી જો આપણે દાખલા તરીકે નામ વાચ અને x વત્તા 1 ને અસાઇન કરવા માગીએ છીએ, તો આ બિંદુએ પ્રત્યેક એક્સમાં મૂલ્ય હોવું જોઈએ, નહિ તો $x + 1$ નું મૂલ્યાંકન કરી શકાતું નથી. તેથી, જો x એ પહેલા અને મારા કોડમાં પહેલી વખત જોવામાં ન આવ્યો હોય તો હું તેને એસાઇનમેન્ટની જમણી તરફ જોઈ શકું છું એટલે તેનો અર્થ એ કે હું x ની કિંમત પેદા કરું છું, પણ કોઈ વેલ્યુ અત્યાર સુધી સોંપવામાં આવી નથી અને આ તમને ભૂલ આપશે. આ શોધવું સહેલું સરળ છે, તેથી જ્યારે તમે કંઈક લખો છો અને તમે જમણી બાજુ પર કંઈક જુઓ છો અને તમે તેને પહેલાં જોયું નથી, તો તેનો અર્થ એ કે એક પાયથોન ભૂલને ધ્વજ કરશે અને તે શા માટે છે તે સમજવું મુશ્કેલ નથી. હવે આપણે જે પ્રકારની યાદી ઇન્ફ્રેશન જોયું છે, તે થોડી વધુ ગૂઢ છે. જ્યારે હું કહું છું કે 1 નો જોડીએ તો ત્યાં કોઈ ચિહ્ન જોવું નથી. તેથી તે તાત્કાલિક સ્પષ્ટ નથી કે લેંટ એન્ડએન્ડ વીને પહેલાથી જ યાદી મૂલ્ય હોવું જરૂરી છે, હું હમણાં જ ખાલી યાદીમાં v તરીકે શામેલ કરી શકતો નથી. ઠીક છે, અલબત્ત હું ખાલી સૂચિમાં v ને ઉમેરી શકું છું કે કેવી રીતે પાયથોન જાણે છે કે એલ ખાલી યાદીમાં છે. તેથી, પાયથોન ને એ જાણવાની જરૂર છે કે 1 એ એક સૂચિ છે, તે આ એપેન્ડ ઇન્ફ્રેશનને લાગુ પાડવા પહેલાં.

(સ્લાઈડ સમયનો સંદર્ભ લો: 16:21)

તેથી, આપણે આ નાની કાર્ય પહેલાં જોયું જે n ના પરિબલોને ગણતરી કરે છે. તેથી અનિવાર્યપણે તે શું કરે છે, તે તમામ નંબરો શ્રેણી 1 થી n માં લે છે. હું 1 થી n વત્તા 1 લે છે તેથી હું કમ 1 થી n સુધી ચાલું છું. અને જો કોઈ સંખ્યાને સરખે ભાગે વહેંચી જાય તો જો કોઈ બાકી ના હોય તો મેં એડેન્ડ ઇન્ફ્રેશનનો ઉપયોગ કર્યો છે જે હવે હું જે પરિબલોને પરત આપશે તેની યાદીમાં ઉમેરવું છું. હવે, આ સાથે કેચ એ છે કે જ્યારે હું આ નિવેદનમાં પહેલી વાર આવું ત્યારે પહેલું પરિબલ જે અલબત્ત 1 હશે કારણ કે 1 હંમેશાં એક પરિબલ હશે. પાયથોન ને પૂછવું જોઈએ કે શા માટે ફ્લિસ્ટ પાસે મૂલ્ય ઉમેરવાની ક્ષમતા છે, કારણ કે ફ્લિસ્ટને આ બિંદુથી ક્યારેય મળ્યું નથી. અમે કોડ લખ્યો ત્યારે અમે ખૂબ કાળજી રાખતા હતા, અલબત્ત અમે વત્તા ઉપયોગ કર્યો છે કારણ કે અમે તે કોડમાં અપડેલનો ઉપયોગ કર્યો નથી પરંતુ તે એક જ વાત છે. અમે આ આરંભ દાખલ કરવા માટે ખૂબ કાળજી રાખી. આ આરંભ માત્ર ત્યારે જ જરૂરી છે જ્યારે આ પ્રથમ એન્ડે થાય છે જ્યારે તે ખરેખર તે કેસ છે કે જે ફ્લિસ્ટ પ્રકારની સૂચિ છે અને તેથી એપેન્ડ ઇન્ફ્રેશન આ નામ પર લાગુ કરવા માટે એક માન્ય કાર્ય છે, આ વગર તમને ભૂલ મળશે. યાદ રાખો કે તમારે હંમેશાં ખાતરી કરવી પડશે કે તમે જે નામનો ઉપયોગ કરો છો તે મૂલ્યને પ્રથમ વખત આરંભ કરવામાં આવે છે, જેથી જ્યારે પણ તે પછીથી દેખાય ત્યારે મૂલ્ય સ્પષ્ટ હોય છે અને તેથી આ નામ માટે કયા ઓપરેશનની મંજૂરી છે તે પણ પાયથોન.

(સ્લાઈડ સમયનો સંદર્ભ લો: 17:47)

સારાંશમાં, આપણે જે જોયું તે એ છે કે આપણે એપેન્ડ, ફેક્સ્ટ અને તેથી વધુ જેવા ઇન્ફ્રેશનોની મદદથી લિંક્સને વિસ્તૃત કરી શકીએ છીએ. અમે યાદીની એક સ્લાઇડમાં નવી મૂલ્ય પણ આપી શકીએ છીએ અને પ્રક્રિયામાં વિસ્તૃત અથવા સૂચિને કરાર કરી શકીએ છીએ, પરંતુ આ કાળજી સાથે કંઈક થઈ શકે છે; તમારે ખાતરી કરવી જોઈએ કે તમે જાણો છો કે તમે શું કરો છો સૂચિમાં કાર્યોમાં ઘણાબધા બાંધી છે; આપણે તેમાંના કેટલાંકને જોશું અને તેમનો ઉપયોગ કરીએ છીએ અને તેમનું વર્ણન કરીએ છીએ તેમ જ તેમને વર્ણવીએ છીએ, પરંતુ તે બધાને દસ્તાવેજીકૃત કરવું અઘરું છે અને તે બધામાંથી પસાર થવું

અશક્ય છે અને તે એક વિધેયોનો ટોળું તેથી, ટ્યુટોરીયલ અને અન્ય દસ્તાવેજીકરણ જુઓ જે ઉપલબ્ધ છે જે મેં અગાઉનાં અઠવાડિયામાં આપેલ છે, જેથી તમે શોધી શકો છો કે કયા પ્રકારનાં કાર્યો ઉપલબ્ધ છે. અને છેવટે, ભૂલશો નહીં કે તમારે કોઈ પણ નામ માટે વેલ્યુ અસાધન કરવા જ જોઈએ તે પહેલાં તે પહેલી વખત વપરાય છે, કારણ કે નામો સ્વયંના પ્રકારો નથી, તેથી પાયરેનને આપેલ નામ સાથે શું કરવું તે ખબર નથી.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રોફેસર. માધવન મુકુંદ,
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 03 લેક્ચર - 03 લૂપમાંથી બહાર ભટકા

(સ્લાઈડ ટાઇ નો ઉલ્લેખ કરો: 00:01)

ચાલો લૂપને ફરીથી આવો. તેથી, આપણે જોયું કે પાયથોન માં આપણી પાસે બે પ્રકારના લૂપ્સ છે. પ્રથમ 'લૂપ માટે' છે જ્યાં મૂલ્ય સામાન્ય રીતે સૂચિ પર હોય છે અથવા શ્રેણી ક્રમના આઉટપુટને કહે છે; તેથી હું 1 માં યાદીમાં દરેક વસ્તુ માટે શરીર પુનરાવર્તન કરશે. ક્યારેક આપણે જાણતા નથી કે કેટલી વાર આપણે અગાઉથી પુનરાવર્તન કરવાની જરૂર છે, તો પછી અમે જ્યારે સ્વરૂપોનો ઉપયોગ કરીએ છીએ. જ્યારે કોઈ શરત લે છે અને તે શરીરને પુનરાવર્તન કરે છે ત્યાં સુધી શરત ખોટી બની જાય છે. પરંતુ આ બન્ને કિસ્સાઓમાં નોટિસ કરવાની મુખ્ય વસ્તુ એ છે કે લૂપ કેટલો સમય લેશે તે અગાઉથી નક્કી થાય છે અથવા તો અમારી પાસે ફોલેક્સ માટે ફલકમાં સંખ્યાઓના આધારે સંખ્યા છે, અથવા આપણે સ્થિતિ ચાલુ રાખીએ ત્યાં સુધી ચાલુ રાખીએ છીએ. અમે તે પહેલાં બંધ ન કરી શકો છો હવે તે ઘણીવાર કુદરતી પરિસ્થિતિઓ છે કે જ્યાં અમે ટૂંકા ટૂંકા કાપે છે અને ખૂબ જ અંત સુધી આગળ વધવું નથી. ધારો કે, આપણે ફંક્શન ફોક્સપોઝ લખવું છે જે યાદીમાં મૂલ્ય વીની પ્રથમ સ્થાન શોધી કાઢે છે. આપણે જોયું કે ઇન્ડેક્સમાં બિલ્ટ ઇન ઇન્ડેક્સ છે જે ખૂબ જ આ કરે છે, તેથી 1 ડોટ ઇન્ડેક્સ વી પ્રથમ સ્થાનની જાણ કરશે જ્યાં v યાદીમાં થાય છે. જો કે, 1 માં v ની કોઈ ઘટના ન હોય તો ઇન્ડેક્સ ફંક્શન ભૂલની જાણ કરશે. તેથી, આપણે તે સહેજ બદલવા માંગીએ છીએ અને આપણે શું કહેવા માંગીએ છીએ તે છે કે જો 1 ની સંખ્યા 1 માં નથી. તો ક્યાં તો આપણે 0 થી n નીચલો 1 ની વચ્ચે મૂલ્ય મેળવીએ, જ્યાં n એ સૂચિની લંબાઈ છે અથવા આપણે મૂલ્ય બાદ 1 મેળવો અને કહીને કે વિ સૂચિમાં નથી. અહીં એક જ્યારે લૂપ અમલીકરણ છે. નામ મળ્યું છે તે નામ સૂચવ્યું છે, તે સૂચવવા માટે સૂચવે છે કે મૂલ્ય મળ્યું છે કે નહીં તેટલું કારણ કે અમે અત્યાર સુધી જોયું છે. શરૂઆતમાં આપણે 1 માં કોઈ મૂલ્યો જોયો નથી, તેથી v છે, મળી - ખોટી છે. અને હું પોઝિશન્સ મારફતે જવા માટે નામ તરીકે ઉપયોગ કરું છું. તેથી હું શરૂઆતમાં 0 છું અને જ્યાં સુધી અમે 1 ની લંબાઈ નહીં ત્યાં સુધી હું વધારીશ. તેથી, જ્યાં સુધી હું 1 ની લંબાઈ કરતાં ઓછું હોય ત્યારે, જો આપણે તે મૂલ્ય જોઈએ છીએ જે આપણે શોધી રહ્યા છીએ તો અમે અપડેટ સાચી હોઈ શકીએ છીએ અને અમે તે સ્થાનને ચિહ્નિત કરીએ છીએ જે અમે તેને પીઓઆઈ આની અંતમાં, જો આપણે મૂલ્ય ન મેળવ્યું હોય, તો મળી આવેલ મૂલ્ય સાચું નથી સુયોજિત કરવામાં આવ્યું છે, એટલે કે સૂચિમાં કોઈ વિ ન હોય તો, આપણે ઓછા મુદ્રા 1 માટે pos સુયોજિત કરીશું જે શરૂઆતમાં આપણે દર્શાવેલ મૂળભૂત મૂલ્ય છે. ત્યારબાદ આપણે વર્તમાનનું મૂલ્ય પરત કરીએ છીએ જે ક્યાં તો આપણે મળ્યું છે તે મૂલ્ય છે અથવા મૂલ્ય 1 થી નીચે રહેશે કારણ કે અમને તે મળ્યું નથી. આના વિશે અવલોકન કરવા બે બિંદુઓ છે; પ્રથમ મુદ્દો જે હાથમાં મુખ્ય મુદ્દો છે તે છે કે આપણે દરેક સ્થાને હું 0 થી લઇને 1 ઓછા 1 ની લંબાઈ સુધી જઈશું, પછી ભલે અમે તેને શોધીએ. ધારો કે, આપણી યાદીમાં સેંકડો હજારો તત્વો છે અને અમે બીજી સ્થાને મૂલ્ય મેળવીએ છીએ અને બાકીની સેંકડો સ્થિતિ સ્કેન કરવા જઈ રહ્યા છીએ. હવે આ ખૂબ બિનજરૂરી લાગે છે અન્ય બિંદુ, જે શુદ્ધતાનો મુદ્દો છે, આ કાર્યક્ષમતાનો મુદ્દો છે કે આપણે આ લૂપ ઘણીવાર બિનજરૂરી રીતે ચલાવી રહ્યા છીએ. જો આપણે વાસ્તવમાં પ્રથમ મૂલ્યનો ખૂબ જ પ્રારંભિક અહેવાલ આપી શકીએ છીએ બીજી સમસ્યા જે શુદ્ધતા છે તે છે કે આપણે પ્રથમ સ્થાનની જાણ કરવા માંગીએ છીએ, પરંતુ જે રીતે અમે ચાલુ રાખીએ છીએ તે દરેક વખતે અમે પ્રથમ સ્થાનને પસાર કરીએ છીએ અને અમને બીજી એક નકલ મળે છે જે અમે નવી વસ્તુ બનવાની સ્થિતિને અપડેટ કરી રહ્યા છીએ. તેથી, આપણે વાસ્તવમાં પ્રથમ સ્થાને નહીં પરંતુ છેલ્લા સ્થાને છીએ તેથી આ આ કરવા માટેનો એક સારો માર્ગ નથી. તેથી, આપણે પહેલા તે

બદલવું જોઈએ. તેથી અમે કહીએ છીએ કે આપણે પ્રથમ સ્થાન જોઈએ છે. તેથી, અમે પહેલી પોઝિશન માગીએ છીએ કે આપણે આ સુધારા માત્ર ત્યારે જ કરવા માગીએ છીએ જો અમને મૂલ્ય ન મળે તો અમે આ વધારાની સ્થિતિ મેળવી શકીએ છીએ જે કહે છે કે લિ વી વી છે અને અમે અત્યાર સુધી સ્થાપી નથી શક્યા પછી અમે સાચી સુધારણા મેળવી શકીએ છીએ અને અમે મેં કહ્યું. આ ખાતરી કરે છે કે POS હું માત્ર ખૂબ જ પ્રથમ વખત અમે પછી જોવા મળ્યું છે કે મૂલ્ય મળ્યું છે આ સુધારા ફરીથી કરવાથી અમને અટકાવે છે. તેથી, ઓછામાં ઓછા અમે યોગ્ય રીતે સ્થિતિનું પ્રથમ મૂલ્ય શોધી રહ્યા છીએ. અને છેલ્લે જો આપણે ક્યારેય તેને શોધી ન શકીએ તો પછી મળેલી કિંમત ક્યારેય સાચી નથી હોતી અને તેથી આપણે પાછો 1 નો અહેવાલ આપીએ. હવે આ મુદ્દો છે કે શા માટે આપણે સંપૂર્ણ યાદીમાંથી પસાર થઈ જવું પડશે, પછી ભલે આપણે વી.બી. 1 માં અમે સુરક્ષિત રીતે પાછા આવી શકીએ છીએ અને તે પોઝિશનની જાણ કરી શકીએ છીએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 04:32)

તેથી, વધુ કુદરતી વ્યૂહરચના નીચે સૂચવે છે; અમે સૂચિને સ્કેન કરીએ છીએ અને જલદી જ અમે મૂલ્ય શોધીએ છીએ ત્યાં સ્કેન બંધ કરીએ છીએ. તેથી, જ્યારે પણ આપણે પ્રથમ વખત મૂલ્ય શોધીએ છીએ ત્યારે અમે તે પદવી આપીએ છીએ અને અમે તે મેળવીએ છીએ. અલબત્ત, સ્કેન મૂલ્ય શોધી શકતું નથી, તેથી જો સ્કેન સફળ થયા વગર સમાપ્ત થઈ જાય તો અમે માઈનસ 1 ની જાણ કરીશું.

(સ્લાઇડ ટાઇ નો ઉલ્લેખ કરો: 04:58)

અમે 'ફો લૂપ' નો ઉપયોગ કરીને આ કરવાનું વિચારી શકીએ છીએ. તેથી, આપણે વેરિયેબલ x નો ઉપયોગ કરીને દરેક મૂલ્યમાંથી પસાર થઈએ છીએ. તેથી, x એ નામ છે જે પ્રથમ એલિમેન્ટ, સેકન્ડ એલિમેન્ટ દ્વારા જાય છે, આ સ્થાનો નથી, હવે આ વાસ્તવિક મૂલ્યો છે. અમે તપાસ કરીએ છીએ કે શું મૂલ્ય x કે જે અમે હાલમાં યાદીમાં પસંદ કરીએ છીએ તે ખરેખર મૂલ્ય છે જે આપણે શોધી રહ્યા છીએ, જો આપણે બહાર નીકળીએ છીએ અને અમે આ x ની સ્થિતિની જાણ કરીએ છીએ. જો આપણે લૂપના અંતમાં આવીએ અને આપણે એક્સને અત્યાર સુધી જેટલું નજરે જોયું તે પહેલાં આપણે માઈનસ 1. ની જાણ કરવી નહીં.

(ધીરે સમયનો ઉલ્લેખ કરો: 05:30)

અહીં તે કરવાના પ્રથમ પ્રયત્નો છે, તો ચાલો આપણે તેમાંથી પસાર થઈએ વ્યવસ્થિત રીતે સૌ પ્રથમ આપણી પાસે આ લૂપ છે, તેથી દરેક x ને સૂચિમાં જો x v છે તો અમે સ્થિતિની જાણ કરીએ છીએ. હવે આપણને પોઝિશન શોધવાનું છે કારણ કે આપણે મૂલ્યોથી પસાર થઈ રહ્યા છીએ તેથી આથી અમને પોઝિશન રેકૉર્ડ કરવા માટે નામ 1 નો ઉપયોગ કરવા માટે ફરજ પડી છે અને અમને મેન્યુઅલી આમ કરવાનું છે. તેથી, જ્યારે લૂપની જેમ આપણે પદ 0 થી શરૂ કરીએ છીએ અને પછી દરેક પુનરાવૃત્તિ આપણે સ્થિતિને વધારીએ છીએ. આ ફક્ત આ જ પ્રથમ વર્ઝન છે, આપણે જોશું કે એફ કેવી રીતે ix આ. તેથી, અમારે અલગ રાખવું પડે છે જ્યારે અમે આ માટે, રાખેલું રાખીએ છીએ, અલગ સ્થાનો પર નજર રાખીએ છીએ અને તેની જાણ કરીએ છીએ. પરંતુ નવું શું છે અને અહીં પ્રકાશિત થવાનું મુખ્ય મુદ્દો એ છે કે આ નિવેદનને બેક કહેવાય છે તો લૂપ બહાર નીકળે એટલે શું વિરામ થાય છે. તેથી આ બરાબર છે કે આપણે શું કરવા માગતો હતો, જો x એ વી પ્રથમ સ્થાન છે જે આપણે ચાલુ રાખવા માંગતા નથી, તો આપણે માત્ર તોડવું છે, જો નહીં તો આપણે સ્થિતિ વધારીશું અને પાછા જઈશું. હવે, આપણે અંતે કેવી રીતે રેકૉર્ડ કરી શકીએ છીએ કે હવે અમે આ વેરિયેબલ વેરીએબલ નથી. અંતમાં આપણે કેવી રીતે જાણીએ છીએ કે આપણે ખરેખર તેને જોયું છે? તેથી, પ્રશ્ન હું pos માટે pos છે અથવા તે હું માટે સુયોજિત થયેલ છે. ઠીક છે, ડિફોલ્ટ મૂલ્ય ઓછા છે 1. સપોઝ, રીઝ રીસેટ નથી, આપણે માઈનસ 1 ની જાણ કરવા માગીએ છીએ. તેથી, આપણા ફંક્શનમાં આપણે વાસ્તવમાં શરૂઆતમાં માપેડ 1 રાખવાનું સેટ કર્યું છે.

તેથી, મૂળભૂત મુલ્ય એ છે કે આપણે કરીએ છીએ શોધવા માટે અપેક્ષા નથી એવું કહીને એવું લાગે છે કે ખોટી છે. તેથી, મૂળભૂત રીતે પોઝિશન 1 ઓછા છે. આ લૂપના અંતમાં જો તમને મળ્યું ન હોય તો તે રીસેટ કરવામાં આવ્યું નથી તેથી તે બાદબાકી બાકી છે. બીજી બાજુ જો આપણે તેને બાકીના તમામ ઘટકોને જોઈ વગર જોયું છે. અમે તેને પ્રથમ સ્થાને મુક્યો છે અને અમે તેને લૂપમાંથી બહાર કાઢવા માટે બ્રેક સ્ટેટમેન્ટ લીધું છે જેથી અમે બિનજરૂરી રીતે સ્કેન કરતા નથી. જ્યારે આપણે કોઈ પણ રીતે અહીં આવીએ છીએ ત્યારે આપણે તેને પ્રથમ સ્થાને અહેવાલ આપીએ છીએ અથવા તે સૌથી ખરાબ કિસ્સામાં અમે સમગ્ર સૂચિમાં સ્કેન કર્યું છે, અને અમે તે કિસ્સામાં પ્રારંભિક મૂલ્ય બાદ 1 ત્યાં નથી મળ્યું. તેથી, કોઈ પણ સંજોગોમાં આપણે સ્થિતિ બદલી શકીએ છીએ અને અમારી પાસે કોઈ સમસ્યા નથી. શબ્દ બ્રેકના ઉપયોગને સમજાવવા માટે આ માત્ર છે, જે અમને અમુક પરિસ્થિતિઓમાં બહાર લાવવા માટે પરવાનગી આપે છે. હવે સૌથી ખરાબ કિસ્સામાં યાદ રાખો કે અમને તેમાં x નથી લાગતો. તેથી, આ લૂપનું સૌથી ખરાબ કેસ વર્તન એ વિરામ વગરની પરિસ્થિતિથી અલગ નથી કે આપણે સમગ્ર સૂચિમાં જવું પડે અને અમે ફક્ત ત્યારે જ બહાર આવવું જોઈએ જ્યારે અમે બધું સ્કેન કર્યું હોય, પરંતુ જો આપણે તેને શોધી કાઢીએ તો અમે કેટલીક બિનજરૂરી ગણતરીઓ

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 08:10)

આપણે આ સ્કેનિંગને બદલે પ્રથમ આને દૂર કરીને થોડુંક સરળ બનાવી શકીએ છીએ, વાસ્તવમાં સ્થિતિને સ્કેન કરવું વધુ સારું છે તેથી, તે કહેવું વધુ સારું છે કે POS ઓછા 1 છે, પરંતુ x માં ચાલ્યા જવાને બદલે તે શ્રેણી 0 માં પસાર થવું વધુ સારું છે. યાદ રાખો હવે આ 0 ની લંબાઈની લંબાઈ 0 છે. તેથી, આપણે જઈએ છીએ હું 0 થી લંબાઈની લંબાઈ માટે, તેથી જો આપણે 0 ન આપીશું તો તે માત્ર એક જ દલીલ આપશે જે તેને ઉપરની બાઉન્ડ તરીકે લેવામાં આવે છે. આ તમામ સ્થાનોમાંથી પસાર થશે. અને x ની ચકાસણીના બદલે અમે i ની સ્થાને i ની તપાસ કરીએ છીએ, તેથી જો લે પોઝિશન 1 v છે તો પછી આપણે આ વર્તમાન મુલ્ય ને સેટ કરીએ છીએ અને અમે ભંગ કરીએ છીએ. તેથી, વેરિએબલને બદલીને કે જે આપણે 'for' માં મુકીએ છીએ, આપણને થોડી વધુ કુદરતી કાર્ય મળ્યું છે. અને ફરી, આ હોશિયાર યુક્તિ છે જે કહે છે કે આપણે શરૂઆતમાં 1 થી ઓછા સ્થાને સુયોજિત કરીએ છીએ જો આપણે તે રીસેટ ન કર્યું હોય તો જો રેન્જ 0 થી n ની 1 ની કિંમતમાં કોઈ મુલ્ય ન હોય તો યાદી મુલ્ય જે આપણે જોઈ રહ્યા છીએ તે મૂલ્ય જેટલું છે. કારણ કે આપણે પહેલાની જેમ પાછો 1 પરત કરીશું. આ માટે આ હોશિયાર યુક્તિ જરૂરી છે. તેથી પ્રશ્ન એ છે કે જો અમારી પાસે એવી કોઈ પરિસ્થિતિ ન હોય કે જ્યાં આવા ચપલ યુક્તિ શક્ય હોય અથવા જો આપણે આ હોશિયાર યુક્તિ વિષે વિચારીશું નહીં તો આ સમયે આપણે કેવી રીતે જાણીશું. તેથી યાદ રાખો કે બે પરિસ્થિતિઓ ક્યાં તો આપણે ભાંગીએ છીએ, તેથી જો આપણે તેનો ભંગ કરીએ તો તેનો અર્થ એ છે કે મૂલ્ય મળ્યું છે, અથવા જો આપણે વિરામ ન કરીએ, જો આપણે સામાન્ય રીતે સમાપ્ત કરીએ, જો લૂપ બધી વસ્તુઓ દ્વારા જઈને અંત થાય તો વી નહીં. યાદ રાખો કે જો વી છેલ્લા સ્થાને જોવા મળે છે તો આપણે સૌપ્રથમ વિરામ કરીશું, આપણે પાછા જઈશું નહીં અને કહીશું કે લૂપ સમાપ્ત થાય છે. તેથી, જો આપણે શરૂઆતથી અંત સુધી કોઈપણ સ્થાને v જુઓ તો અમે વિરામનું નિવેદન ચલાવશું નહીં જો નહીં. પ્રશ્ન એ છે કે આપણે આ લૂપને ભાંગીએ કે નહીં કે આપણે તેને અલગથી સમાપ્ત કરીએ કે નહીં તે શોધી શકીએ છીએ.

(સ્લાઈડ સમયનો ઉલ્લેખ કરો: 10:06)

પાચથોન એવી કેટલીક ભાષાઓમાંની એક છે જે વાસ્તવમાં આ કરવા માટે માર્ગ પૂરો પાડે છે. તેથી તે કંઈક પરવાનગી આપે છે જે નામથી થોડું વિચિત્ર લાગે છે જે તેને બીજું કંઈક કહેવા આપે છે જે આપણે જોયું છે, તે લૂપ માટે બીજું પણ પરવાનગી આપે છે. વિચાર એ છે કે આ ભાગને જો એક્ઝિક્યુટ કરવામાં આવશે જો કોઈ વિરામ ન હોય તો લૂપ સામાન્ય રૂપે સમાપ્ત થાય છે, તેથી એ હકીકત વિશે ચિંતા ન કરો કે બીજું તેનો અર્થ એ નથી કે આ અંગ્રેજીમાં છે તેથી તે માત્ર અર્થસભર કરવાનો અથવા નવા શબ્દોની સંખ્યા છે તમારે ઉપયોગ કરવાની જરૂર છે જો તમે 'ફોર' સાથે કોઈ અન્ય

જોડાયેલા છે તો તે 'જ્યારે' સાથે જોડી શકાય છે, એનો અર્થ એ કે 'જ્યારે' અથવા 'માટે' કુદરતી રીતમાં સમાપ્ત થાય તો તે દરેક મૂલ્ય દ્વારા પુનરાવર્તિત કરવા માટે કે જે તેને પુનરાવર્તન કરવાની ધારણા હતી દ્વારા અથવા જ્યારે શરત ખોટી બની હતી. બીજી બાજુ બ્રેક સ્ટેટમેન્ટ દ્વારા રદ કરવામાં આવે તો બીજાને એક્ઝેક્યુટ કરવામાં નહીં આવે. હમણાં માટે અહીં, હવે અમે પ્રારંભ નથી અમારી પાસે આ હોશિયાર યુક્તિ નથી, અમારી પાસે હવે આ નથી. તેથી આપણે શું કહીએ છીએ કે આ શ્રેણી માટે હું તપાસ કરું છું અને જો તે ત્યાં છે તો આપણે વર્તમાનની સ્થિતિને સ્થાનાંતરિત કરીએ છીએ અને પછી આપણે ભંગ કરીએ છીએ. હવે, જો આપણે ખરેખર સમગ્ર સૂચિમાંથી પસાર થઈ ગયા છીએ અને તે મળ્યું નથી તો તે અવ્યાખ્યાયિત છે. જો પીઓપી અવ્યાખ્યાયિત હોય તો આપણે મૂલ્ય પરત કરીએ તે પહેલા તેને વ્યાખ્યાયિત કરવાની જરૂર છે, તેથી અમારી પાસે બીજું નિવેદન છે. હવે અમે કહીએ છીએ કે આ આખી વસ્તુ છે અને કોઈ બ્રેક નહીં, એલમાં નથી કારણ કે અન્યથા અમે બ્રેક કર્યું હોત અને અન્યથા પદ 0 થી n ની રેન્જમાં માન્ય મૂલ્ય પર સેટ કરવામાં આવશે. 1. તેથી, ત્યાં કોઈ વિરામ નથી ત્યાં કોઈ વી નથી. ચાલો આપણે કહીએ કે 1 બાદબાકી અને પછી તેને પાછો આપીએ.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 11:44)

સારાંશ માટે, અકાળેથી લૂપમાંથી બહાર નીકળી જવા માટે ઘણી વાર ઉપયોગી છે અમે તે માટે બન્ને માટે કરી શકીએ છીએ અને જ્યારે આપણે બ્રેક સ્ટેટમેન્ટ એક્ઝેક્યુટ કરી શકીએ છીએ અને તેના કુદરતી સમાપ્તિ પહેલાં લૂપમાંથી નીકળી જઈ શકીએ છીએ. અને કેટલીકવાર આપણે જાણવું મહત્વનું છે કે આપણે શા માટે તે સમાપ્ત કરીએ છીએ, આપણે તેને સમાપ્ત કરીએ છીએ કારણ કે લૂપ કુદરતી રીતે અંતમાં છે અથવા કારણ કે અમે બ્રેક સ્ટેટમેન્ટનો ઉપયોગ કર્યો છે. આ વૈકલ્પિક બીજું સપ્લાય કરીને કરી શકાય છે. બંને માટે, જ્યારે અને અંતમાં અન્યને પણ પરવાનગી આપે છે, અને અન્ય અંદરનું નિવેદન ફક્ત ત્યારે જ ચલાવવામાં આવે છે જ્યારે લૂપ સામાન્ય રીતે સમાપ્ત થાય છે

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રોફેસર. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 03 લેક્ચર - 04 એરેઝ વિ લિસ્ટ્સ, બાઈનરી શોધ

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 00:02)

પરિસ્થિતિઓમાં જ્યાં આપણે મૂલ્યોનો ક્રમ સંગ્રહિત કરવા માંગીએ છીએ. હવે તે એક પ્રોગ્રામમાં અથવા પ્રોગ્રામિંગ ભાષા અમલીકરણમાં જોવા મળે છે, ત્યાં બે મૂળભૂત રીત છે જેમાં આપણે આવા ક્રમને સંગ્રહિત કરી શકીએ છીએ. આને સામાન્ય રીતે એરેઝ અને યાદી તરીકે ઓળખવામાં આવે છે. તેથી, યાદી એરેઝ અને સૂચિ વચ્ચેના તફાવતને જોઈએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 00:22)

એરે સામાન્ય રીતે ક્રમ છે જે મેમરીમાં એક બ્લોક તરીકે સંગ્રહિત છે. તેથી, તમે કલ્પના કરી શકો છો કે તમે ઇચ્છો છો કે તમારી મેમરી ચોક્કસ રીતે ગોઠવવામાં આવે અને પછી તમારી પાસે એક એરે છે, તેથી સામાન્ય રીતે જે શબ્દો કહેવામાં આવે છે તે ગોઠવાય છે. શબ્દ તમે શું સંગ્રહિત કરી શકો છો અથવા મેમરીમાંથી પુનઃપ્રાપ્ત કરી શકો છો, અને એક એરે સામાન્ય રીતે કોઈ પણ અવકાશ વિના એક સતત અવરોધ હશે. અને, ખાસ કરીને આ ત્યારે લાગુ થશે જ્યારે એરેમાં ફક્ત એક પ્રકારનું મૂલ્ય હોય છે, તેથી ક્રમના તમામ તત્વો ક્યાં તો પૂર્ણાંક અથવા ફ્લોટ્સ અથવા કંઈક છે જ્યાં એરેના દરેક ઘટકની લંબાઈ સમાન કદ છે. અમે સામાન્ય રીતે આ બ્લોકમાં કેટલું મોટું છે તે અગાઉ કોઈ એરેમાં નથી. તેથી આપણે જાણીએ કે તે 100 એન્ટ્રી છે, તેથી અમારી પાસે 100 નું કદ છે. હવે જ્યારે આવું થાય છે, ત્યારે શું થાય છે જો તમે ક્રમાંકના જથ્થા એલિમેન્ટ અથવા અનુક્રમના i th એલિમેન્ટ જોવા માંગો છો, તો પછી તમે શું વિચારી શકો છો તે આ બ્લોકનું મેમરી 1, 2, 3 થી શરૂ થાય છે, હું જમણી તરફ અને તમે i th તત્વ ઝડપથી મેળવવા માંગો છો પરંતુ કારણ કે બધું એકસમાન કદનું છે અને તમે જાણો છો કે આ ક્યાંથી શરૂ થાય છે, અમે જાણીએ છીએ કે ક્રમ ક્યાંથી શરૂ થાય છે તમે માત્ર એક જ વખતના આ કદની ગણતરી કરી શકો છો અને ઝડપથી જઈ શકો છો અને મેમરીમાં સ્થાન પર એક શોટ જ્યાં i th તત્વ સાચવવામાં આવે છે. . તેથી, એક એરેના i th તત્વને એક્સેસ કરવાથી ફક્ત એરેના પ્રારંભિક બિંદુથી શરૂ કરીને અને i th પોઝિશન માટે i આગળ એકમોને પસાર કરીને સરનામાંના અંકગણિત ગણતરીની જરૂર છે. અને આ શું આપણે સતત સમય કોલ કરી શકે છે તે કરી શકાય છે. સતત સમય સુધી અમારો અર્થ શું છે તે વાસ્તવમાં હું પર આધાર રાખતો નથી. એરેનો છેલ્લો ભાગ મેળવવા માટે કોઈ સરળ અથવા મુશ્કેલ નથી, કારણ કે તે એરેના બીજા તત્વ પર વિચારવું છે, તે હું સ્વતંત્ર છે. તે કોઈપણ માટે y ની શ્રેણી મેળવવા માટે ચોક્કસ સમય લે છે. હવે, આનો એક પરિણામ એરે દાખલ કરવો અથવા કરાર કરવો ખર્ચાળ છે, કારણ કે હવે જો મારી પાસે 0 થી 99 સાથે એક એરે છે અને હું અહીં એક નવું મૂલ્ય ઉમેરવા માંગું છું તો તે સ્થાને કહીએ તો, આ તમામ એરેનો પ્રથમ 0 થી 100 અને હવે જે વસ્તુ પછી હું જે જગ્યાઓ સમાવવા માટે સ્થળાંતરિત કરું તે પછી જો આપણે સમગ્ર એરે સાથે સમાન પ્રતિનિધિત્વ રાખવા ઇચ્છતા હોઈએ તો એક બ્લોક તરીકે સંગ્રહિત થાય છે. તેથી, જ્યારે અમારી પાસે મેમરીનો એક બ્લોક હોય છે, તે તેનો કોઈ પણ ભાગ મેળવવા માટે કાર્યક્ષમ છે, તે ઝડપથી વિસ્તૃત કરવા માટે તે ખૂબ જ કાર્યક્ષમ નથી કારણ કે અમારે પછી બધું જ શિફ્ટ કરવું પડશે. ઉદાહરણ તરીકે સૌથી ખરાબ કેસ, જો આ ગ્રીન બ્લોક 0 મા સ્થાને આવે તો સમગ્ર એરેને એક પદ દ્વારા ખસેડવામાં આવે છે. તે જ રીતે સંકોચન પણ મોંઘું છે કારણ કે અમને કેટલાક અર્થમાં એક છિદ્ર બનાવવું પડશે. જો આપણે આ ઘટકને દૂર કરીએ તો આપણી પાસે અહીં એક છિદ્ર છે અને પછી આપણે આ છિદ્રને અવરોધિત કરવા બધું ઉપર દબાણ

કરવું પડશે, કારણ કે - યાદ રાખો કે એરેમાં તમામ ઘટકો સંલગ્ન હોવા જોઈએ જે પ્રારંભિક સ્થિતિથી શરૂ થતા કોઈ અવકાશ વિના છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 03:27)

ક્રમ જાળવવાનો બીજો રસ્તો એ છે કે તે એક સમયે એક ઘટક સંગ્રહિત કરે છે અને મેમરીમાં એકબીજાના સંબંધમાં કેવી રીતે આ ઘટકો છે તે અંગે ચિંતા ન કરો. હું આ મેમરીને મોટી જગ્યા તરીકે વિચારી શકું છું અને હવે મારી પાસે અહીં એક તત્વ હોઈ શકે છે, તેથી આ મારું પહેલું તત્વ છે અને પછી મને એવું કહેવાનો રસ્તો મળશે કે અહીંથી આગળનું તત્વ બીજે ક્યાંક છે, આ આપણે કહીએ છીએ લિંક તેથી ઘણી વખત અમલીકરણમાં આને કડી થયેલ સૂચિ કહેવાય છે, તેથી મારી પાસે અહીં પ્રથમ ઘટક હોઈ શકે છે. હવે વિવિધ કારણોને લીધે હું બીજું તત્વ અહીં અને તેથી આગળ મૂકી શકું છું. તમે કલ્પના કરી શકો છો કે જો તમારી પાસે તમારા કબાટમાં કંઈક કહેવું હોય અને પછી તમે વસ્તુઓ લઈ જાઓ છો અને પછી તમે વસ્તુઓને પાછું મૂકી દો છો પરંતુ તમે વસ્તુઓને પ્રથમ સ્થાને મૂકો છો જ્યાં તમારી પાસે ખાલી સ્લોટ છે, પછી તમે જે ક્રમાંકમાં મૂકી છે તે પાછો વિચારે છે અનુક્રમનો આદર ન કરી શકે કે જેમાં તેઓ શેલ્ડમાં છેલ્લે દેખાય છે. તેથી, અહીં તે જ રીતે આપણે કેવી રીતે આ તત્વો સંગ્રહિત કરવામાં આવ્યા છે તે વિશે કોઈ ભૌતિક ધારણા નથી, અમારી પાસે માત્ર પ્રથમ ઘટકથી આગામી ઘટક સુધી લોજિકલ લિંક છે અને તેથી આગળ. આનો બીજો ભાગ એ છે કે આપણે સૂચિની એકંદર લંબાઈ અંગે ચિંતા કરવાની જરૂર નથી કારણ કે અમે જાણીએ છીએ કે અમે 0 મા સ્થાને શરૂ કર્યું છે અને અમે નીચે વૉકિંગ ચાલુ રાખીએ છીએ. છેલ્લી પોઝિશન પર એમ કહીએ કે છેલ્લા સ્થાને હકીકત બે છે તો અહીં એવું કોઈ સંકેત હશે કે આગળ કોઈ તત્વ નથી, તેથી બે છેલ્લો ભાગ છે. એક સૂચિ લવચીક કદ હોઈ શકે છે અને દેખીતી રીતે કારણ કે આપણે એક ઘટકને બીજામાં જ નિર્દેશ આપી રહ્યા છીએ, અમે પાચથોનમાં જે પણ જોઈ શકીએ તે પણ સમાવી શકીએ છીએ જ્યાં સૂચિની દરેક ઘટક કદાચ અલગ પ્રકારની હોઈ શકે છે અને તેથી પ્રત્યેક મૂલ્યમાં અલગ અલગ કદ હોઈ શકે છે. એરેની વિપરીત મહત્વની નથી કે બધા મૂલ્યો બરાબર એ જ કદ છે કારણ કે આપણે ગણતરી કરવા માગીએ છીએ કે કેટલા પ્રમાણમાં ith ઘટક મેળવવા માટે અવગણો. અહીં, અમે અવગણીએ છીએ કે અમે ફક્ત આ લિંક્સ વટાવી રહ્યા છીએ. કારણ કે આપણે આ લિંક્સને શોધવાનો એકમાત્ર રસ્તો છે કે જ્યાં ith તત્વ એ 0 મી તત્વમાંથી શરૂ કરવાનું છે અને પછી પ્રથમ તત્વ પર જાઓ પછી બીજા તત્વ પર જાઓ અને તેથી આગળ, કારણ કે પ્રાધાન્ય આપણી પાસે કોઈ ખ્યાલ નથી ith તત્વ છે. તેથી, હું પગલાં પછી અમે ith તત્વ સુધી પહોંચી જશે. અને જો અમારી પાસે મોટા મૂલ્ય છે તો મને ત્યાં રહેવાની સમય લાગે છે. તેથી અનુક્રમમાં ith પોઝિશનનો ઉપયોગ કરીને જ્યારે શ્રેણીને સૂચિ તરીકે સંગ્રહિત કરવામાં આવે છે ત્યારે હું સમયની પ્રમાણસર મેળવે છે, અમે એમ ધારી શકતા નથી કે આપણે એરેમાં વિપરીત સતત સૂચિમાં સૂચિમાં કોઈપણ સ્થાન સુધી પહોંચી શકીએ છીએ. બીજી બાજુ તે આમાંની એક સૂચિમાં તત્વ શામેલ કરવું અથવા કાઢી નાખવું પ્રમાણમાં સરળ છે. ધારો કે, અમારી પાસે આ જેવી સૂચિ છે. ધારો કે આપણે 0 મા સ્થાનેથી શરૂ કરીએ છીએ અને ith પદ પર આવી શકીએ છીએ અને હાલમાં જો આપણે કહીએ છીએ કે ith પોઝિશન i વત્તા 1 થ સ્થાને નિર્દેશ કરે છે જે બાકીના તરફ નિર્દેશ કરે છે, અને ધારો કે આપણે અહીં કંઈક દાખલ કરવા માંગો છો, તો તે ખૂબ સરળ છે આપણે કહીએ છીએ કે આ નવો આઈ વત્તા 1 મી સ્થાને છે. આપણે આ મુલ્ય સંગ્રહવા માટે મેમરીમાં એક નવું બ્લોક બનાવીએ અને પછી આપણે આ બિંદુ અહીં બનાવીશું. તેથી, તે પ્લમ્બિંગની જેમ છે, અમે એક પાઇપ દૂર કરીએ છીએ અને અમે ith એલિમેન્ટમાંથી નવા ઘટક સાથે એક પાઇપને જોડીએ છીએ અને અન્ય પાઇપને નવા તત્વ સાથે જોડીએ જે અગાઉ ith તત્વની બહાર હતું. અમારે આ ત્રણ લિંક્સને આજુબાજુ પાછી ખસેડવી પડશે અને જ્યાં પણ અમારે આવું કરવાનું છે ત્યાં કોઈ ફરક નહીં પડે, સૂચિમાં કોઈપણ જગ્યા જો અમારી પાસે હોય, તો આ લિંક્સમાં આ સ્થાનિક પરિવર્તન કરવું પડશે. અને તેથી આ નિવેશ હવે સતત સમયની ક્રિયા બની જાય છે જો આપણે પહેલેથી જ તે સ્થાને હોવ જ્યાં આપણે ફેરફાર કરવા માગીએ છીએ. તે જ રીતે જો આપણે એવી વસ્તુ કાઢી નાખવા માંગીએ છીએ જે હકીકતમાં પણ સરળ છે તો તે વધુ સરળ છે. તો મેં કહ્યું છે કે હું 1 વત્તા 1 તરફ ઇશ્યુ કરું છું અને વત્તા 2 તરફ ઇશારો કરું છું અને હું આ તત્વ દૂર કરવા માંગું છું, તો પછી હું આ લિંક્સ

સીધું જ આગામી એક પર લઈ જઈશ. યાદ રાખો કે આ લિંક્સ અમારા માટે ઉપલબ્ધ છે, અમે આ લિંકને જાણીએ છીએ, આ લિંક આપણે જાણીએ છીએ, તેથી અમે જાણીએ છીએ કે હું ક્યાં વત્તા બીજો એલિમેન્ટ છે. એ જ રીતે, જ્યારે આપણે એક નવું તત્વ બનાવવું હોય ત્યારે આપણે તેના માટે એક લિંક મેળવીએ છીએ કારણ કે અમે તેને બનાવીએ છીએ અને આપણે જાણીએ છીએ કે અહીં કડી કડી છે કારણ કે અહીં પહેલેથી જ અહીં છે. તેથી આપણે તેને *ith* તત્વથી નવા તત્વમાં નકલ કરી શકીએ છીએ. તેથી, સૂચિમાં તે *ith* તત્વ મેળવવા માટે ખર્ચાળ છે, જે આપણે મેળવવા માટે પ્રયાસ કરી રહ્યા છીએ તે સ્થિતિને પ્રમાણસર ગણાય છે, જો કે, તે સ્થિતિમાં તત્વને દાખલ કરવા અથવા કાઢી નાખવા માટે સ્થિતિ સતત સમયની છે. એરેથી વિપરીત, જો આપણે કોઈ સ્થાન પર શામેલ અથવા કાઢી નાંખો તો આપણે ઘણા મૂલ્ય આગળ અથવા પાછળ શબ્દો પાળી શકીએ છીએ અને તે સમય લે છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 07:47)

ચાલો સામાન્ય ઓપરેશન્સ જોઈએ જે આપણે સિક્વન્સ પર કરીએ છીએ. તેથી એક સામાન્ય ઓપરેશન, હવે જો હું માત્ર સિક્વન્સને વધુ અનુકૂળ રીતે સિક્વન્સ તરીકે દર્શાવું છું તો આપણે તેને ચિત્રકામ કરી રહ્યા છીએ. ધારો કે, હું *i* અને *j* પર મૂલ્યોનું વિનિમય કરવા માંગુ છું. આ એરેમાં સતત સમય લેશે કારણ કે આપણે જાણીએ છીએ કે આપણે *ith* પોઝિશન પર મૂલ્ય મેળવી શકીએ છીએ, *i* અને *j* ના સ્વતંત્ર સમયે સ્થિર સ્થાને જેથ પદ પર મૂલ્ય મેળવીએ છીએ અને પછી અમે તેમને અદલાબદલ કરીએ છીએ તેમાં ફક્ત આનો સામનો કરવો પડે છે અને અન્ય એક પાછા. બીજી બાજુ યાદીમાં મને પ્રથમ *ith* પોઝિશન નીચે જવું પડશે અને પછી બે પોઝિશન્સ મેળવવા માટે જથ પોઝિશન નીચે જવું પડશે જેથી મારી પાસે યાદીમાં હશે અને મને લિંક્સનો શ્રેય હશે અને પછી બીજી પાસે લિંક્સનો ક્રમ ત્યારબાદ હવે બ્લોક ઓળખાય છે જ્યાં *ith* મૂલ્ય છે અને બ્લોક જ્યાં જથ મૂલ્યો પછી હું મામૂલી પરિવર્તન બદલ્યા વગર તેમને વિનિમય કરી શકું છું, હું ફક્ત મૂલ્યોને પાછળની બાજુએ નકલ કરું છું, પરંતુ *ith* અને જથ મૂલ્યોને શોધવા માટે સમય લાગે છે *i* અને *j* ની પ્રમાણસર, તેથી તે રેખીય સમય લે છે. બીજી બાજુ આપણે પહેલેથી જ જોયું છે, જો તમે પોઝિશન પર મૂલ્ય કાઢી નાખવા માગો છો અથવા પોઝિશન પછી મૂલ્ય દાખલ કરો છો તો આ યાદીમાં કાર્યક્ષમ રીતે કરી શકીએ છીએ કારણ કે આપણે ફક્ત કેટલાક લિંક્સને આસપાસ ખસેડવાની જરૂર છે, જ્યારે એરેમાં આપણે વસ્તુના પહેલાં અથવા પછીના કેટલાક મૂલ્યોના મોટા જથ્થામાં કેટલાક સ્થળાંતર કરવું પડશે અને તે માટે અમને સમયસર પ્રમાણમાં લેવાની જરૂર છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 09:12)

એરે અને સૂચિ તરીકે ક્રમના બે રજૂઆત વચ્ચેના આ તફાવતોનો પરિણામ એ છે કે કેવી રીતે એલ્ગોરિધમ્સ કે જે અમે કેવી રીતે અનુક્રમ માટે ડિઝાઇન કરવા માંગીએ તે વિશે સાવચેત રહો. ક્રમ ખરેખર રજૂ થાય છે એક એલ્ગારિધમ જે સૂચિ માટે કાર્યક્ષમ રીતે કામ કરે છે તે કદાચ એરે અને ઊલટી માટે કાર્યક્ષમ રીતે કાર્ય કરી શકે નહીં. આને સમજાવવા માટે, ચાલો આપણે એવી કોઈ વસ્તુ પર નજર કરીએ જે તમે ઓછામાં ઓછા અનૌપચારિક રીતે કહેવાય બાઈનરી શોધ સાથે પરિચિત છો.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 09:42)

જે સમસ્યા આપણને રસ છે તે જાણવા માટે એ છે કે મૂલ્ય *v* સંગ્રહમાં હાજર છે કે પછી આપણે તેને અનુક્રમે કોલ કરી શકીએ છીએ જે શ્રેણીને આપણે સીક કહીએ છીએ. તેથી, આપણી પાસે મૂલ્યોનો ક્રમ છે, જે આપણે તપાસવું છે કે આપેલ મૂલ્ય છે કે નહીં. દાખલા તરીકે, અમે એવા લોકોની રોલ નંબરની યાદી જોઈ રહ્યા હોઈએ છીએ કે જેઓને તમે પ્રોગ્રામ માટે પસંદ કરવામાં આવ્યા છે કે કેમ તે તપાસો કે અમારું રોલ નંબર છે કે નહીં. બે પ્રશ્નો છે જે આપણે પૂછીએ છીએ; એક એ મહત્વનું છે કે શું ક્રમ એરે તરીકે અથવા યાદી તરીકે જાળવવામાં આવે છે અને તે એ મહત્વનું પણ છે કે તેને એરે અથવા એક સૂચિ

તરીકે જાળવવામાં આવે છે કે નહીં તે કેટલીક વધારાની માહિતી છે જે આપણે જાણીએ છીએ, તે માટે ઉપયોગી છે એરે ચડતા ક્રમમાં સોર્ટ કરવામાં આવે છે, જે તમામ ઘટકો શરૂઆતથી અંત સુધી સચોટ રીતે એક ક્રમમાં જાય છે, સૌથી નીચલો સૌથી વધુ, અથવા સૌથી વધુ નીચો છે, અથવા તે વાંધો છે, તે કોઈ બાબત નથી.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 10:37)

અહીં એક ક્રમમાંગોળાં અનુક્રમમાં મૂલ્ય શોધવા માટેનો એક ખૂબ જ સરળ પાચથોન પ્રોગ્રામ છે. આ તે પહેલાંની સમાન છે જે આપણે પહેલાં જોયું છે કે જ્યાં આપણે અનુક્રમમાં મૂલ્યની પ્રથમ સ્થિતિની સ્થિતિ શોધી રહ્યા છીએ, જે અમે નથી હોતી, પણ પોઝિશનની જરૂર નથી, આપણે ફક્ત સાચી કે ખોટા જરૂર છે, તે છે કે છે તે નથી, તે ખૂબ સરળ વસ્તુ છે આપણે શું કરીએ તે આપણે ક્રમના તમામ ઘટકો દ્વારા લૂપ કરીએ છીએ અને તપાસ કરીએ છીએ કે શું કોઈપણ મૂલ્ય તે મૂલ્ય છે જે આપણે શોધી રહ્યા છીએ. એકવાર આપણે તેને શોધી લીધા પછી આપણે બહાર નીકળી શકીએ છીએ, તેથી આ મૂલ્ય સાચી સાથે ફંક્શનમાંથી બહાર નીકળે છે. અને જો આપણે સમગ્ર સૂચિમાંથી પસાર થવામાં સફળ થયા હોય, પરંતુ અમે સાચાથી બહાર ન આવ્યા હોય તો એનો અર્થ એ છે કે અમને આ વસ્તુ મળી નથી, તેથી અમે તે પછી અસ્પષ્ટપણે કહી શકીએ છીએ કે અમે આ બિંદુએ પહોંચી ગયા છીએ તો આપણે મૂલ્ય v ને મળ્યું નથી. અમે શોધી રહ્યા છે અને તેથી અમે ખોટા પાછા જોઈએ. કારણ કે આપણે પોઝિશન શોધી રહ્યા નથી તો આપણી પાસે ખૂબ સરળ કોડ છે, જો તમે પાછા જાઓ અને કોડને અમે શોધી કાઢો, તો આપણે ત્યાં પહેલેથી જ પોતાનું સ્થાન જાળવી રાખવું જોઈએ અને મૂલ્યની જગ્યાએ મૂલ્ય તપાસવું જોઈએ. અને બીજું, જ્યારે આપણે લૂપ પૂરું કરીએ છીએ ત્યારે આપણે નક્કી કર્યું હતું કે આપણે તેને શોધી લીધું છે કે નહીં પણ અમને તે મળ્યું નથી, શું યાદ છે કે આપણે પ્રથમ વખત લૂપની બહાર નીકળી જવા માટે બ્રેકનો ઉપયોગ કરીએ છીએ. અમે શોધી કાઢીએ કે અમે તૂટી ગયા કે નહીં, જો અમારી પાસે વિરામ ન હોય તો અમે તેને શોધી લીધું, જો અમારી પાસે બ્રેક ન હોય તો અમને તે મળ્યું નથી. તદનુસાર, વિરામની કિંમત નક્કી કરવામાં આવી હતી અથવા તે સેટ કરી ન હતી અને જો તે સેટ ન હોય તો આપણે તેને 1 ઓછા કરવી જોઈએ. તેથી તે વધુ જટિલ છે, આ ખૂબ સરળ છે.

(સ્લાઇડ સમય જણાવો: 12:07)

આ કાર્યનો મુખ્ય મુદ્દો એ છે કે શરૂઆતથી અંત સુધી સ્કેન કરતા અન્ય શોધનો અમારો કોઈ ઉકેલ નથી. વી શોધવાનો એકમાત્ર વ્યવસ્થિત માર્ગ એ શરૂઆતમાં શરૂ થાય છે કે શરૂઆતમાં જ શરૂ કરવું અને અંત સુધી જવું અને દરેક મૂલ્યને ચકાસવું કારણ કે અમારી પાસે કોઈ કિંમત નથી જ્યાં આ મૂલ્ય હોઈ શકે. આ ક્રમની લંબાઈના પ્રમાણમાં સામાન્ય રીતે સમય લેશે. અમે સામાન્ય રીતે આ કાર્યને સૌથી ખરાબ કિસ્સામાં કેટલો સમય લેશે તે અંગે રસ ધરાવીએ છીએ. તો સૌથી ખરાબ કેસ શું છે? ઠીક છે, કારણ કે એક ખરાબ કેસ એ છે કે જો આપણે યાદીના અંતે મૂલ્ય શોધીએ તો. તેથી, વી એ છેલ્લો મૂલ્ય છે, તો આપણે બધાને જોવાનું છે. પરંતુ વધુ સામાન્ય રીતે વી યાદીમાં નથી. વી એ સૂચિમાં નથી કે આપણે નક્કી કરી શકીએ છીએ કે વી એ સૂચિમાં નથી, દરેક મૂલ્ય ચકાસવા અને તે મૂલ્ય મળ્યું નથી તે નક્કી કરવાનું છે. અને આ ગુણધર્મ કે જેના માટે આપણે આખા ક્રમને સ્કેન કરવું પડે છે અને તેથી આપણે ક્રમ અનુક્રમે સમય કાઢવો તે નક્કી કરવા માટે છે કે શું વી ક્રમ છે કે નહીં તે અનુક્રમ એક એરે છે કે સૂચિ છે, પછી ભલે તે એક છે એરે અથવા સૂચિ છે કે જેમાં આપણે દરેક મૂલ્યથી માહિતીને લગતી સંસ્થાઓ દ્વારા ભરી નથી શકતા શું બાબતો એ હકીકત છે કે માહિતી માટે કોઈ વધારાના માળખું નથી, માહિતી કોઈ પણ રીતે કોઈ પણ રીતે સોર્ટ કરવામાં આવી નથી, આપણે કોઈ પણ સમયે આપી શકીએ છીએ અને કહી શકીએ છીએ કે અત્યાર સુધી અમે તેને જોયું નથી તેથી અમે તેને પાછળથી જોઈ શકતા નથી.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 13:26)

બીજી બાજુ, જો અમારી પાસે અનુક્રમે ક્રમ હોય તો અમારી પાસે એક પ્રક્રિયા છે જે ઓછામાં ઓછા તમારી સાથે અનૌપચારિક રીતે પરિચિત હશે. ઉદાહરણ તરીકે, જ્યારે આપણે શબ્દકોષમાં કોઈ શબ્દની શોધ કરીએ છીએ ત્યારે, શબ્દકોષ શબ્દોની મૂળાક્ષરે ક્રમમાં ગોઠવવામાં આવે છે. જો આપણે કોઈ શબ્દ શોધી રહ્યા છીએ અને જો આપણે રેન્ડમ પર પૃષ્ઠ ખોલીએ છીએ, તો આપણે શબ્દ વાંદરા માટે જોઈ રહ્યા છીએ અને અમે એક શબ્દ પર શબ્દકોષ ખોલો કે જ્યાં મૂલ્યો અથવા શબ્દ મેં સાથે શરૂ કરે છે, પછી આપણે જાણીએ છીએ કે હું મીટર પછી આવું છું. ઇંગ્લીશ મૂળાક્ષરની શબ્દકોષ ક્રમમાં. તેથી, અમારે ફક્ત શબ્દકોષના બીજા અર્ધમાં જ શોધ કરવાની જરૂર છે, હું તે પહેલા કોઈ પણ શબ્દ જોવાની જરૂર નથી. સામાન્ય રીતે જો અમારી પાસે અનુક્રમ છે કે જે આ મૂલ્ય શોધવાનો કાર્યક્ષમ માર્ગ છે તે પહેલા મધ્યમ મૂલ્યને જોવું છે, તેથી અમે v માટે જોઈ રહ્યા છીએ, તેથી અમે તપાસ કરીએ છીએ કે અહીં શું થાય છે. તેથી, ત્રણ કેસો ક્યાં છે તે કઈ રીતે આપણે સારા છીએ, જો આપણે તેને મળ્યું ન હોય તો આપણે જે મૂલ્ય શોધીએ છીએ તેની સરખામણી કરીએ છીએ. જો અમે જે મૂલ્ય શોધી રહ્યાં છીએ તે મૂલ્ય કરતાં ઓછું છે જે આપણે ત્યાં જોયું છે, તે આ અર્ધમાં હોવું જોઈએ. બીજી તરફ જો આપણે જે મૂલ્ય શોધીએ છીએ તે મોટું છે, તે આ અર્ધમાં જ હોવું જોઈએ. તેથી આપણે શોધવા માટે જગ્યાની માત્રાને અડધા કરી શકીએ છીએ અને આપણે ખાતરી કરી શકીએ છીએ કે અર્ધ આપણે હકારાત્મક નજરમાં નથી જઈ રહ્યા કારણ કે મૂલ્ય નથી કારણ કે અમે એમ ધારી રહ્યા છીએ કે આ ક્રમ સોર્ટ થાય છે. તેને બાઈનરી શોધ કહેવામાં આવે છે. આ ઉદાહરણ માટે પણ તમે શું કરો જ્યારે તમે વીસ પ્રશ્નો જેવી રમત રમે છે, જો તમે તે રમી રહ્યા છો જ્યારે કોઈ વ્યક્તિ તમને કોઈ વ્યક્તિનું નામ અનુમાન કરવા માટે પૂછે છે જે પછી તેઓ વિચારી રહ્યા હોય તો તમે પ્રથમ પ્રશ્ન પૂછો કે તે સ્ત્રી સ્ત્રી છે, જો વ્યક્તિ માદા પછી વ્યક્તિઓ છે અને તેમનું જવાબ હા છે, તો તમારે ફક્ત સ્ત્રીઓ વિશે વિચારવું જ જોઈએ, જો વ્યક્તિ કહે તો પછી તમારે માત્ર એમ વિશે વિચારવું જ પડશે, તેથી અમારી પાસે પુરૂષો છે. તેથી, આપની કલ્પનામાં અડધોઅડધ લોકો છે જેના વિશે આપણે વિચારવું પડશે. દરેક મુદ્દા પર, દરેક પ્રશ્નનો જવાબ પછી - હા કે ના.

(સ્લાઇડ સમય જણાવો: 15:22)

બાઈનરી શોધ માટે અહીં કેટલાક પાયથોન કોડ છે. તેથી, સામાન્ય રીતે દ્વિસંગી શોધ સંપૂર્ણ સૂચિથી શરૂ થશે અને પછી આપણે કહ્યું હતું કે તે મિડપોઇન્ટને જોશે અને ડાબી બાજુ પર નિર્ણય કરશે, તેથી આપણે ફરીથી આ પર દ્વિસંગી શોધ કરવી પડશે. અમે તે કેવી રીતે કરશે? ફરી આપણે આ ભાગની મિડપોઇન્ટ જોશું તો આપણે ફરીથી આગળના ભાગની મિડપોઇન્ટ કહીએ છીએ જે આપણે જોઈ શકીએ છીએ. સામાન્ય દ્વિસંગી શોધમાં સૂચિના અમુક સેગમેન્ટમાં મૂલ્ય માટે બાઈનરી શોધ કરવાનો પ્રયાસ કરવામાં આવે છે. તેથી આપણે l અને r નો ઉપયોગ કરીને તે સેગમેન્ટને સીમાંકિત કરીશું. તેથી, અમે આ સ્લાઇડ ક્રમ શોધી રહ્યા છીએ જે l થી શરૂ થાય છે અને r ઓછા 1 પર જઈને, અમે એમ માનીએ છીએ કે ક્રમ સોર્ટ થાય છે અને અમે ત્યાં v માટે જોઈ રહ્યા છીએ. પ્રથમ તો સ્લાઇડ ખાલી છે, તેથી આ કહે છે કે સ્લાઇડ ખાલી છે જે આપણે આ વસ્તુને અડકે છે અને અમે છેવટે જોવા માટે મૂલ્યોમાંથી નીકળી ગયા છીએ. છેલ્લા વસ્તુ જે આપણે જોયેલી હતી તે લંબાઈ 1 નું સ્લાઇડ હતું અને અમે તેને 2 માં વિભાજિત કરી દીધું અને આપણને એક લંબાઈ 0 નો સ્લાઇડ પસંદ કરવામાં આવ્યો. પછી આપણે કહી શકીએ કે તે હજુ સુધી મળ્યું નથી, તેથી અમે તેને ક્યારેય શોધી શકતા નથી. અને અમે ખોટા ફેરવીએ છીએ. બીજી તરફ જો સ્લાઇડ ખાલી ન હોય તો, આપણે શું કરીએ તે પહેલા આપણે મિડપોઇન્ટની ગણતરી કરીએ છીએ. બિંદુની ગણતરી કરવા માટે એક સરળ રીત છે આ પૂર્ણાંક ડિવિઝન કામગીરીનો ઉપયોગ કરવો. ધારીએ કે, હવે આપણે 4 થી 7 ના સ્લાઇડ સુધી આગળ વધીએ છીએ અને આગળના તબક્કે આપણે 11 થી 2 પૂર્ણાંક મુજબની ગણતરી કરીશું અને આપણે 5 પર જઈશું. $4, 5, 6, 7$. યાદ રાખો. આપણે ક્યાં 6 કે 7 પછી પસંદ કરી શકીએ? તેને બે ભાગોમાં વિભાજિત કરો, કારણ કે આપણે 6 ની તપાસ કરવા જઈ રહ્યા છીએ અને પછી $4, 5$ અને 7 જુઓ અથવા 5 અને પછી $4, 7$ જુઓ. જો આપણે પૂર્ણાંક ડિવિઝન કરીએ તો આપણે નાના આઉટપુટ પસંદ કરીશું. તેથી, આપણે મિડપોઇન્ટ શોધીએ છીએ. હવે અમે તપાસ કરીએ છીએ કે મૂલ્ય તે મિડપોઇન્ટ પર મૂલ્ય છે કે નહીં તે જો આપણે સાચું પાછી મેળવીએ, જો તે ન હોય તો અમે તપાસ કરીએ છીએ કે નાનું, જો આપણે વર્તમાન ડાબી બાજુથી મિડપોઇન્ટની ડાબે સુધી અમારી

શોધ ચાલુ રાખીશું. હવે આપણે આ પાયથોન નો ઉપયોગ કરી રહ્યા છીએ, એવું વિચારીએ કે આનો મતલબ છે કે આ મધ્યથી એક સ્લાઇસ છે અને તેથી તે મધ્યભાગ 1 પર અટકી જાય છે. તેથી, તે ફરીથી મૂલ્યની તપાસ કરતી નથી. તે સખત તેની ડાબી બાજુએ બધું જ જોશે જો આપણે જે મૂલ્ય શોધીએ છીએ તે મિડપોઇન્ટ સાથેનું મૂલ્ય નથી અને તે મિડપોઇન્ટ કરતા નાનું છે, ડાબી તરફ જુઓ, અન્યથા તમે સખત રીતે જમણી તરફ જુઓ છો, તમે મિડ વત્તા એકથી શરૂ કરો છો અને વર્તમાન જમણી લીટી ઉપર જાઓ છો. આ ફરી યાદ આવવું કાર્ય છે તે દરેક તબક્કે આ કરવાનું ચાલુ રાખશે અંતરાલ અડધા છે, તેથી આખરે ધારણા છે કે અમારી પાસે ફોર્મનું એક સ્લાઇસ માત્ર એક મૂલ્ય છે, તેથી 5 થી 6 ઉદાહરણ તરીકે, પછી આગામી બિંદુએ જ અમે ફક્ત એક 5 થી 5 અથવા 6 થી 6 સુધી સ્લાઇસ કરો અને આ આપણને એક સ્લાઇસ આપે છે જે ખાલી છે કારણ કે આપણે ડાબી બાજુએ જમણી બિંદુ પર મળશે તે જ છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 18:08)

તો, ટ્રિસંગી શોધ અલ્ગોરિધમનો કેટલો સમય લે છે? મુખ્ય મુદ્દો એ છે કે દરેક પગલા અંતરાલને અડકે છે જે અમે શોધી રહ્યા છીએ અને જો અમારી પાસે ખાલી અંતરાલ હોય તો અમે તાત્કાલિક જવાબ મેળવીએ છીએ. તેથી, આપણે જે સામાન્ય રીતે કરીએ છીએ તે ક્રમ અથવા એરે અથવા સૂચિના કદ પર આધારીત સમય રેકોર્ડ કરવા માટે છે, તેથી અમે અહીં એરે લખ્યું છે, પરંતુ તે સામાન્ય રીતે ક્રમ હશે. જો શ્રેણીની લંબાઈ 0 હોય તો તે માત્ર એક જ પગલું લે છે કારણ કે અમે ફક્ત જાણ કરીએ છીએ કે તે ખોટી છે, જો કોઈ ઘટકો બાકી નથી તો અમે તેને શોધી શકતા નથી. નહિતર, આપણે મિડપોઇન્ટનું પરીક્ષણ કરવું પડશે, જેથી એક અમૂર્ત પગલા લઈએ તો તમે મિડપોઇન્ટની ગણતરી કરી શકો છો અને તપાસો કે મૂલ્ય શું છે તે આપણે એક અમૂર્ત પગલામાં તૂટી જઈશું. અને પછી જવાબ પર આધાર રાખીને, યાદ રાખો કે અમે ખરાબ કિસ્સામાં કમ્પ્યુટિંગ છીએ, જ્યારે તે સૌથી ખરાબ કેસમાં જવાબ છે જ્યારે તે ક્રમમાં જોવા મળે છે. તેથી, સૌથી ખરાબ કેસ તે મિડપોઇન્ટ નહીં હોય તો આપણે અડધા ક્રમ જોવો પડશે. અમે ફરીથી નવી સૂચિ માટે ટ્રિસંગી શોધ હટાવવી પડશે જે જૂની સૂચિની અડધી લંબાઈ છે, તેથી n તત્વો માટે લેવામાં આવતો સમય 1 વત્તા 2 ઘટકો દ્વારા n માટે લેવામાં સમય છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 19:18)

અમે ટીના n માટે એક અભિવ્યક્તિ જોઈએ છે જે સંતોષ કરે છે, તેથી તે સામાન્ય રીતે પુનરાવૃત્તિ તરીકે ઓળખાય છે. એન કાર્યરત શું આ સંતુષ્ટ કરશે? તે કરવા માટેનો એક રસ્તો ફક્ત અવેજીમાં જ રહે છે અને શું થાય છે તે જુઓ. આપણે પોતાની રીતે unwinding શરૂ, તેથી, અમે તે જ પુનરાવૃત્તિ દ્વારા છે 1 વત્તા T એ n દ્વારા 4, કારણ કે હું આ લે છે અને તેને અડધા. તેથી, ટી નું n એ 1 વત્તા 1 વત્તા T ની n 4 છે. તેથી, આપણે 1 વત્તા T ની સાથે 2 દ્વારા શરૂ કરીએ અને હું આનો વિસ્તૃત કરો. પછી મને 1 વત્તા 1 વત્તા T ની 2 ચોરસ દ્વારા મળશે અને આ કિસ્સામાં મને ફરીથી 1 વત્તા 1 વત્તા 1 દ્વારા ટીના 2 સમઘન દ્વારા મળશે. K પગલાંઓ પછી સામાન્ય રીતે આપણે 1 વત્તા 1 વત્તા 1 k વત્તા 1 ગુણ્યા અથવા k વખત અને n ની 2 ઘાત એ k થી. હવે અમે ક્યારે બંધ કરીએ? જ્યારે અમે વાસ્તવમાં ટી 0 મેળવીએ છીએ અથવા અમે એમ પણ કહી શકીએ કે ટી 1 માટે તે એક પગલું લે છે, ત્યારે આપણે સાવચેત રહેવું છે. તેથી, જ્યારે આ અભિવ્યક્તિ 1 બને છે ત્યારે n એ 2 ની બરાબર 2 થાય છે. તેથી, જ્યારે n એ 2 ની બરાબર થાય, તે બરાબર લોગની જમણી વ્યાખ્યા છે. કેટલી વખત મને 2 ને ગુણાકાર કરવો પડશે, n ને મેળવવા માટે અને તે એ છે કે જે આપણે જોઈએ તે મૂલ્ય છે. લોગ એન પગલાઓ પછી આ શબ્દ 1 ની બહાર આવશે. અમે આશરે લોગ એન ટાઇમ્સ 1 સાથે જોડાઈશું અને તેથી આપણે લોગ એન પગલાઓ મેળવીશું. તેથી આપણે જે કહીએ છીએ તે ખરેખર છે, જો આપણે 1000 મૂલ્યો સાથે શરૂ કરીએ, તો પછીના પગલામાં આપણે 500, આગળનું પગલું 250, આગળનું પગલું 125, આગળનું પગલું 62 અને તેથી વધુ શોધ કરીશું. અને જો આપણે આ કરવાનું ચાલુ રાખીએ તો આપણે 0 અથવા 1 લંબાઈના તુચ્છ શ્રેણીમાં

જઈશું. સાડુ, 1000 બાય 2 ને વિભાજીત કરતા રહો, આપણે કેટલી વાર 1000 બાય 2 ને વિભાજીત કરી શકીએ, તે બરાબર બેઝ માટે 1000 નું લોગ છે અને તે લોગની સમકક્ષ વ્યાખ્યા છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 21:20)

આ બીજી બિંદુ પર પાછા આવે છે હવે અમે કહ્યું છે કે જો આપણે મૂલ્યોની ક્રમમાં ગોઠવણી કરી હોય તો અમે આ હોશિયાર દ્વિસંગી શોધ કરી શકીએ છીએ, પરંતુ યાદ રાખો કે તેમાં ફૂદી જવાની જરૂર છે અને મધ્યમાં ગણતરી કરવી જરૂરી છે અને પછી અમે મધ્યબિંદુ પર મૂલ્ય જોવાની જરૂર છે અને અમે એમ ધારી રહ્યા છીએ કે મધ્યભાગની આ ગણતરી અને મધ્યબિંદુના મૂલ્યને સતત અવધિની સરખામણી કરતા, તેથી જ અમે કહ્યું હતું કે તે 1 વત્તા T ની 2 ઘાત બરાબર છે અને આ મધ્યમાં કમ્પ્યુટિંગનો સમાવેશ થાય છે અને મધ્યબિંદુમાં આવર્તન જોઈ રહી છે. પરંતુ આ ફક્ત એરે માટે કરી શકાય છે કારણ કે માત્ર એરે માટે આપણે પોઝિશન લઈ શકીએ છીએ અને ફૂદકો મારવું જોઈએ અને સતત સમયે તે પોઝિશન પર મૂલ્ય મેળવી શકીએ છીએ, તે યાદીઓ માટે કામ કરશે નહીં, કારણ કે આપણને ith પોઝિશન સતત સમય અલબત્ત, એક અગત્યનું અને સંભવત એટલું સ્પષ્ટ વસ્તુ નથી જો તમે દ્વિસંગી શોધ વિશે વિચાર કરો તો માત્ર એક બહુ ઓછી સંખ્યામાં મૂલ્યો જોઈને, દાખલા તરીકે, અમે તમને 1000 એન્ટ્રીઓની સૉર્ટ કરેલી સૂચિ આપીએ છીએ કારણ કે મેં કહ્યું છે કે મૂલ્ય નથી. ત્યાં આપણે ફક્ત 10 શક્ય એન્ટ્રીઓ શોધવી પડશે, કારણ કે આપણે લોગ એન બાદ રાખીએ છીએ, જે યાદ રાખવું છે કે 2 થી 10 એ 1024 બરાબર બે ગુણ્યા, બે ગુણ્યા, બે દશ ગુણ્યા 1024 છે. 0 અથવા 1 ની નીચે આવે છે. અમે ચોક્કસપણે તે કહીશું કે તે ત્યાં છે કે નહીં તે અમે ચોક્કસપણે સક્ષમ કરીશું. તેથી, અમે ફક્ત 1000, 999 મૂલ્યોમાંથી 10 મૂલ્યોને જોયા છીએ, જે ક્રમમાં નકાર્યા વગરના અવલંબનોથી વિપરીત નથી, જ્યાં આપણે ઉકેલવા પહેલાં દરેક શક્ય મૂલ્ય જોવાની જરૂર છે. તે અત્યંત કાર્યક્ષમ દ્વિસંગી શોધ છે, પરંતુ તે સતત સમય માં ith પોઝિશનમાં ફૂદકો કરવાનો પ્રયત્ન કરવા માટે જરૂરી છે તેથી જો મેં ખરેખર સૂચિ પર દ્વિસંગી શોધ કરી હોય તો પણ તે સૉર્ટ કરેલ હોય અને અરે પર જ્યાં હું પ્રારંભ કરું 0 મી સ્થાને છે અને નીચે આપેલ લિંક્સ દ્વારા ith પોઝિશન સુધી ચાલવું કમનસીબે દ્વિસંગી શોધ મને એરેનો ઉપયોગ કરતી વખતે અપેક્ષિત બોનસ આપશે નહીં. (સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 23:18) તેથી આ અસ્પષ્ટતાપૂર્વક ચર્ચા કરી, આપણે અલબત્ત પાયથોન્સના સંદર્ભમાં કામ કરી રહ્યા છીએ. પ્રશ્ન એ છે, પાયથોન માં યાદીઓમાં બનાવવામાં આવે છે તે તેઓની યાદી આપે છે કારણ કે આપણે તેમના વિશે વાત કરી છે અથવા તેઓ એરે છે. ખરેખર, દસ્તાવેજીકરણ સૂચવે છે કે જો તમે પાયથોન દસ્તાવેજોને જોશો કે તે યાદીઓ છે કારણ કે તમારી પાસે આ વિસ્તરણ અને સંકોચન કાર્યો છે તેથી અમે જોયું કે આપણે એક જોડીએ કરી શકીએ છીએ અથવા આપણે મૂલ્ય દૂર કરી શકીએ છીએ અને આ રીતે. તેઓ આ સાનુકૂળ વસ્તુઓને સમર્થન આપે છે, જે યાદીઓની લાક્ષણિકતા છે, જોકે, પાયથોન આ અનુક્રમિત સ્થિતિને યોગ્ય રીતે આધાર આપે છે, જેથી તે અમને i પર જોવા માટે પરવાનગી આપે છે. જો તમે તેને મોટી સૂચિ પર અજમાવી જુઓ તો તમને મળશે કે તે વાસ્તવમાં કહે છે કે તે લાખો તત્વોની યાદી રચવા માટે તેટલા વધારે સમય નથી લેતો, તમને મળશે કે છેલ્લા સ્થાન પર જવા માટે તે વધુ સમય લેશે નહીં તમે સામાન્ય રીતે સૂચિમાં અપેક્ષા રાખતા પ્રથમ સ્થાને અમે કહ્યું હતું કે છેલ્લા સ્થાને જવા માટે વધુ સમય લેવો જોઈએ. તેમ છતાં જ્યાં સુધી આપણે ચિંતિત છીએ ત્યાં સુધી અમે તેમને એરે તરીકે ગણવા પડશે, અને અમે ભારપૂર્વક આ અભ્યાસક્રમમાં વધુ આગળ વધીએ ત્યારે કેવી રીતે યાદીઓ કામ કરે છે તેના પર ભાર મૂકે છે, અમે વાસ્તવમાં કેટલાક ડેટા માળખાં કેવી રીતે અમલ કરવું તે તપાસ કરીશું. અને આપણે જોશું કે આ પોઇન્ટર સાથેની સૂચિને કેવી રીતે સ્પષ્ટપણે અમલ કરવો કે જે એક તત્વથી બીજામાં નિર્દેશ કરે છે. આ અભ્યાસક્રમના બાકીના ભાગો માટે જ્યારે પણ આપણે પાયથોન સૂચિ પર નજર કરીએ છીએ ત્યારે આપણે તેને એક અરે તરીકે ઉપયોગમાં લઈશું, જેથી જ્યારે આપણે વધુ સૉર્ટ કરવાનું ગાણિતીક નિયમોની ચર્ચા કરીશું અને તે બધા એલ્ગોરિથમ્સ માટે એલોગરિથમ્સનું વિશ્લેષણ કરશે, તો તે એરે હશે, પાયથોન ની સૂચિમાં પાયથોન અમલીકરણનો ઉપયોગ કરીને, પરંતુ જ્યાં સુધી અમે ચિંતિત છીએ ત્યાં સુધી આ સૂચિ આ કોર્સના હેતુ માટે એરેની સમકક્ષ છે.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રોફેસર. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટીટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 03 લેક્ચર - 05

ક્ષમતા. જ્યારે આપણે ટ્રિસંગી શોધમાં જોયું, તેથી ચાલો આપણે થોડોક સમય પસાર કરીએ કે અનૌપચારિક રીતે સમજવું કે અમે કેવી રીતે ગાણિતીક નિયમોની કાર્યક્ષમતાને જુઓ.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 00:02)

સામાન્ય રીતે એક એલ્ગોરિધમનો ઘણા વિવિધ કદના ઇનપુટ પર કાર્ય કરશે, તેથી તે ઇનપુટ કદના કાર્ય તરીકે કાર્યક્ષમતા વિશે વાત કરવા માટે અર્થપૂર્ણ છે. ઇનપુટનું કદ એ છે કે આપણે એન નો ઇનપુટ પર લેવામાં આવેલા સમય વિશે વાત કરવા માટે ટી નો n એ ફંક્શનનો ઉપયોગ કરીશું. અલબત્ત, એક જ કદથી પણ, વિવિધ ઇનપુટ એલ્ગોરિધમ માટે એક્ઝેક્યુટ કરવા માટે અલગ સમય લેશે, તેથી તેમાંથી આપણે કાર્યક્ષમતાના અમારા માપ પ્રમાણે લઈએ છીએ. સંમેલન સૌથી ખરાબ કેસ વર્તન વાપરવાનું છે. કદના તમામ ઇનપુટ્સમાં, જે અમારા એલ્ગોરિધમને સૌથી લાંબો સમય લેવા માટે દબાણ કરશે, અને આ તે છે જેને આપણે સામાન્ય રીતે સૌથી ખરાબ કેસ કાર્યક્ષમતા કહીએ છીએ. દાખલા તરીકે, બાઈનરી શોધ અથવા રેખીય સ્કેનની શોધના કિસ્સામાં હવે, અમે કહ્યું છે કે જ્યારે સૌથી વધુ ખરાબ કિસ્સામાં આપણે શોધવાનો પ્રયાસ કરી રહ્યા છીએ તે આ ક્રમમાં મળ્યું નથી. તેથી, આપણને રેખીય સ્કેનના કિસ્સામાં શોધીએ તે પહેલાં આપણે સમગ્ર ક્રમ અથવા એરે અથવા સૂચિમાંથી સ્કેન કરવું પડશે. અને ટ્રિસંગી શોધની દ્રષ્ટિએ આપણે શોધ અંતરાલને તુચ્છ અંતર સુધી ઘટાડવો પડશે તે પહેલાં આપણે કહી શકીએ કે કિંમત ત્યાં નથી. તેથી તે સૌથી ખરાબ કેસ છે હમણાં, તે ચાલુ થઈ શકે છે કે ઘણા બધા ગાણિતીક નિયમો માં સૌથી ખરાબ કેસ દુર્લભ છે. તે એલ્ગોરિધમનો કેટલો ખરાબ અથવા સારી છે તે વિશે કોઈ પ્રતિનિધિ વિચાર હોઈ શકતો નથી અને તે કદાચ સરેરાશ કેસ વર્તનની જેમ કંઈક આપવાનું હોઈ શકે છે. હવે કમનસીબે, ગાણિતીક રીતે ચોક્કસ રીતે સરેરાશ કેસની જેમ કંઈક નક્કી કરવા માટે સરળ નથી, અમારી પાસે તમામ ઇનપુટ્સ પર સંભાવના વિતરણ હોવું જરૂરી છે અને પછી વિવિધ ઇનપુટ્સ અને વિવિધ આઉટપુટનું માપ લેવું અને પછી આ માટે સંભવના અર્થનું ગણતરી કરો. તેથી મોટા ભાગના કિસ્સાઓમાં આ શક્ય નથી કારણ કે અમે સૌથી ખરાબ કેસ કાર્યક્ષમતા માટે પતાવટ.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 01:59)

જ્યારે આપણે કાર્યક્ષમતા વિશે વાત કરીએ છીએ, કારણ કે અમે કહ્યું છે કે ઇનપુટ કદ અને આઉટપુટ કદ વચ્ચેના સંબંધમાં વ્યાપકપણે રસ છે તેથી અમે આ પ્રમાણપ્રાપ્તિ સુધી વ્યક્ત કરીએ છીએ. તેથી અમે ખરેખર ચોક્કસ સ્થિરાંકોમાં ખરેખર રસ ધરાવતા નથી, ઉદાહરણ તરીકે આપણે એન નો લોગીન કરવાના પ્રમાણમાં T નો ઉપયોગ કરીએ છીએ, ઉદાહરણ તરીકે બાયનરી સર્ચ અથવા n ના કિસ્સામાં રેખીય સ્કેન અથવા એન લો, એન જેવા મોટા મૂલ્યો ચોરસ, એન સમઘન, અથવા તે પણ ઇનપુટ પર ઘાતાંકીય આધાર રાખે છે, તે છે 2 માટે n એ. આપણે આનો ઉપયોગ કરીને આ લખીએ છીએ, મોટા O નોટેશનને શું કહેવામાં આવે છે. તેથી જ્યારે તમે કહો કે T નો n એ મોટા ઓ છે n નો અર્થ શું છે તે છે કે T નો n એ અમુક સતત વખત n છે. એ જ રીતે T ની n મોટી છે o n લોગ એનો અર્થ એ છે કે T નો n એ અમુક સતત વખત n લોગ n છે. બીજા શબ્દોમાં કહીએ તો, તે મૂલ્યમાં અમુક અમુક દ્વારા પ્રમાણસર છે. તેથી, અમે આ અભ્યાસક્રમમાં મોટા ભાગની વિગતમાં જઈશું નહીં કે મોટા ઓ કેવી વ્યાખ્યાયિત છે અને

તે ગણતરી કરે છે, પરંતુ તે એલ્ગોરિધમ્સની કાર્યક્ષમતાને વર્ણવવા માટે ઉપયોગી છે. તેથી અમે અનૌપચારિક રીતે તેનો ઉપયોગ કરીશું અને તમે વધુ ઔપચારિક રીતે વ્યાખ્યાયિત છો તે શોધવા માટે તમે ગાણિતીક નિયમોને વાંચી અને વાંચી શકો છો. આ સંકેતની દ્રષ્ટિએ જ્યારે આપણે કહીએ છીએ કે રેખીય સ્કેન એરેની લંબાઈ અથવા સૂચિની પ્રમાણસર છે, ત્યારે આપણે કહી શકીએ કે રેખીય સ્કેનને એન નો મોટા સમય લાગે છે. એક સોર્ટ કરેલા એરે બાઈનરી શોધ માટે તે જ રીતે, n નો સમય મોટું O લોગ લેશે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 03:24)

તો, અહીં એક કોષ્ટક છે જે ઇનપુટના જુદા જુદા મૂલ્યો માટે ચઢાવવામાં આવે છે જે લોગ n, એન, એન લોગ એન, એન સ્ક્વેર્ડ અને તેથી પરના અનુરૂપ મૂલ્યો હશે. અને આપણે કદાચ અંદાજ કાઢવા માગીએ છીએ તે આ મૂલ્યો, આ નિરપેક્ષ સંખ્યાઓ, વાજબી ઇનપુટ્સ હોઈ શકે છે, જે આપણે થોડીક સેકન્ડની અંદર ગણતરી કરવાની અપેક્ષા રાખી શકીએ. હવે જો આપણે આપણા કમ્પ્યુટર પર કંઈક લખીએ છીએ અને અમને આ દિવસોમાં જલદી જ પ્રતિસાદ મળ્યો નથી, તો આપણે જાણીએ છીએ કે કંઈક ખોટું છે. તો, ચાલો કહીએ કે આપણે એક અથવા બે સેકન્ડમાં ઇનપુટ જોવા માગીએ છીએ, તો આપણે તેને બિનકાર્યક્ષમ ગણતા હોઈશું. તો, જો આપણે આને જોયું તો, અમારે હવે એ સમજવું પડશે કે અમારા કમ્પ્યુટર્સ કેટલા ઝડપી છે. તેથી, કેટલાક સરળ હેન્ડ પ્રયોગો દ્વારા તમે માન્ય કરી શકો છો કે પાયથોન બીજામાં 10 થી 7 મૂળભૂત પગલાઓ કરી શકે છે.

(સ્લાઈડ સમયનો ઉલ્લેખ કરો: 04:19)

તો આપણે શું કરી શકીએ છે તે મોટી લૂપ કરવાનો પ્રયાસ કરો અને એક્ઝેક્યુટ કરો અને જુઓ કે તે કેટલો સમય લે છે. અહીં આપણી પાસે પ્રોગ્રામનો સમૂહ છે જો તમે પહેલાથી જ લખેલું હોવ અને અહીં એક નમૂનો છે. તેથી જો હું કહીશ speed4 dot py. તે મૂળભૂત રીતે લૂપ 10 થી 4 વખત ચલાવે છે, તેથી તેનું નામ 4 છે. તેથી, શ્રેણી 0 થી 10000 ઓછા 1 માં મીટર માટે, તે માત્ર મૂલ્યને m નું મૂલ્ય નક્કી કરે છે અને છેલ્લે આ નિવેદન છે જે આપણે અત્યાર સુધી જોયું નથી, પરંતુ તે તદ્દન સાહજિક હોવી જોઈએ જે કહે છે કે n ની કિંમત પ્રિંટ કરે છે. તે જ રીતે speed5 એ 10 થી 5 વખત કરે છે, speed6 આ 10 થી 6 વખત કરે છે, speed7 આ 10 થી 7 વખત અને તેથી વધુ. આ સ્ક્રિપ્ટોનું એક ટોળું છે જે આપણે પાયથોન માટે speed4 થી speed9 સુધી લખ્યું છે. હવે જો તમે યુનિક્સ અથવા લિનક્સમાં કામ કરી રહ્યાં હોવ તો ત્યાં એક સરસ આદેશ કહેવાય છે. સૌ પ્રથમ હું પાયથોન લઈ શકું છું અને હું આ જેમ પાયથોન પ્રોગ્રામના નામનો સીધો ઉપયોગ કરી શકું છું. તેથી, હું પાયથોન 3.5 કહી શકું છું અને આ સ્ક્રિપ્ટનું નામ આપી શકું છું અને તે તેને એક્ઝેક્યુટ કરશે અને તમને જવાબ આપશે. પરંતુ હવે વધુમાં, ઉપયોગી સમય કહેવાય છે. તેથી, સમય અમને કહે છે કે આ વસ્તુ કેવી રીતે અમલમાં લાવે છે અને તે સામાન્ય રીતે ત્રણ જથ્થામાં આ અહેવાલ આપે છે; વાસ્તવિક સમય, વપરાશકર્તા સમય અને સિસ્ટમ સમય. તેથી, શું વાઇડખરેખર ખરેખર કહેવાતા વપરાશકર્તા સમય જોવાની જરૂર છે કે જો હું આ લૂપ 10 થી 4 વખત કરું છું તો તે આપણને સેકન્ડ 0.03 સેકન્ડનો અપૂર્ણાંક લેશે. જો હું આ બીજી બાજુ 5 વખત કરું છું, તો તે 0.03 થી 0.5 સુધી જાય છે. તેથી, તે આશરે 10 નો પરિબળ છે કારણ કે તમે કલ્પના કરો કે તે વાજબી છે. જો હું આ બિંદુ 6 વખત કરું તો ફરીથી તે 10 ના પરિબળથી નહીં પણ તે 0.2 સેકન્ડ જેટલું વધ્યું છે. હવે આપણે મર્યાદા તરફ આવીએ છીએ કે આપણે 10 થી 7 નો દાવો કરીએ છીએ. તેથી, જો આપણે speed7 dot py દાખાવીએ છીએ, જે લૂપ 10 થી 7 છે તે લગભગ 1 સેકન્ડ લે છે. મારો અર્થ એ કે આ એક ચોક્કસ ગણતરી નથી, પરંતુ જો તમે તેને વારંવાર ચલાવો તો તમે દરેક સમયે કહેશો, કારણ કે કેટલાક અન્ય પરિબળો છે જેમ કે સિસ્ટમ માટે પાયથોન ઇન્ટરપ્રીટર અને તે બધું લોડ કરે છે, પરંતુ જો તમે તે કરો છો વારંવાર તમે જુઓ છો કે 10 થી 7 એ બીજા કે તેથી વધુ વિશે લે છે. આ મારા કહેવાનો આધાર છે કે પાયથોન બીજામાં 10 થી 7 ઓપરેશનો કરી શકે છે. અને ફક્ત સમજાવવા માટે, જો તમે વાસ્તવમાં 10 થી 8 માટે તે કરો છો તો તમે જોઈ શકો છો કે તે ખૂબ લાંબો સમય લે છે, અને વાસ્તવમાં તે લગભગ 10 થી 12

સેકન્ડમાં એક્ઝિક્યુટ થાય છે જેથી અમે આશા રાખીએ કે આઉટપુટ જુઓ. જેમ તમે જોઈ શકો છો કે 10 સેકન્ડ ખૂબ જ લાંબો સમય અમને નથી લાગતો, પરંતુ તે ખૂબ જ લાંબો સમય છે જ્યારે તમે પ્રતિભાવની રાહ જોઈ રહેલા સ્ક્રીનની આગળ બેસી રહ્યા છો. તેથી હવે આપણે શું દાવો કરીએ છીએ કે, થોડીક સેકન્ડ લાગે છે તે કંઈક છે જેને આપણે અસરકારક ઇનપુટ તરીકે ગણવું જોઈએ કે જે અમે અમારા કમ્પ્યુટર પર હલ કરી શકીએ છીએ.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 07:10)

તેથી, આપણી ટેબલ પર પાછા આવો એમ ધારી રહ્યા છીએ કે 10 થી 7 ની મર્યાદા એ છે કે આપણે જોઈ રહ્યા છીએ, ચાલો જોઈએ કે જ્યારે આપણે આ અલગ કોલમ પર 10 થી 7 ના ચિહ્નિત કરીએ છીએ. તે સમયની લોગ એન લે છે, પણ 10 થી 10 માટે તે ફક્ત 33 પગલાં લે છે અને અમે દંડ છીએ. અલબત્ત, જો ઇનપુટ રેખીય હોય તો આપણે સતત અવગણીએ છીએ તો કદ 10 થી 7 ની ઇનપુટ 10 લેશે, તેથી આ રેખા અહીં આવે છે. બીજી તરફ જો અમારી પાસે n લોગ એન છે હવે તે એન લોગ એન ની બહાર વળે છે, તેથી તે જાણવું ઉપયોગી છે કે આપણે પહેલા 10 થી 10 નો ઉલ્લેખ કર્યો છે. તેથી, 2 થી 20 પાવર 10 થી 10 થશે, અને 2 ની 30 ઘાત 10 થશે. શક્તિ 9. અહીં લોગ લાઇનરલી વધે છે, કારણ કે આ વસ્તુ 10 ની સત્તાની દ્રષ્ટિએ વધતી જાય છે. તેથી, જ્યારે આપણી પાસે 10 થી 7 હોય તો લોગ કંઈક 20 કંઈક બનશે, તેથી તે ઓર્ડર જ હશે 10 ની, તે એક 0 મૂકવા જઈ રહ્યું છે. તેથી, આપણે કહીએ છીએ કે માપ 10 થી 6 ની ઇનપુટ માટે, અહીં લોગ 10 ના પરિબળને ફાળવવા જઈ રહ્યું છે તેથી તે 10 થી 7 સુધી સમય લાગી રહ્યું છે. હવે નોંધ લો કે જ્યારે તમે ચોરસ કરો છો તો 10 થી 3 પહેલેથી જ 10 થી 6 લેશે. તેથી, 1000 અને 10000 ની વચ્ચે ક્યાંક 5000 ની આસપાસ હોવું જોઈએ જો તમે નસીબદાર હોવ તો તે કંઈક માટે શક્ય મર્યાદા હશે જે એન સ્ક્વેર્ડ ટાઇમ લેશે. અને જ્યારે આપણે n ની ઘડીએ જઈએ છીએ ત્યારે મર્યાદા થોડા હજારથી થોડાક સો ઘટી જાય છે. તેથી, આપણી પાસે 10 થી 6 ની વચ્ચે છે અને 10 થી 9 છે. તેથી, 100 અને 1000 ની વચ્ચે ક્યાંક સ્કેલિંગ 10 થી 6 થી 10 ની 9 ઘાત થાય છે, તેથી 10 થી 7 ક્યાંક 200 અથવા 300 ની આસપાસ હશે જ્યારે તમે એન અને એન ફેક્ટોરિયલ માટે 2 જેવા એક્સપોન્ડેન્શિયલ્સ મેળવો, તો પછી જ્યાં સુધી તમારી પાસે કોઈ ઇનપુટ ન હોય કે જે ખરેખર 10 જેવું નાનું હોય અથવા કંઈક એવું હોય કે આપણે સમસ્યાઓ હટાવતા હોઈએ, કારણ કે અમારી પાસે થોડાક દાયકા છે જે તમે પહેલેથી જ પ્રયંસ છે 10 થી 30 જેવી સંખ્યા. આ આપણને એક વિચાર આપે છે કે જે આપણી સિસ્ટમ છે કે જે અમે કામ કરી રહ્યા છીએ કે જે પાયથોન બીજામાં 10 થી 7 પગલાઓ કરી શકે છે, આપણે ખરેખર આ કોષ્ટકની તપાસ કરવા જોઈએ કે તે કઈ પ્રકારની ઇનપુટ્સ હશે સમયસરના અલ્ગોરિધમનો પ્રક્રિયા કરવા માટે વાસ્તવવાદી છે જે અમે ચલાવી રહ્યા છીએ. હવે પાયથોન 10 થી 7 છે. પાયથોન અન્ય ભાષાઓ કરતા થોડી ધીમી છે, પરંતુ જો તમે C અથવા C++ જેવી ખૂબ ઝડપી ભાષા વાપરી રહ્યા હોવ તો તમે વાસ્તવિકતાથી 10 થી 8 અથવા 10 થી 9 સુધી જવાની અપેક્ષા રાખી શકતા નથી. તેથી આ કોષ્ટક વિવિધ ભાષાઓમાં થોડાક દાયકાની સ્કેલિંગ સુધી વધુ અથવા ઓછા માન્ય છે તેથી, તમે તેને સમગ્ર ભાષાઓમાં વાજબી અંદાજ તરીકે લઈ શકો છો.

(સ્લાઈડ ટાઇમ નો ઉલ્લેખ કરો: 09:43)

સૈદ્ધાંતિક રીતે જો તમે એલ્ગોરિધમ પુસ્તકો અથવા જટીલતા સૈદ્ધાંતિક પુસ્તકો, કોઈ પણ બહુપરીમાણીય, એન પરની કોઈપણ અવલંબન કે જે સતત k માટે k નો ફોર્મ n છે તે કાર્યક્ષમ ગણવામાં આવે છે. આ કહેવાતા પોલિનોમિયલ ટાઇમ એલ્ગોરિધમ્સ છે. તેથી n cubed, n થી 5, n થી 7, આ તમામ સૈદ્ધાંતિક રીતે કાર્યક્ષમ ગાણિતીક નિયમો ગણવામાં આવે છે, જે 2 થી n ની સરખામણીમાં અને તેથી વધારે છે. તેથી તમારી પાસે n થી 7 ની વિરુદ્ધ n છે. તેથી, 7 થી 7 ને કાર્યક્ષમ ગણવામાં આવે છે, 2 થી n એ નથી. પરંતુ જો તમે પહેલાની કોષ્ટક જો જુઓ તો ટેબલ શું કહે છે તે એ છે કે એન ચોરસમાં ખૂબ તીવ્ર મર્યાદા છે, આપણે ફક્ત 4 થી 5000 સુધી કરી શકીએ છીએ. જો તમે એન સ્ક્વેર્ડ ટાઇમમાં કંઈક કરી રહ્યા હોવ તો આપણે આનાથી કંઈક મોટું કાર્ય કરી શકતા નથી થોડા હજારો હવે ઘણી વસ્તુઓ

કે જે આપણે વાસ્તવિક જીવનમાં જોયેલી છે, જેમ કે અમારી પાસે મોટી સ્પ્રેડશીટ છે અથવા અમારી પાસે તે કંઈક છે અને અમે તેને સોર્ટ કરવા માંગીએ છીએ તો તે થોડા હજાર એન્ટ્રીઓ હોવાની શક્યતા છે. ધારો કે, જો તમે માત્ર એક મધ્યમ કદના કંપનીમાં અથવા વર્ગના તમામ શાળામાં અથવા શાળામાં અથવા તેના જેવા તમામ કર્મચારીઓને જોવા માગો છો, તો કેટલાંક હજારો મોટી સંખ્યામાં નથી એના પરિણામ રૂપે, આપણે શું જોઈ શકીએ છીએ કે જો આપણે તેનાથી આગળ વધીએ છીએ તો એન સ્કવેર્ડ અલ્ગોરિધમનો ગણતરી કરવા માટે ઘણો સમય લેશે. તેથી ખરેખર આપણે શું કરી શકીએ તેની મર્યાદા શું છે તે અંગે ખૂબ જ સખત વિચાર કરવો પડે છે અને એટલે શ્રેષ્ઠ શક્ય અલ્ગોરિધમનો ઉપયોગ કરવો તે ખૂબ મહત્વનું છે. કારણ કે કંઈક વધુ સારી રીતે ઉપયોગ કરીને તમે નાટ્યાત્મક ઇનપુટની શ્રેણીને સુધારી શકો છો કે જેના પર તમારું અલ્ગોરિધમનો કાર્ય કરે છે.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રો. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
અઠવાડિયું - 03 લેક્ચર - 06 પસંદગી સોર્ટ કરો

(સ્લાઇડ ટાઇ નો ઉલ્લેખ કરો: 00:02)

આપણે જોયું છે કે શોધ વધુ કાર્યક્ષમ બને છે જો અમારી પાસે છટણી ક્રમ છે. તેથી, એક ક્રમમાં ગોઠવાયેલ નથી એરે અથવા સૂચિ માટે, રેખીય સ્કેન આવશ્યક છે અને આને ઓર્ડર n સમય લાગે છે. જો કે, જો આપણી પાસે સોર્ટ કરેલ એરે હોય તો આપણે દ્વિસંગી શોધનો ઉપયોગ કરી શકીએ છીએ અને દરેક સ્કેનને શોધવા માટે અમે અડધા અંતરાલનો ઉપયોગ કરી શકીએ છીએ અને તેથી, લોગ એન સમય લો. હવે સોર્ટિંગ પણ આડપેદાશ તરીકે અમને કેટલીક ઉપયોગી માહિતી આપે છે. દાખલા તરીકે, મધ્યમ મૂલ્ય - સમૂહમાં સરેરાશ મૂલ્ય એ મૂલ્ય છે જે અડધા મૂલ્ય મોટો છે અને અડધા નાના છે. એકવાર અમે ક્રમને સોર્ટ કર્યા પછી, મિડપોઇન્ટ આપમેળે અમને મધ્યમ આપે છે. અમે આવર્તન કોષ્ટકો બનાવવા અથવા ડુપ્લિકેટ્સની ચકાસણી કરવા જેવી વસ્તુઓ પણ કરી શકીએ છીએ, આવશ્યકપણે એકવાર આપણે ક્રમાંકને સોર્ટ કરીએ છીએ, બધા સરખા મૂલ્યો બ્લોક તરીકે એક સાથે આવે છે. તેથી, પ્રથમ બે વખત બ્લોક છે કે નહીં તે ચકાસીને, અમે ચકાસી શકીએ છીએ કે શું અમારી સૂચિમાં ડુપ્લિકેટ છે? અને દરેક બ્લોક માટે, જો આપણે બ્લોકનું કદ ગણીએ, તો આપણે ફિક્સ્ડ નંબર ટેબલ બનાવી શકીએ છીએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 01:06)

ચાલો સિક્વન્સને સોર્ટ કરવાના અમુક રીતો જોઈએ. તેથી, અરેને ભૂલી જાઓ અને ક્ષણ માટે સૂચિ કરો, અને ચાલો આપણે ભૌતિક કાર્યો તરીકે સોર્ટ કરવાનું વિચારીએ. ધારો કે તમે કોઈ અભ્યાસક્રમ માટે અધ્યક્ષ સહાયક છો, અને શિક્ષક અથવા પ્રશિક્ષકને પરીક્ષા કાગળ સુધારવાનો સમાપ્ત કર્યો છે અને હવે તમે તેમને ગોઠવવું ઇચ્છો છો, જેથી સૌથી વધુ ગુણવાળા ગુણ - સૌથી વધુ ગુણ ટોચ પર છે, એક બીજો સૌથી વધુ માર્ક નીચે છે અને તેથી વધુ. તેથી, તમારું કાર્ય ગુણના ઉત્તરતા ક્રમમાં સુધારણા પછી જવાબના કાગળો ગોઠવવાનું છે, ટોચની સૌથી વધુ સૌથી ઉચ્ચતમ ચિહ્ન હોવું જોઈએ.

(સ્લાઇડ સમય જણાવો: 01:46)

આ કરવા માટે એક કુદરતી વ્યૂહરચના છે. તેથી, આપણે શું કરી શકીએ તે વારંવાર સૌથી મોટા અથવા નાના કાગળ માટે જુઓ. હવે આ કિસ્સામાં, અમે તળિયેથી સ્ટેક બનાવવાની તૈયારી કરીએ છીએ, જો સૌથી વધુ ચિહ્ન ટોચ પર હોય તો સૌથી નીચા ચિહ્ન તળિયે હશે. તો, આપણે શું કરીએ છીએ તે આપણે સમગ્ર સ્ટેકને સ્કેન કરીએ છીએ, અને ન્યુનતમ ગુણ સાથે કાગળને શોધી કાઢીએ છીએ. આપણે આ કેવી રીતે કરી શકીએ, જ્યાં આપણે દરેક કાગળને બદલાવમાં રાખીએ છીએ, દર વખતે જ્યારે આપણે નાના માર્ક સાથે કાગળ શોધીએ છીએ ત્યારે આપણા હાથમાં જે એક હોય તે આપણે બદલીએ છીએ અને તેને આપણે શોધી કાઢીએ છીએ. સ્કેનના અંતે, અમારા હાથમાં અમારી પાસે લઘુત્તમ ગુણ સાથે કાગળ હશે. શરૂઆતમાં, અમે એવું ધારીએ છીએ કે ટોચનો સૌથી વધુ કાગળ લઘુત્તમ ગુણ ધરાવે છે અને અમે નીચે જતા રહેવું અને તેને શોધીએ છીએ તે કોઇપણ નિમ્ન ચિહ્ન સાથે બદલીએ છીએ. આ સ્કેન કર્યા પછી, અમે અમારા હાથમાં કાગળ લઇએ છીએ અને તેને કોરે મૂકી દઈએ છીએ અને બીજું સ્ટેક કરો જ્યાં આ તળિયે સૌથી વધુ વસ્તુ છે. હવે આપણી પાસે n ઓછા 1 પૃષ્ઠ છે, આપણે પ્રક્રિયાને પુનરાવર્તન કરીએ

છીએ. અમે આ n ઓછા 1 પેપર્સમાં ન્યૂનતમ ગુણ શોધી કાઢીએ છીએ અને બીજા બધા ઉપર સૌથી વધુ ચિહ્નિત કરીએ છીએ. હવે, આપણી પાસે બે પેપર સ્ટેક્સ છે, ક્રમમાં આપણે આ કરવાનું ચાલુ રાખીશું તો આપણે સ્ટેકને ઉપરથી ઉપરની તરફ બનાવશું જે તળિયે સૌથી ઓછું ચિહ્ન છે, અને ટોચ પર સૌથી વધુ ચિહ્ન છે.

(સ્લાઇડ સમય જણાવો: 03:07)

ધારો કે આ 6 પેપર્સ છે. તેથી, અમે 74, 32, 89, 55, 21 અને 64 માર્ક સાથેના કાગળો ધરાવીએ છીએ. જો આપણે આ સૂચિને ડાબેથી જમણી બાજુએ સ્કેન કરીએ છીએ, તો અમે શોધીશું કે 21 સૌથી નીચલું ચિહ્ન છે. તેથી, અમારી વ્યૂહરચના કહે છે કે તે સૌથી નીચલા માર્ક સાથે એક પસંદ કરો અને તેને નવા ક્રમ અથવા નવા સ્ટેક પર ખસેડો, જેથી અમે તે કરીએ.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 03:38)

હવે ફરી, અમે ડાબેથી જમણી તરફ સ્કેન કરીએ છીએ, અલબત્ત 21 ની અવગણના થઈ ગઈ છે, તેથી અમારી પાસે સ્કેન કરવાના પાંચ નંબરો છે. અમે શોધીશું કે 32 આપણી આગામી છે. અને પછી આગળના પગલામાં આ રીતે આગળ વધતા અમે 55 અને પછી 64 અને પછી 74 અને આખરે 89 ની પસંદગી કરીશું. આ રીતે છ તત્વોની યાદીમાં છ સ્કેન કરવાથી, આપણે એક નવી અનુક્રમ બનાવ્યું છે જે આમાં છે. છ તત્વો તેમની કિંમત અનુસાર આદેશ આપ્યો.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 03:59)

આ ચોક્કસ વ્યૂહરચના જે ખૂબ જ કુદરતી અને સાહજિક છે તેનું નામ પસંદગી સોર્ટ કહેવાય છે, કારણ કે દરેક તબક્કે આપણે ક્રમાંકિત ક્રમમાં આગામી તત્વ પસંદ કરીએ છીએ અને તે અંતિમ સોર્ટ કરેલી સૂચિમાં ખસેડી જે સાચું છે ઓર્ડર

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 04:14)

અલ્ગોરિધમનો કે જે હમણાં હમણાં અમલ કર્યો છે, અમને સોર્ટ કરેલ મૂલ્યોને સંગ્રહિત કરવા માટે બીજી સૂચિ અથવા સેકન્ડ ક્રમ બનાવવાની જરૂર છે. તેથી, અમે પ્રથમ ક્રમાંકમાંથી વસ્તુઓ ખેંચીને રાખ્યા હતા, અને તેને બીજા ક્રમાંકમાં મૂકી દીધું. જો કે, થોડુંક વિચાર અમને જણાવશે કે અમને આ કરવાની જરૂર નથી. જ્યારે પણ આપણે સૂચિમાંથી એક તત્વને બીજા નાનામાં આગળ લઈએ છીએ, ત્યારે આપણે તેને શરૂઆતમાં ખસેડી શકીએ છીએ અને તે શરૂઆતમાં જે છે તેની સાથે બદલાવ કરી શકીએ છીએ. આપણે પ્રથમ સ્થાને મુલ્ય સાથે ન્યૂનતમ મુલ્ય સ્વેપ કરી શકીએ છીએ, ત્યારબાદ આપણે બીજા સ્થાને આગળ જુઓ અને બીજું લઘુતમ મૂલ્ય શોધીએ અને બીજા સ્થાને સ્વેપ કરીએ અને આ રીતે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 04:53)

તો, જો આપણે પહેલા આ જ ઇનપુટ પર આ ફેરફાર કરેલ અલ્ગોરિધમનો અમલ કરીશું. અમારી પ્રથમ સ્કેનમાં, અમે પ્રથમ સ્થાને ડાબી બાજુથી 74 થી શરૂ કરીશું, અને ન્યૂનતમ 21 હશે. હવે, 21 ને નવી સૂચિમાં ખસેડવાને બદલે, હવે અમે 21 અને 74 ને સ્વેપ કરીશું.

(સ્લાઇડ ટાઇમ: 05 : 09)

તો, 21 એ શરૂઆતમાં આવે છે અને 74 એ એવી સ્થિતિ પર જાય છે જ્યાં 21 હતી. હવે આપણે 21 થી કંઈપણ ચિંતા કરવાની જરૂર નથી, જો તમે તેને 32 માથી શરૂ કરવા માંગો છો, તો અમારે આ સ્લાઇડને જોવાની જરૂર છે. અમે આ કરીએ છીએ અને અમને બીજા નાના તત્વ મળે છે. હવે, પ્રારંભિક તત્વ 32 છે અને બીજો સૌથી નાનું તત્વ પણ 32 થાય છે જે આ સ્લાઇડમાંનું સૌથી નાનું ઘટક છે. તેથી, અમે ફક્ત 32 જ રાખીએ છીએ જ્યાં તે છે. હવે આપણે બે પોઝિશનમાંથી આગામી સ્લાઇડ શરૂ કરીએ. શરૂઆતનું તત્વ 89 છે પરંતુ સૌથી નાનું તત્વ 55 છે. તેથી, આ સ્કેન સમાપ્ત કર્યા પછી આપણે કહી શકીએ 55 ને ત્રીજા સ્થાને ખસેડશે અને 89 શતેને બદલવું જોઈએ

(સ્લાઇડ સમય જણાવો: 05:58)

આ રીતે આપણે ચાલુ રાખીએ છીએ હવે અમે 64 મૂક્યા છે જ્યાં 89 છે, અને આખરે 74 બરાબર છે અને 89 એ સાચું સ્થાન છે. અને આપણી પાસે પસંદગી સોર્ટનો ઉપયોગ કરીને એક સોર્ટ ક્રમ છે જે બીજી શ્રેણી બનાવવાને બદલે, અમે હમણાં જ વ્યવસ્થિત રીતે નાના ઘટક ખસેડ્યાં છે જે અમે હમણાં જ શોધી રહ્યાં છીએ તે સેગમેન્ટ અથવા વિભાગ સાથે શરૂઆતમાં જોવા મળે છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 06:17)

અહીં એક અત્યંત સરળ પાયથોન ફંક્શન છે જે પસંદગી પસંદગીને લાગુ કરે છે. પસંદગીની સોર્ટ વિશેનું મુખ્ય વિચાર એ છે કે અમારી પાસે આ અનુક્રમ છે જે સાથે શરૂ થતાં n તત્વો છે. પ્રથમ વખત, અમે સમગ્ર ક્રમ સ્કેન કરીશું, અને અમે આ બિંદુને આ નાનું તત્વ ખસેડીશું. પછી અમે એક પછીથી સ્કેનને સ્કેન કરીશું, ત્યારબાદ આપણે બે પછીની શ્રેણીને સ્કેન કરીશું, અને દરેક બિંદુએ ગમે તે વિભાગમાં, અમે નાના ઘટકને શરૂઆતમાં ખસેડીશું. અમે દરેક સ્કેનના આ પ્રારંભિક બિંદુઓ ધરાવીએ છીએ, તેથી પ્રારંભ બિંદુ શરૂઆતમાં 0 થી શરૂ થાય છે, અને પછી તે 1, 2 સુધી લંબાઈની લંબાઈ 1 સુધી જાય છે. તેથી, 0 થી શરૂ થતી મૂલ્યો માટે, અસ્પષ્ટ રીતે આ યાદ રાખવું 0 છે, 0 ની લંબાઈ 0 થી, અમે સૌ પ્રથમ ન્યૂનતમ મૂલ્ય શોધવાની જરૂર છે. અમે ધારીએ છીએ કે લઘુત્તમ મૂલ્ય આ સ્લાઇડની તે સ્થિતિની શરૂઆતમાં છે. તેથી અમે જણાવ્યું હતું કે પ્રારંભની સ્થિતિમાં લઘુત્તમ સ્થિતિ છે; યાદ રાખો કે શરૂઆતની સ્થિતિ 0 થી લંબાઈ 1 ની લંબાઈથી બદલાઈ રહી છે. તેથી, દરેક સ્લાઇડની શરૂઆતની સ્થિતિ એ હાલની સ્થિતિ છે જે અમે હાલમાં જોઈ રહ્યા છીએ. પછી અમે આ પદ પરથી સ્કેન કરીએ છીએ અને જો આપણે સખત નાની કિંમત શોધીએ છીએ. જો હું મારું લઘુત્તમ મૂલ્ય છે તે અમે યોગ્ય રીતે માનવું તે કરતાં નાની છે, તો અમે વર્તમાન ઇન્ડેક્સ દ્વારા ન્યૂનતમ સ્થિતિને બદલીએ છીએ. આ સમગ્ર વસ્તુમાંથી પસાર થયા પછી, આપણે જોયું હશે કે આ સ્થિતિ લઘુત્તમ મૂલ્યની સ્થિતિ છે. પછી આપણે આ બંનેનું વિનિમય કરવાની જરૂર છે, તેથી અમે પ્રારંભની સ્થિતિ અને લઘુત્તમ સ્થિતિને લઈએ છીએ અને આ એકસાથે ચાલીએ છીએ, જે આપણે બે મૂલ્ય લે તે પહેલાં આપણે જોયું છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 07:59)

ચાલો જોઈએ કે આ કેટલો સમય લે છે. દરેક પુનરાવર્તનમાં અથવા દરેક રાઉન્ડમાં, અમે લંબાઈ K નો ટુકડો જોઈ રહ્યા છીએ, અને અમે તેમાં ન્યૂનતમ શોધી રહ્યા છીએ અને પછી શરૂઆત સાથે તેને આપ-લે કરી રહ્યા છીએ. હવે આપણી પાસે લંબાઈના મૂલ્યોનું ક્રમમાંગોઠવાયેલું અનુક્રમ છે, આપણે લઘુત્તમ મૂલ્ય શોધવા માટે તે બધાને જોવાની જરૂર છે, કારણ કે અમને કોઈ ખ્યાલ નથી કે તે ક્યાં છે. અમે કોઈપણ સમયે બંધ ન કરી શકીએ છીએ અને જાહેર કરી શકીએ છીએ કે આ ઉપરાંત કોઈ નાના મૂલ્યો નથી. તેથી, લંબાઈ K ના એક ક્રમમાંગોઠવાયેલનથી સેગમેન્ટમાં ન્યૂનતમ શોધવા માટે, તેને K પગલાંઓ એક સ્કેનની જરૂર છે. અને હવે આપણે તે આખા સ્લાઇડના સેગમેન્ટ સાથે શરૂ કરીએ છીએ જે લંબાઈની સ્લાઇડ છે, પછી લંબાઈ n નો 1 નું એક સ્લાઇડ અને તેથી વધુ. અને તેથી, જો આપણે સામાન્ય ટી બરાબર T તરીકે લખીએ છીએ તો તે પસંદગીના સોર્ટનો ઉપયોગ કરીને આકાર n નું

ઇનપુટ લે છે, આ પ્રથમ સ્લાઇસ માટે n હશે, બીજી સ્લાઇસ માટે n બાદ 1, હું તેનો અર્થ એક પોઝિશન એક પછીથી, બેવડા પદ માટે અને પછીથી બેવડા 2 નો. અને જો હું આ બધાને ઉમેરીશ તો આપણે આ પરિચિત રકમ 1 વત્તા 2 વત્તા 3 ને n સુધી રાખીએ છીએ, જે તમે આશા રાખશો કે તમે જોઈ શકો છો આ એક્સપ્રેશન n દ્વારા n વત્તા 1 થી 2 માં આપવામાં આવે છે. હવે n માં n વત્તા 1 2 દ્વારા, જો આપણે વિસ્તૃત કરીએ તો તે 2 ચોરસ દ્વારા 2 વડે n થાય છે. હવે આ મોટું ઓ સંકેત જે અમને કહે છે કે તે અ ચોરસનો પ્રમાણસર છે; જ્યારે આપણી પાસે એવા અભિવ્યક્તિઓ હોય છે કે જે n , n ચોરસ, એન ક્યુબ જેવા વિવિધ શબ્દો ધરાવે છે, તે તારણ આપે છે કે આપણે ફક્ત સૌથી વધુ શબ્દ રેકૉર્ડ કરવાની જરૂર છે કારણ કે, એન ચોરસ એ સૌથી વધુ શબ્દ એન ચોરસ એન કરતાં વધુ ઝડપી વધે છે, અમે ઓ એન ચોરસમાં તેને સરળ બનાવી શકીએ છીએ. જો તમે આ શા માટે આવું છે તે જોવા માંગો છો, તો તમારે કોઈપણ પ્રમાણભૂત એલ્ગારિધમ પુસ્તકની શોધ કરવી જોઈએ, તે તમને સમજાવશે કે તમે મોટા ઓ કેવી રીતે ગણતરી કરો છો, પરંતુ અમારા હેતુઓ માટે તે યાદ રાખવું પૂરતું છે કે મોટાભાગના શબ્દપ્રયોગ કે આપણે જોઈ રહ્યા છીએ

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 09:57)

અમે કહ્યું છે કે પસંદગીના સોર્ટ જેવા એલ્ગારિધમને સોર્ટ કરવા માટે, જે ઓર્ડર એન સ્ક્વેર લે છે તે ખૂબ મોટા મૂલ્ય માટે કામ કરશે નહીં જે 5000 કરતા વધારે લંબાઈ માટે કહે છે. તો ચાલો જોઈએ કે આ વસ્તુઓ કેવી રીતે કામ કરે છે. પ્રથમ, આ તે જ કોડ છે જે અમારી સ્લાઇડમાં હતો, તેથી પસંદગી સોર્ટ સ્કેન સ્લાઇસેસ 0 થી 1 ની લંબાઈ 1 સુધી. ચાલો પાયથોન ઇન્ટરપ્રીટર શરૂ કરીએ. અને હવે આપણે આ ફાઇલની પસંદગી સોર્ટ લોડ કરીશું. હવે પસંદગી સોર્ટ જે રીતે કામ કરે છે તેની નોંધ લો, તે વાસ્તવમાં મૂલ્ય નહીં આપે છે કે કઈ પસંદગી સોર્ટ કરે છે તે મૂલ્ય લે છે જે તે સૂચિને પસાર થાય છે અને તે તેને સ્થાનાંતરિત કરે છે આમાંથી કંઈપણ જોવા માટે, આપણે તેને પ્રથમ નામ આપવું પડશે. તેથી, ચાલો આપણે 3, 7, 2, જેમ કે લિસ્ટ લઈએ અને ઉદાહરણ પસંદ કરીએ. અને હવે આપણે 1 નો જોઈએ, તે 2, 3, અને 7 ની ચઢતા ક્રમમાં યોગ્ય રીતે સોર્ટ થાય છે. હવે સામાન્ય રીતે આપણે લાંબી યાદી લઈ શકીએ છીએ. દાખલા તરીકે, આપણે આ રેંજ ફંક્શનનો ઉપયોગ કરી શકીએ અને કહીએ છીએ કે રેંજને લઈને જે શ્રેણી બનાવવામાં આવે છે તે આપો 500 થી 0 ની બાદબાકીના પગથિયું સાથે. તેથી, તે લંબાઈ 500 ની ઉતરતા સૂચી છે.

(સ્લાઇડ ટાઇમ: 11 : 28)

જો હું એલ પર નજર કરું છું, તો તે 500 ની નીચે 1 છે.

(ધીરે ધીરે ટાઇપ કરો: 11:30)

અને હવે જો હું કહું છું કે પસંદ કરેલ પસંદગીઓને સોર્ટ કરો, તો તે 1 થી 500 તરીકે સોર્ટ થાય છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો : 11:44)

હવે અમારો દાવો છે કે આ 5000 ની આસપાસ અસરકારક રીતે કાર્ય કરવાનું બંધ કરશે. તો ચાલો જોઈએ કે હું 500 ની જગ્યાએ 1000 ને આ બનાવે છે, અને પસંદગીની પસંદગી કરો તો તમે જોઈ શકો છો કે ત્યાં એક નોંધપાત્ર તફાવત છે. હવે જો હું કહું છું 2000 માટે, તો પછી થોડો સમય અંતર છે જો હું તે 5000 માટે કરું તો તમે જોઈ શકો છો કે થોડોક સમય લાગે છે કે તે એક સેકન્ડથી વધુ સમય લે છે. આ ફક્ત અમારા દાવાને માન્ય કરાવવા માટે છે કે પાયથોન માં જો તમે એક સેકન્ડમાં કંઈક અપેક્ષા રાખતા હોવ તો પછી તમે ખાતરી કરો કે પગલાંની સંખ્યા 10 થી 7 ની નીચે છે. અને 5000 ચોરસ તમને 10 થી 7 સુધી આગળ લઈ જાય છે, તમે એક ખૂબ લાંબા સમય લેવાની અપેક્ષા કરી શકો છો.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રોફેસર. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ
ઇન્ડિયન ઇન્સ્ટીટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
અઠવાડિયું - 03 લેક્ચર - 07

નિવેશ સૌંદર્ય પહેલાના વ્યાખ્યાનમાં અમે સૌંદર્ય માટે એક કુદરતી વ્યૂહરચના જોયું છે, જે તમે ક્યારે અરજી કરશો હાથ દ્વારા કંઈક કરવું, પસંદગીની સૌંદર્ય કરો

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 00:02)

હવે ચાલો આપણે એક અન્ય કુદરતી વ્યૂહરચના પર વિચાર કરીએ જે આપણા બધા કોઈ સમયે ઉપયોગ કરે છે. તેથી, બીજી વ્યૂહરચના નીચે મુજબ છે:

(સ્લાઇડ ટાઇ નો ઉલ્લેખ કરો: 00:17)

હવે અમારી પાસે પેપરનો સ્ટેક છે, જે તેમના પર ગુણ સાથે યાદ આવે છે અને અમારે એક નવા સ્ટેકની ગણતરી કરવી પડશે કે જે આ ગુણ ઉતરતા ક્રમમાં ટોચથી નીચે સુધી ગોઠવાય છે. તો, આપણે સ્ટેકનાં પહેલા કાગળ લઈશું અને વ્યાખ્યા દ્વારા નવી સ્ટેક બનાવીશું આ નવું સ્ટેક હવે સૌંદર્ય થાય છે કારણ કે તેમાં ફક્ત એક પેપર છે. હવે આપણે જૂના સ્ટેકમાંથી બીજા પેપરને પસંદ કરીએ છીએ અને આપણે તેના ગુણને પ્રથમ પેપરની તુલનામાં જોયા છે જે અમે ખેંચી લીધો છે. જો તે નાનું હોય, તો અમે તેને નીચે મૂકીશું; જો તે વધારે છે, તો અમે તેને ઉપર મૂકીશું. તેથી, આ પ્રક્રિયામાં, હવે અમે ઉતરતા ક્રમમાં ગોઠવાયેલા બે પેપરનો નવી સ્ટેક ધરાવો છો. ત્રીજા કાગળ સાથે અમે શું કરીએ છીએ, સાથે સાથે ત્રીજો પેપર ત્રણ હોદ્દામાંથી એક હોઈ શકે છે; તે ક્યાં તો અમે પહેલાં જોયું બે કરતાં મોટી હોઈ શકે છે. તેથી તે ટોચ પર જઈ શકે છે, અથવા તે બે વચ્ચે હોઈ શકે છે, અથવા તે નીચે જઈ શકે છે તેથી, આપણે શું કરીએ તે ઉપરથી ઉપરથી નીચે સુધી સ્કેન કરીએ છીએ અને તે જો લાંબા સમય સુધી જો આપણે જોયું હોય તો પેપર કરતા ઓછું હોય તો, જ્યાં સુધી તે યોગ્ય સ્થળે ન લાગે ત્યાં સુધી તેને નીચે દબાવો. અમે કાગળ શામેલ કરીએ છીએ કે જે અમે આગળની સૌંદર્ય કરેલા સ્ટેકમાં અમે નિર્માણ કરી રહ્યા છીએ તે યોગ્ય સ્થાને આગળ વધીએ છીએ. તેથી, દરેક અનુગામી કાગળ માટે આ કરવાનું ચાલુ રાખો, અમે ચોથા પેપર લઈશું અને બાકીના ત્રણ વચ્ચે યોગ્ય સ્થાને દાખલ કરીશું અને આ રીતે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 01:31)

આ દેખીતી રીતે ઇન્સેર્શન સૌંદર્ય કહેવામાં આવે છે. તો, ચાલો જોઈએ તે કેવી રીતે કામ કરશે. તેથી, આપણે આ જ સૂચિ સાથે શું કરીએ છીએ કે આપણે પસંદગીના સૌંદર્ય માટે છીએ તે આપણે પ્રથમ મુલ્ય પસંદ કરીશું અને તેને નવા સ્ટેક પર ખસેડીશું, હવે મારી પાસે એક નવું સ્ટેક છે જેનું બરાબર એક મુલ્ય 74 છે. પછી જ્યારે હું પસંદ કરું છું 32, 74 થી નાના 32 થી, હું તેને 74 ની ડાબી બાજુએ દબાવીશ. હવે 89 બંને કરતા મોટો છે, તેથી હું તેને જમણી બાજુએ સ્ટેકની ટોચ પર રાખું છું; 55, હવે હું 89 અને 74 ની આદર સાથે જોઉં છું, તેથી તે 89 કરતા ઓછું છે. તેથી, તે 89 ની ડાબી બાજુએ જાય છે, પછી હું 74 ને જોઉં છું, તે 74 કરતાં નાનું છે, તે ડાબી બાજુએ જાય છે. તેથી, આખરે તે 32, 55, 74, 89. 21 ની જેમ સ્થાયી થાય છે, તેવી જ રીતે મને ટોચથી શરૂ થવું પડશે અને કહેવું છે કે તે 32 કરતા ઓછી 55 ના હોવાની તુલનામાં નાની 74 કરતાં નાની છે, તેથી તે બધી રીતે જાય છે. બાકી અને આખરે, 64 84 અને 89 અને 74 ની સ્થિતિને નીચે ખસેડશે, પરંતુ તે 55 થી ઉપર બંધ રહેશે. તેથી, આ રીતે દાખલ થવું એ નવી

સૂચિનું નિર્માણ કરશે. તમે આગલા મૂલ્યને પસંદ કરી રહ્યાં છો અને તે પહેલાથી જ સોર્ટ કરેલી સૂચિમાં શામેલ કરો છો

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 02:44)

આપણે એક અનુક્રમિત સોર્ટ અનુક્રમનું નિર્માણ કરવાનું શરૂ કરીએ છીએ જેમાં એક તત્વ આગામી ક્રમમાં ગોઠવાયેલ નથી તત્વને પસંદ કરે છે અને તે પહેલાથી સોર્ટ ક્રમમાં તેને યોગ્ય સ્થાને દાખલ કરે છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 02:56)

આપણે આ કરી શકીએ છીએ કે જેમ આપણે એક નવી શ્રેણી બનાવ્યાં વિના દાખલ કર્યું છે અને અહીં એક ફંક્શન નિવેશ સોર્ટ છે જે પાચથોન માં વ્યાખ્યાયિત કરે છે. તેથી, આપણે શું ધારીશું કે કોઈપણ બિંદુએ, અમારી પાસે ક્રમ 0 થી n ઓછા 1 છે અને કેટલાક સ્થાનો છે, જેથી આ બિંદુ સુધી બધું સોર્ટ થાય. અને હવે હું શું કરીશ, હું અહીં આગળના ભાગને પસંદ કરીશ અને જ્યાં સુધી હું તેને મૂકવા માટે યોગ્ય સ્થાન ન મળે ત્યાં સુધી હું તેને ખસેડીશ, જેથી હવે સોર્ટ કરેલ વસ્તુ આ લંબાઈને લંબાઈ લાગી શકે છે તેથી, અમે લંબાઈના એક ક્રમની શ્રેણી લઈએ છીએ અને અમે તેને વર્તમાન સૂચિમાં હું વત્તા એક સ્થાને દાખલ કરીને હું વત્તા 1 જેવી સેકંડ સોર્ટ અનુક્રમમાં વિસ્તૃત કરી શકીએ છીએ. તેથી, આપણે આ સ્થાનને સ્લાઇસ એન્ડ જમણી બાજુએ લઈ જઈશું, સ્લાઇસ એન્ડ એ છેલ્લી પોઝિશન હશે જે આપણે પહેલાથી જ છટાણી કરી છે. તેથી, આ સ્લાઇસ અંત હોઈ રહેવા આવે છે તેથી, આપણે કહીએ છીએ કે તે સ્લાઇસ અલ મુલ્ય 0 થી શરૂ થાય છે અને n બાદ 1 થી સ્થાને જાય છે. અને દરેક સમયે, અમે મૂલ્ય પર જુઓ વાસ્તવમાં સ્લાઇસ અપ સ્લાઇસ એન્ડ માટે છે 1 માફ કરશો. તેથી, સ્લાઇસ એન્ડ સંખ્યાબંધ ઘટકો છે જેને અમે સોર્ટ કર્યા છે. અમે તે પછી તરત જ મૂલ્યને જોશું જે સ્થિતિને સ્લાઇસ એન્ડ કહેવાય છે અને જ્યાં સુધી આ પદ 0 કરતા મોટો હોય; અને જો તે સ્થિતિનું મૂલ્ય અગાઉની સ્થિતિથી મૂલ્ય કરતાં ઓછું નાનું હોય, તો અમે આ બે અધિકારનું વિનિમય કરીએ છીએ. તેથી, અમે શું કરી રહ્યા છીએ તે છે કે અમે એમ કહી રહ્યા છીએ કે આપણે તેને ફરીથી ડ્રો કરીશું. આપણી પાસે પહેલાથી ક્રમમાં ગોઠવેલ સ્લાઇસ છે 0 થી સ્લાઇસ n બાદ -1, અને આપણી પાસે આ પોઝિશન સ્લાઇસ એન્ડ છે. અમે પછી ધારે છે કે આ છટાણી કરવામાં આવે છે. તેથી, અમે આ મૂલ્ય સાથે સરખામણી કરીએ છીએ અને જો આ નાનું હોય તો આપણે તેનું વિનિમય કરીએ છીએ. હવે જો તમે તેનો આદાન-પ્રદાન કરો તો, આ મૂલ્ય હવે અહીં ગયું છે. હવે, આપણે પાછલા મૂલ્યની સરખામણી કરીએ છીએ, અને જો તે નાનું હોય તો આપણે તેનું વિનિમય કરીએ છીએ. તેથી, ફરી આનો અર્થ એ થાય કે તે એક વધુ સ્થાન પર જાય છે. અમે ફક્ત ત્યાં સુધી જતા રહીએ છીએ જ્યાં સુધી આપણે શોધી કાઢીએ છીએ કે આ સ્થિતિ પર તેની ડાબી બાજુની કિંમત તેનાથી બરાબર અથવા નાની છે. તેથી, આપણે તેને સ્વેપ ન કરવું જોઈએ અને આપણી પાસે તે યોગ્ય સ્થાને છે, તેથી તે આ કરી રહ્યું છે. જ્યાં સુધી તમે ડાબા હાથની અંત સુધી પહોંચી નથી ત્યાં સુધી તમે ટી સરખામણી કરો છો તે મૂલ્ય છે કે તમે હવે તેની ડાબી બાજુએ મૂલ્ય જોઈ રહ્યા છો; તેની ડાબી બાજુની કિંમત સખત મોટી છે, આ વ્યક્તિએ અદલાબદલી કરવી જોઈએ અને પછી તમે સ્થિતિને ઘટાડશો.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 05:29)

ચાલો આને આપણે આ ચોક્કસ ક્રમ પર લખ્યું છે. તેથી, અમે શું કરીએ છીએ અમે શરૂઆતમાં ધારીએ છીએ કે આ વસ્તુ ક્રમમાં ગોઠવાયેલ નથી છે તેથી, અમારી પ્રથમ વસ્તુ અહીં છે. અને તેથી જ્યારે આપણે તેને સોર્ટ કરીએ છીએ, ત્યારે આપણે ફક્ત 74 ની લંબાઈની સૂચિવાળી સૂચિ મેળવી શકીએ છીએ. પછી આપણે આને ધ્યાનમાં લઈએ છીએ અને તેને આ સૂચિમાં દાખલ કરવું આવશ્યક છે 74. તેથી આ 74 કરતાં નાનું છે, તેથી તેને વિનિમય મળે છે અને આપણે હવે નવી સૂચિબદ્ધ સૂચિ 32, 74 અને હવે અમને આ સૂચિમાં 89 દાખલ કરવી જ જોઈએ અને હવે આપણે જોયું કે 89 એ 74 કરતાં મોટી છે, તેથી કંઈ બને

નહીં. આ સૂચિ હવે હું 32 થી 89 થી સોર્ટ કરું છું, હવે અમે આમાં 55 દાખલ કરવાનો પ્રયાસ કરીએ છીએ. આપણે સૌ પ્રથમ આની તુલના કરીએ છીએ, અને એમ કહીશું કે 55 ની કિંમત તેના ડાબા કરતા ઓછો છે, તેથી આપણે અદલાબદલી કરવી જોઈએ. હવે આપણે ફરીથી 55 ની કિંમતને તેના ડાબા સાથે સરખાવીશું, આપણે અદલાબદલી કરીશું. હવે આપણે 55 ની કિંમત તેના ડાબા સાથે સરખાવીશું અને તેમાં કોઈ ફેરફાર નથી. હવે આપણી પાસે 4 ની સૂચિવાળી સૂચિ છે. એ જ રીતે, આપણે 21 ને જમણી બાજુએ લઈશું, અને આપણે તેને 89 સાથે સરખાવીશું; કારણ કે 21 89 કરતા નાની છે, તે સ્વેપ થશે; કારણ કે 21 74 કરતા નાની છે, તે ફરી સ્વેપ થશે; કારણ કે 21 55 કરતા નાની છે, તે સ્વેપ થશે; કારણ કે 21 32 કરતા નાની છે, તે સ્વેપ થશે, પરંતુ હવે સ્થિતિ માફ કરશો અને હવે સ્થિતિ 0 હશે. તેથી, અમે બંધ નહીં કરીએ કારણ કે ડાબી બાજુ કંઈક છે જે મોટા છે, પરંતુ કારણ કે અમારી પાસે ડાબી બાજુ કંઈ નથી.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 07:02)

જો તમને યાદ છે કે અલ્ગોરીધમ કહેવામાં આવે છે કે કાં તો પોઝીટીવ હોવું જોઈએ, તો પોઝિશન 0 કરતા વધારે હોવી જોઈએ અથવા આપણે તેના ડાબા જમણા મૂલ્યની તુલના કરીશું. આ કિસ્સામાં, અમારી પાસે તેના ડાબા માટે કોઈ મૂલ્ય નથી, તેથી અમે બંધ કરીએ છીએ.

(સ્લાઇડ સમય જણાવો: 07:15)

અમે આ કેવી રીતે વિશ્લેષણ કરીએ છીએ? સારું, દરેક રાઉન્ડમાં, આપણે શું કરી રહ્યા છીએ, આપણે લંબાઈ K ના સોર્ટ કરેલ સેગમેન્ટમાં નવું મૂલ્ય દાખલ કરી રહ્યા છીએ. તેથી, લંબાઈ 0 સેગમેન્ટ સાથે શરૂ કરીએ, આપણે તેમાં એક મૂલ્ય શામેલ કરીએ છીએ, આપણે એક લંબાઈના ક્રમની લંબાઈ મેળવી શકીએ છીએ, અમે તે મૂલ્ય દાખલ કરીએ છીએ જેમાં આપણે બે લંબાઈની સોર્ટ ક્રમ મેળવી શકીએ છીએ. જ્યાં સૌથી ખરાબ કિસ્સામાં, જ્યારે અમે શામેલ થઈ રહ્યા છીએ ત્યારે અમારે સેગમેન્ટની શરૂઆતમાં મૂલ્ય લેવાનું રહેશે. સૌથી ખરાબ કિસ્સામાં લંબાઈના કેગની સોર્ટ કરો, k પગલાં લે છે, તેથી ફરીથી આપણી પાસે એ જ પુનરાવૃત્તિ નથી કે જે પસંદગીના સોર્ટ માટે હતી, તે કહે છે કે T નો 1 વત્તા 2 વત્તા 3 અપ n ઓછા 1 છે જે n માં n છે. ઓછા 1 થી 2 જે ઓર્ડર એન ચોરસ છે. તેથી, ફરી યાદ રાખો કે આ n એ ચોરસ છે 2 ઓછા 2 દ્વારા n અને તેથી આ સૌથી મોટો મુદત છે અને તે જ આપણે મેળવીએ છીએ.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 08:10)

ફરી એક વાર જોઈએ કે કેવી રીતે ઇન્જેશન સોર્ટ ખરેખર પિથન ઇન્ટરપ્રીટરમાં કામ કરે છે અને જ્યારે આપણે તેને રન કરીએ છીએ ત્યારે પસંદગી સોર્ટથી કંઈક અલગ દેખાશે. પ્રથમ, ચાલો કોડને જોઈએ. આ તે કોડ છે જે આપણે સ્લાઇડમાં જોયું છે. અમે ફક્ત સ્કેનિંગ સેગમેન્ટ્સને જાળવી રાખીએ છીએ, પોઝિશન પર મૂલ્ય લઈએ છીએ અને તે પોઝિશન સુધી પહેલાથી સોર્ટ કરેલા ક્રમાંકમાં તેને દાખલ કરીએ છીએ. જો આપણે પાચથોન દુભાષિયો શરૂ કરીએ અને આ કાર્ય આયાત કરીએ, તો આપણે પહેલાં ઉદાહરણ તરીકે, લાંબી સૂચિ લઈએ છીએ અને તેને સોર્ટ કરો પછી 1 ને સોર્ટ કરવામાં આવે છે. તેથી, પહેલાં હું ઉતરતા ક્રમમાં હતો.

(સ્લાઇડ સમયનો સંદર્ભ લો: 08:57)

હવે આપણે તેને સોર્ટ કરીએ છીએ, અને હવે હું ચઢતા ક્રમમાં છે.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 09:01)

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 09:03)

હવે આપણે જે કહ્યું તે પહેલા છે કે જો આપણે આશરે 5000 ની લંબાઈ માટે આ કરવાનો પ્રયત્ન કરીએ તો તે ઘણી નાનો અને ખૂબ ધીમી અધિકાર હશે. તેથી, તમે જોઈ શકો છો કે તે લાંબો સમય લે છે અને તે છે કારણ કે InsertionSort, તે ફરીથી સ્કવેર સોર્ટ ક્રમ છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 09:24)

તેથી, જો કે તે આખરે તે લાંબા સમય લે છે,

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 09:28)

પરંતુ અહીં થોડો તફાવત છે. તેથી, ધારીએ કે આપણે તે બીજી રીતે કરીએ, ધારીએ છીએ કે આપણે પહેલેથી જ છટણી કરેલી સૂચિ લઈએ છીએ, અને હવે અમે તેને સોર્ટ કરવા કહીએ છીએ, પછી તે તરત જ પાછો આવે છે. આ કેમ હોવું જોઈએ તે વિશે વિચારવું જોઈએ કે હવે શું થઈ રહ્યું છે તે સૂચિમાં પહેલાથી ક્રમમાં છે. તેથી, જ્યારે આપણે કોઈ પણ પદ પર મૂલ્ય લેવાનો પ્રયાસ કરીએ છીએ અને તેને ડાબી તરફ ખસેડીએ છીએ, ત્યારે તે તરત જ શોધે છે કે તેની ડાબી બાજુનું મૂલ્ય તેના કરતા ઓછું છે, તેથી કોઈ સ્વેપ થતી નથી. તેથી, દરેક શામેલ પગલું માત્ર એક પુનરાવૃત્તિ લે છે. શામેલ થવાનું બંધ કરવા માટે પ્રથમ ઘટકની બહારની કોઈ પણ વસ્તુમાં જવાની જરૂર નથી. તેથી, વાસ્તવમાં જો આપણે 10,000 કે 100000 જેવા મોટા મૂલ્ય પણ લઈએ તો આ કામ કરવું જોઈએ. સોર્ટિંગ સોર્ટ કરો જ્યારે તમારી પાસે પહેલાથી સોર્ટ કરેલી સૂચિ ખૂબ ઝડપી હશે કારણ કે શામેલ પગલું તાત્કાલિક છે, જ્યારે આ પસંદગી સોર્ટ સાથે થતું નથી પસંદગીના ક્રમમાં, દરેક પુનરાવર્તનમાં, આપણે શ્રેણીમાં લઘુત્તમ મૂલ્યને શોધી કાઢવું પડશે અને અનુક્રમ શું જુએ છે તેના કોઈ પહેલાંના જ્ઞાન સાથે તેને હંમેશા શરૂઆતથી અંત સુધી સ્કેન કરે છે. પસંદગી સોર્ટ માટેનો સૌથી ખરાબ કેસ, ઇનપુટ પહેલેથી જ સોર્ટ કરેલ છે કે નહીં તે ધ્યાનમાં લીધા વગર થશે; જ્યારે સોર્ટિંગ સોર્ટ જો સૂચિને સોર્ટ થાય છે, તો શામેલ પગલું ખૂબ જ ઝડપી હશે, અને તેથી જો તમે વાસ્તવમાં મોટી વસ્તુઓને સોર્ટ કરી શકો છો તે અર્થમાં સોર્ટિંગ સોર્ટ પસંદગી સોર્ટ કરતાં વધુ સારી રીતે વર્તે શકે છે, તેમ છતાં તે બન્નેને તકનીકી રીતે સૌથી ખરાબ કેસમાં ઓર્ડર એન સ્કવેર્ડ પ્રકારની છે.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગારિધમ્સ ઇન પાયથોન
પ્રો. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટીટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 03 લેક્ચર - 18

રિકર્શન આ અઠવાડિયાની છેલ્લી લેક્ચર માટે, અમે ફરી યાદ આવતી કાર્યોને જોશું.

(સ્લાઇડ સમયનો સંદર્ભ લો: 00:05)

રિકર્સિવ ફંક્શન્સ સામાન્ય રીતે અમે પ્રત્યક્ષ પ્રમાણિત વ્યાખ્યાઓને આધારે છે. તેથી, અંકગણિત ઘણા વિધેયોમાં સ્વાભાવિક રીતે અર્થપૂર્ણ રૂપે એક સચોટ રીતે વ્યાખ્યાયિત કરવામાં આવે છે. ચાલો આને કેટલાક ઉદાહરણો દ્વારા સમજાવીએ. કાર્યનું પ્રથમ અને સૌથી સામાન્ય ઉદાહરણ નિર્વિવાદપણે વ્યાખ્યાયિત કરે છે, કારણદર્શી કાર્ય છે. તેથી, આપણે કહીએ છીએ કે શૂન્ય ફેક્ટોરિયલ 1 છે અને તે પછી, આપણે કહીએ છીએ કે એન ફેક્ટોરિયલ એન દ્વારા ગુણાકાર કરીને n બાદ 1 ફેક્ટોરિયલમાંથી મેળવી શકાય છે. યાદ રાખો કે n ફેક્ટોરિયલ એ કંઈ નથી પણ n માં n ઓછા 1 માં n ઓછા 2 ઉત્પાદન બધી રીતે નીચે 1 છે. તેથી, આપણે શું નિરીક્ષણ કરી રહ્યા છીએ એ છે કે n પછી, શું દેખાય છે તે n બાદ 1 ફેક્ટોરિયલ તરીકે ફરીથી લખી શકાય છે. અનુક્રમિત n ઓછા 1 ફેક્ટોરિયલ મૂલ્ય n દ્વારા ગુણાકાર કરીને n ફેક્ટોરિયલ સુધી વિસ્તારી શકાય છે. તેથી, આપણે આ અન્ય કાર્યો માટે પણ કરી શકીએ છીએ. જ્યારે હું મીટર વખત કહેતો હોય ત્યારે તમને યાદ આવે કે તમે તે ગુણાકારને ખરેખર પુનરાવર્તિત કરી શકતા નથી, તેનો અર્થ એમ વત્તા મીલ મીટર મીટર મીટર મીટર, એન વખત. તો, આપણે આ કેવી રીતે વ્યાખ્યાયિત કરીએ છીએ તે પ્રમાણે આપણે એમ કહીએ છીએ કે m વખત 1 એ ફક્ત મીટર જ છે અને મીટર વખત n વત્તા 1 મી છે અને અનુક્રમે ગુણાકારને n નો ઉપયોગ કરવો. આપણે સમકક્ષ આ લખી શકીએ છીએ. જો તમે પહેલાના કેસ સાથે સેમટ્રિક હોવ તો મીટર વખત n એ મીટર વત્તા મીમ ટાઇમ n બાદ 1, એક જ વસ્તુ. તમે જે કહી રહ્યા છો તે એમ છે કે તમે m વખત n ની દ્રષ્ટિએ n નો વ્યક્ત કરી શકો છો અને પછી તેને ઉમેરી રહ્યા છે. તેથી, આ બન્ને કિસ્સાઓમાં અમારી પાસે શું છે કે આપણી પાસે બેઝ કેસ છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 01:50)

અમારી પાસે 0 ફેક્ટોરિયલ અથવા એમ ટાઇમ્સ 1 છે, જ્યાં કિંમતો અમને સ્પષ્ટ રીતે આપવામાં આવે છે અને પછી, આપણી પાસે એક પ્રત્યક્ષ પગલું છે જ્યાં n ની એક n ની વ્યાખ્યામાં વ્યાખ્યાયિત કરવામાં આવે છે n અને 1 સામાન્ય રીતે, તે વધુ નાની દલીલોની દ્રષ્ટિએ પણ વ્યાખ્યાયિત કરી શકાય છે. તેથી, તેનું એક ઉદાહરણ ફિબોનાકી શ્રેણી છે. જો તમે ફિબોનાકી શ્રેણી જોયું હોય, તો ફિબોનાકી શ્રેણી 1 2 3 5 થી શરૂ થાય છે અને તેથી આગળ અને આ અગાઉના બે મૂલ્યો લઈને અને પછી ઉમેરીને મેળવી શકાય છે. તેથી, ફિબોનાકી માટેનું સામાન્ય નિયમ એ છે કે પ્રથમ મૂલ્ય બીજો મૂલ્યની બરાબર 1 છે અને n ની બીજો કિંમત ફિબોનાકી પછી n એ બાદબાકી 1 વત્તા ફિબોનાકી નો n ની 2 ની બાદબાકી છે. સામાન્ય રીતે પુનરાવર્તિત અથવા પ્રત્યક્ષ પ્રમાણમાં વ્યાખ્યા નાના ઇનપુટ્સ માટે ફંક્શનના 1 કે નાના મૂલ્યોની દ્રષ્ટિએ n માટે મૂલ્ય વ્યક્ત કરી શકે છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 02:49)

અસ્પષ્ટ વ્યાખ્યામાં આપણી રુચિ એ છે કે એક સવિમુખ્ય વ્યાખ્યાને સ્વૈચ્છિક ગણતરી તરીકે કુદરતી પ્રતિનિધિત્વ છે. ચાલો આપણે ફેક્ટોરિયલ જુઓ. અહીં વ્યાખ્યાયિત કરવામાં આવે છે તે કારણદર્શી તરીકે ખૂબ જ સરળ પાયથોન છે, તે મૂલ્ય n ચકાસે છે અને કહે છે કે n એ 0 છે, પછી વળતર 1 અન્યથા ફક્શનલ n બાદ -1 ની ગણતરીની કિંમત n વખત પરત કરે છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 03:33)

આ ખૂબ જ સ્પષ્ટ છે કે આપણે પહેલાં કહ્યું છે; તે શૂન્ય ફેક્ટોરિયલ 1 કહે છે અને અન્યથા જો n એ 0 નથી, એન ફેક્ટોરિયલ n વખત n ઓછા 1 ફેક્ટોરિયલ છે. તેથી, આ બરાબર છે કે આપણે અગાઉ ફરી યાદ આવવું ગણતરી તરીકે ભાષાંતર કર્યું તે પહેલાં આપણે લખ્યું હતું. ગુણાકાર માટે આપણે તે જ કહી શકીએ છીએ. તમે કહી શકો છો કે જો તમે n દ્વારા મલ્ટીપ્લાય કરવા માંગો છો, જો n એ 1 છે, તો આપણે મીટર પરત કરીશું નહિ તો અમે m નો n નો બાદ 1 નો ગુણાકાર કરવાના પરિણામને મીટર ઉમેરીશું. ફરીથી અમે લખ્યું હતું કે m વખત 1 એ n અને m વખત પહેલાં n એ મીટર વત્તા મીટર સમય n બાદ 1. જો તમારી પાસે પ્રત્યક્ષ પ્રમાણભૂત વ્યાખ્યા હોય, તો તે તરત જ ફરી યાદ આવવું કાર્યને ઉભરી કરે છે જે સમાન વ્યાખ્યાઓનું ગાણિત કરે છે. આ ફાયદો એ છે કે આપણે એટલો સમય વિતાવવાની જરૂર નથી કે આ કાર્ય યોગ્ય છે કારણ કે તે સીધા પ્રત્યક્ષ પ્રમાણમાં વ્યાખ્યાઓ દર્શાવે છે, કાર્યની ગણતરીની અમે ગાણિતિક વ્યાખ્યાઓ છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 04:14)

હવે, શું ઓછું સ્પષ્ટ થઈ શકે છે કે આપણે યાદી જેવી માળખા માટે સમાન વસ્તુ કરી શકીએ છીએ. સૂચિમાં એક સચોટ માળખું હોય છે જે એક સમયે એક ઘટક ઉમેરીને ખાલી સૂચિમાંથી એક યાદી બનાવવામાં આવી શકે છે. તેથી, અમે એક પગલું વિસર્જન કરવું વિચારી શકીએ છીએ, તે પગલું ઉલટાવી શકીએ છીએ. તેથી, અમે એક સમયે એક ઘટક ઉમેરીને એક ખાલી સૂચિમાંથી એક યાદી બનાવવાનું શરૂ કરીએ છીએ કે અમે તેમને ડાબી બાજુએ ઉમેરીએ છીએ અમે છેલ્લો એલિમેન્ટ ઉમેર્યું છે અને આપણે બીજો છેલ્લો એલિમેન્ટ ઉમેર્યું છે. પરંતુ ઊલટી આપણે કહી શકીએ કે એક સૂચિ આપેલ છે જે આપણે પ્રથમ ઘટક કહીને તે પ્રથમ સમાંતર અને પ્રથમ તત્વને દૂર કર્યા પછી બાકીની સૂચિને જોઈને તેને સડવું કરી શકીએ છીએ જે એક ઘટક હોય. આ અમને આપણો ઇન્ડક્શન આપે છે અમે એક નાના માળખું ધરાવીએ છીએ જેના પર આપણે કોઈ કાર્યને વ્યક્ત કરવાનો પ્રયત્ન કરી શકીએ અને પછી આપણે તેને મોટા ભાગની કિંમત મેળવવા માટે પ્રથમ એલિમેન્ટ સાથે ભેગા કરી શકીએ. તેથી, આપણી પાસે બેઝ કેસ હશે જ્યાં ફક્શન ખાલી યાદી માટે અથવા સાદી 1 ની સાદી સૂચિ માટે વ્યાખ્યાયિત કરવામાં આવે છે અને 1 ની પ્રત્યક્ષ પગલું એફ માં નાના sublists ની દ્રષ્ટિએ વ્યાખ્યાયિત કરવામાં આવશે.

(સ્લાઈડ સમયનો ઉલ્લેખ કરો: 05:15)

ફરીથી આ અમુક સરળ વ્યાખ્યાઓ દ્વારા સમજી શકાય છે, આપણે લેપની લંબાઈ ગણતરી કરવા માગીએ છીએ. વેલ બેઝ કેસ જો સૂચિ ખાલી હોય તો તેની પાસે શૂન્ય લંબાઈ છે. જો 1 બરાબર 0 - 1 ખાલી યાદીની બરાબર છે, તો આપણે 0 પરત કરીશું; નહિ તો આપણે શું કરીએ છીએ કે આપણી પાસે સંખ્યાબંધ મૂલ્યો છે. તેથી, આપણે આ પ્રથમ મૂલ્ય ખેંચવા અને અમે કહીએ છીએ કે તે એક લંબાઈમાં ફાળો આપે છે અને હવે અમે બાકીના લંબાઈની ગણતરી કરીએ છીએ, જમણે. આપણે 1 વત્તા સ્થાને સ્થાને શરૂ થતા સ્લાઇસની લંબાઈ પરત કરીએ છીએ. આ લંબાઈની પ્રત્યક્ષ પ્રમાણમાં વ્યાખ્યા છે જેનો પુનરાવર્તિત કાર્યમાં અનુવાદ થાય છે અને આ ફક્શનના માળખાને જોઈને એક વાર ફરી અનુવાદ થાય છે, તે ખૂબ જ સ્પષ્ટ છે કે તે લંબાઈને યોગ્ય રીતે ગણતરી કરે છે કારણ કે આ બરાબર છે કે તમે લંબાઈને કેવી રીતે નક્કી કરો છો.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 06:00)

હવે અહીં અન્ય કંક્શન છે જે લંબાઈ ગણતરી કરવાને બદલે તેના જેવું જ કંઈક કરે છે, તે તમામ નંબરોને ઉમેરે છે કે જે સૂચિ નંબરોની સૂચિ છે. ફરી જો મારી પાસે કોઈ સંખ્યા ઉમેરવા નથી, જો મારી પાસે ખાલી યાદી હોય, તો આ રકમ 0 હશે કારણ કે મારી પાસે આ રકમમાં મૂકવાનો કંઈ નથી. પ્રશ્નની બીજી બાજુએ, જો મારી પાસે કેટલીક સંખ્યાઓ ઉમેરવા માટે હોય, તો આ રકમમાં પ્રથમ મૂલ્ય લેવું અને બાકીનામાં તે ઉમેરીને છે. જો મારી પાસે x_1, x_2 ને x_n સુધીની યાદી છે, તો હું કહી રહ્યો છું કે આ x_1 વત્તા x_2 ની $x \times x$ નો સરવાળો છે. હું આ રકમ ફરી યાદ આવવું અથવા આગ્રમી કોલ દ્વારા મેળવી શકું છું. પછી, હું આ મૂલ્યને ફક્ત તે જ ઉમેરું છું

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 06:45)

નિશાની સોર્ટ જે આપણે જોયું છે તેમાં ખરેખર સરસ વ્યાખ્યા છે. જો અમારી પાસે કદ 0 અથવા કદ 1 ની સૂચિ હોય, તો તે પહેલેથી જ સોર્ટ કરેલું છે. આપણે કાંઈ કરવાની જરૂર નથી, તેથી આ બેઝ કેસ બને છે. પ્રશ્નની બીજી બાજુએ, જો અમારી પાસે બે અથવા વધુ લંબાઈની સૂચિ હોય, તો અમે પ્રારંભિક રીતે શરૂઆતથી સ્લાઇડને અનુરૂપ કરીએ છીએ, પરંતુ છેલ્લી સ્થિતિ સિવાય આ 0 ની લંબાઈની યાદીમાંથી બાદબાકી 1 ની છે, તો પછી આપણે છેલ્લી પોઝિશન લઈએ છીએ અને પછી આ 1 ઓછા થવું જોઈએ. તેથી, આપણે છેલ્લી પોઝિશન પર વેલ્યુ લઈએ છીએ અને આપણે તે છટાણી સ્લાઇડમાં દાખલ કરીએ છીએ. અમે છેલ્લી પોઝિશનને બાદ કરતાં અનુગામી ક્રમમાં ગોઠવેલી સ્લાઇડમાં છેલ્લી પોઝિશન દાખલ કરીએ છીએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 07:36)

અહીં પાયથોનમાં દાખલ કરવાના સોર્ટની પુનરાવર્તિત વ્યાખ્યા છે. પાયથોન અથવા અન્ય કોઈ વસ્તુની કુદરતી વસ્તુ કહે છે કે અમે શ્રેણીને ક્રમબદ્ધ કરવાના પરિણામ પર છેલ્લું સ્થાન દાખલ કરવું છે, પરંતુ છેલ્લું સ્થાન સિવાય, પરંતુ જે રીતે અમે અમારી નિવેશ સોર્ટ લખ્યું છે; આ કાર્ય સૂચિ પરત કરતું નથી. તે જગ્યાએ આ પ્રકારની ગોઠવે છે આ કોલમાં સાચો પ્રકાર ન હોવો જોઈએ કારણ કે શામેલ છે એક અનુક્રમ અને સ્થાન લેશે અને આ સ્થાન પર તેની ડાબી બાજુએ આ મૂલ્ય શામેલ કરો. તેથી, આપણે તેને બે અલગ વસ્તુઓ તરીકે લખીએ છીએ. સૌ પ્રથમ અમારી પાસે એકંદરે નિવેશ સોર્ટ છે જે અનુક્રમ લે છે અને તે આ સહાયક કાર્યને કોલ કરશે જે કહે છે: આ પદ સુધી આ ક્રમને સોર્ટ કરો. તેથી, ઇસ્ટ એ સ્લાઇડને 0 થી લઈને ઘાત 1 સુધી ફેરવે છે. તેથી, આર્ટ ટુ કહે છે? ઇયરસ્ટોટ તપાસ કરે છે જો કર્સ કે 0 હોય અથવા કે 1 હોય તો તે મૂળભૂત કેસ છે. જો હું ઓપ્રથમ પોઝિશન સુધી ભરવા અથવા હું કોઈ વસ્તુને સોર્ટ કરતો નથી, જમણે - પછી હું ફક્ત ક્રમ પાછો આપીશ. આ મને 0 થી k ઓછા 1 થી સોર્ટ કહે છે. જો તે 0 છે, તો મારી પાસે ખાલી યાદી છે. જો મારી પાસે 0 થી 1 હોય, તો મારી પાસે એક પદની સૂચિ છે, તે ફક્ત ત્યારે જ છે કે મારી પાસે 0 થી 2 ની પાસે ઓછામાં ઓછા બે તત્વો છે. અને જો આમ હોય, તો હું શું કરું છું હું સોર્ટ કરો k minus 1 positions અને હું આ ક્રમમાં છેલ્લી સ્થિતિ દાખલ કરું છું. શામેલ કરે છે? જ્યારે આપણે નિયમિત નિવેશ સોર્ટ કર્યું ત્યારે અમે જે કર્યું તે બરાબર કરે છે. તે પોઝિશન વેરીએબલ અથવા નામને છેલ્લી પોઝિશન પર સેટ કરે છે અને ડાબેથી જ ચાલે છે અને સ્વિંગ રાખે છે. જ્યાં સુધી તે ડાબા હાથની હાંસિયા સુધી પહોંચી ન હોય અને લાંબા સમય સુધી તે ડાબી બાજુ કંઈક શોધે છે જે તમે જે જોઈ રહ્યા છો તે કરતાં સખત મોટી છે. તેથી, આ આપણે અગાઉ કર્યું તે છે. આ ભાગ, જે આ ફરી યાદ આવવું કોલ છે, તે આ પદ સુધી ક્રમને સોર્ટ કરે છે, તે જ આર્ટોર્ટનો ઉપયોગ કરે છે પરંતુ ઇન્ડેક્સને k થી k minus 1 માં બદલી દે છે અને પછી વર્તમાન શ્રેણીને આ ક્રમમાં દાખલ કરો.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 09:51)

તેથી, પહેલાં આપણે પાચથોન માં આ રન કરીએ. અહીં કોડ isort rec dot py છે જેમાં નિવેશ સોર્ટની આ રિકર્સિવ અમલીકરણનો સમાવેશ થાય છે. તેથી, જો હું હવે આ આયાત કરું છું, તો પહેલાની જેમ આપણે કહીએ છીએ કે એ 500 મૂલ્યોની શ્રેણી છે, ઉતરતા ક્રમમાં, તો પછી જો આપણે તેમાં સામેલ થવા માટે અરજી કરીએ, તો પછી ચડતા ક્રમનો 1 થી 500 નિર્માણ થાય છે. કે n સ્કેર્ડ સોર્ટ માટે, આપણે મોટું મૂલ્યો જોવું જોઈએ. ધારો કે અમે શ્રેણી 1000 લે છે, હવે જો આપણે શ્રેણી 1000 લે છે, પછી કંઈક આશ્ચર્યજનક થાય છે. આપણને પાચથોન તરફથી એક ભૂલ સંદેશો મળી રહ્યો છે કે તે આને સોર્ટ કરી શકતો નથી કારણ કે તે મહત્તમ રિકર્ઝન ઊંડાઈ તરીકે ઓળખાય છે તેથી, જ્યારે આપણે ફરી યાદ આવવું કોલ કરીએ ત્યારે શું થાય છે કે વર્તમાન કાર્ય સસ્પેન્ડ છે. અમે લંબાઈ 1000 કંઈક સોર્ટ કરવાનો પ્રયાસ કરીએ. તે પ્રથમ 999 ના ક્રમમાં સોર્ટ કરવાના પરિણામે 1000 મી તત્વને દાખલ કરવાનો પ્રયાસ કરશે. તેથી, અમે પ્રથમ કોલને સસ્પેન્ડ કરીએ છીએ અને 999 ને સોર્ટ કરવાનો પ્રયાસ કરીએ છીએ. આમાં 999 તત્વને તત્વમાં દાખલ કરવાનો પ્રયાસ કરવામાં આવશે. પ્રથમ સોર્ટ 998. તેથી, તે સસ્પેન્ડ કરશે અને પ્રથમ 998 ને સોર્ટ કરવાનો પ્રયત્ન કરશે. દરેક વખતે અમે ફરી યાદ આવવું કોલ કરીએ છીએ, વર્તમાન કાર્ય સસ્પેન્ડ થઈ ગયું છે અને નવું કાર્ય શરૂ થયું છે. તેથી, તેને રિકર્ઝનની ઊંડાઈ કહેવામાં આવે છે. અમે આ પ્રક્રિયા કરી રહ્યા છીએ ત્યારે અમે કેટલા કાર્યો નિલંબિત કર્યા છે? હવે કેટલીક અન્ય ભાષાઓમાં વિપરીત, પાચથોન આની મર્યાદા લાદશે અને મર્યાદા લાદશે કારણ કે આપણે જોઈ શકીએ છીએ તે 1000 કરતાં ઓછું છે કારણ કે અમે આ વિશેષ પદ્ધતિનો ઉપયોગ કરીને 1000 ની સૂચિને સોર્ટ કરી શકતા નથી. તો, આપણે આ કેવી રીતે મેળવી શકીએ? ઠીક છે, સૌ પ્રથમ, ચાલો જોઈએ અને જુઓ કે આ મર્યાદા શું છે. અમે જાણીએ છીએ કે આપણે 1000 ને સોર્ટ કરી શકતા નથી, પરંતુ આપણે 500 ને સોર્ટ કરી શકીએ છીએ. તેથી, આપણે દાખલા તરીકે 750 સોર્ટ કરી શકીએ છીએ, તે દર્શાવે છે કે આપણે 750 ને સોર્ટ કરી શકીએ છીએ. હમણાં, તે ચાલુ થશે, દાખલા તરીકે, આપણે 900 સોર્ટ કરી શકીએ છીએ. અમે ખરેખર આ મર્યાદાને આપણે અગાઉ કર્યું તે કરીને કરી શકીએ - દ્વિસંગી શોધ. હું હળવી રાખી શકું છું, જો કે મેં 750 પછી સખત ધોરણ ના કર્યું હોત. મને ખબર છે કે તે 1000 પર નિષ્ફળ જાય છે, તે 750 પર નિષ્ફળ થતું નથી, મને 875 કરવાનો પ્રયત્ન કરવો જોઈએ, પણ હું તમને બાઈનરી શોધને બચાવીશ અને હું તમને બતાવીશ કે જો હું ઉપયોગ કરું 997, પછી તે કામ કરશે, પણ હું 998 નો ઉપયોગ કરું છું, પછી તે નિષ્ફળ જશે. તેથી, ક્યાંક 998 ની આસપાસ રિકર્ઝનની મર્યાદા છે કે જે પાચથોન મૂળભૂત રીતે ઉપયોગ કરે છે. હવે, સદભાગ્યે આની આસપાસ જવાનો માર્ગ છે. તેથી, તમે આ રિકર્ઝનની મર્યાદાને સેટ કરી શકો છો અને તમે જે રીતે કરો છો તે નીચે મુજબ છે. તમારે સૌપ્રથમ વિધેયોનું એક ટોળું આયાત કરવું પડશે જે આયાત સિસ્ટમ્સને કહીને સિસ્ટમ આધારિત કાર્યો તરીકે ઓળખાય છે અને પછી, SYS માં એક સેટ રિકર્ઝન સીમા કહેવાય છે અને અમે આને 10000 કરતાં વધુ મૂલ્યની કિંમત તરીકે સેટ કરી શકીએ છીએ. હવે, જો આપણે દાખલા તરીકે, 1000 ની સૂચિને સોર્ટ કરવા માટે પૂછો, પછી તે અમને ભૂલ આપતું નથી એ જ રીતે, હું 5000 પણ કહી શકું છું કારણ કે 5000 પણ માત્ર એક જ પ્રકારની મર્યાદા બનાવશે કારણ કે તે 10000 ની નીચે છે. યાદ રાખો કે આમાં આપણે રિકર્સિવ બરાબર સંખ્યાઓની સંખ્યા કરી રહ્યા છીએ કારણ કે ત્યાં તત્વો છે કારણ કે અમે શામેલ કરીએ છીએ, દાખલ કરીએ છીએ, શામેલ કરો અને પ્રત્યેક નિવેશને આવશ્યકપણે તેના ડાબી બાજુએ વસ્તુઓને સોર્ટ કરવું આવશ્યક છે. એટલા માટે આપણે પુનરાવર્તિત પુનરાવર્તનોની સ્ટેક મેળવીએ છીએ, પરંતુ યાદ રાખવું તે વસ્તુ એ છે કે મૂળભૂત રીતે, પાચથોન ને પુનરાવર્તિત પુનરાવર્તિત સંખ્યા પર બંધાયેલ છે જે 1000 કરતાં ઓછું છે, જો તમે તેને રિકર્ઝન સેટ કરીને બદલી શકો છો સ્પષ્ટપણે મર્યાદિત છે, પરંતુ પાચથોન તમને તેને અમર્યાદિત મૂલ્ય પર સેટ કરવાની મંજૂરી આપતું નથી તમારે તેને ઉપરથી બંધ રાખવું જોઈએ. તેથી, તમે કહી શકતા નથી કે રિકર્ઝનને લાંબા સમય સુધી ચલાવવાની જરૂર છે. તમે આપેલી ઇનપુટ પર રિકર્ઝન ખરેખર કેટલી લેશે તે અંદાજ આપવો પડશે, જેથી તમે આને સ્પષ્ટ રૂપે સેટ કરી શકો. હવે, પીચથન આ કારણ છે કારણ કે તે એક ભૂલભરેલી ગણતરી માટે રિકર્સિવ ગણતરીને સમાપ્ત કરવાનો પ્રયત્ન કરવા માંગે છે. ફરી યાદ આવવું ગણતરી સાથેની એક સામાન્ય ભૂલ, અમે લૂપ્સ માટે કહીએ છીએ તે એક બીટ છે, જો આપણે ક્યારેય શરત ખોટી ન બનાવીએ, તો જ્યારે લૂપ કાયમ અમલ કરશે. એ જ રીતે જો આપણે બેઝ કેસ યોગ્ય રીતે સેટ ન કરીએ,

તો ફરી યાદ આવવું ગણતરી પણ કાયમ માટે જઈ શકે છે. જે રીતે પાયથોન આને બંધ કરે છે અને તમને જવા અને કોડનું પરીક્ષણ કરવા માટે દબાણ કરે છે તે ચોક્કસ ઊડાણની બહાર રિકર્ડનની મર્યાદા હોવાને કારણે, તે કોડને અમલમાં મૂકવાનો ઇન્કાર કરશે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 14:51)

તો, આપણે શું જોયું છે કે આ રિકર્ડનની મર્યાદા છે અને આપણે આ sys મોડ્યુલ આયાત કરીને અને sys ની અંદર સેટ રિકર્ડન મર્યાદા ફંક્શનને સુયોજિત કરીને જાતે જ એકત્ર કરી શકીએ છીએ.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 15:00)

તો આપણે કેવી રીતે રિકરિવ નિવેશ સોર્ટની જટિલતાનું વિશ્લેષણ કરીશું? પહેલાંની જેમ, આપણે 'ટી' નો ઉપયોગ લંબાઈની n ના ઇનપુટ પર દાખલ કરવાના સમયને દાખલ કરવા માટે કરીએ છીએ. ઉચ્ચતમ સ્તર પર નિવેશ સોર્ટ બે પગલાંઓ ધરાવે છે. આપણે પ્રથમ લંબાઈ n બાદ -1 ની પ્રારંભિક સ્લાઇડને સોર્ટ કરવી પડશે અને વ્યાખ્યા દ્વારા, આને n ની બાદબાકી 1 મળશે અને ત્યારબાદ, આપણે સૌથી ખરાબ કેસમાં n બાદ 1 પગલાંની જરૂર પડશે જેથી છટણી કરેલ સ્લાઇડમાં છેલ્લી સ્થિતિ દાખલ કરી શકીએ. આનાથી આપણે પુનરાવૃત્તિ શું કહીએ છીએ. અમે આ જોયું જ્યારે અમે કેવી રીતે દ્વિસંગી શોધ વિશ્લેષણ કરવા માટે શોધી રહ્યા હતા જે ફરી યાદ આવવું અલ્ગોરિધમનો હતા આપણી પાસે સામાન્ય રીતે ટી n એ n બાદ 1 વત્તા T નો n બાદ 1 અને ટી 1 છે જ્યારે આપણે બેઝ કેસ પર આવીએ છીએ 1. દ્વિસંગી શોધ સાથે, આપણે પુનરાવર્તન વિસ્તરણ અથવા અનૈલું કરીને આને હલ કરી શકીએ છીએ. તો, આપણી પાસે T નો n એ n બાદ 1 વત્તા T n ની 1 છે. જો આપણે T n બાદ 1 લે અને એ જ વ્યાખ્યાઓ લાગુ કરીએ, તો આપણને n ની 2 વત્તા T n ની 2 ઘાત મળશે. પછી, આપણે T નો બાદ 2 અને એ જ વ્યાખ્યા લાગુ કરો, આપણને n ઓછા 3 અને T n ઓછા 3 મળે છે અને આ ચાલુ રાખશે જ્યાં સુધી આપણે n ની સંખ્યા ટી બરાબર 1 ની બરાબર નથી. બીજા શબ્દોમાં જ્યારે n એ n પગલાંઓ પછી n બને છે, ત્યારે આપણે 1. તેથી, આપણી પાસે n બાદ 1 વત્તા n ની 2 ઘાત 2 ની નીચે હશે જે આપણે દાખલ કરવાના ક્રમાંકના પુનરાવર્તન સંસ્કરણ - n માં n ને 1 ની 2 ઘાત બરાબર છે અને આ ક્રમાંક સ્કેલ છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 16:41)

અમે બે ઓર્ડર એન સ્કેલ સોર્ટિંગ એલ્ગોરિધમ્સ જોયા છે જે બંને ખૂબ જ કુદરતી સાહજિક એલ્ગોરિધમ્સ છે જે અમે હાથથી કરીએ છીએ જ્યારે અમે સોર્ટ કરવાના કાર્યો કરીએ છીએ - પસંદગી સોર્ટ અને દાખલ સોર્ટ. અમે જોયું છે કે જો આ બંને મોટી સંખ્યામાં n ના હોય અને એટલા મોટા ન હોય તો બંનેમાં સામાન્ય રીતે કામ નહીં કરે; જો અમારી પાસે 5000 ના મૂલ્યના મૂલ્યો છે. પણ આપણે પહેલાના બે પ્રવચનોમાં જોયું છે કે દાખલ સોર્ટ ખરેખર પસંદગી સોર્ટ કરતાં સહેજ વધુ સારી છે કારણ કે પસંદગી સોર્ટ, સૌથી ખરાબ કેસ હંમેશાં હાજર છે કારણ કે આપણે હંમેશા આખા સ્લાઇડને સ્કેન કરવું પડશે યોગ્ય સ્થાનમાં જવા માટે ન્યૂનતમ મુલ્ય એલિમેન્ટ શોધો જ્યાં નિવેશ સોર્ટ જલદી બંધ થઈ જશે કારણ કે તે કોઈ યોગ્ય ક્રમમાં છે તે શોધે છે. તેથી જો અમારી પાસે એક સૂચિ છે જે પહેલાથી જ સોર્ટ કરેલી છે, તો દાખલ સોર્ટ વાસ્તવમાં n સ્કેલ કરતાં વધુ સારી કામગીરી કરશે, પરંતુ અમે તેના પર આધાર રાખી શકતા નથી, અને કોઈપણ રીતે અમે સૌથી ખરાબ કેસની જટિલતાને ગણતરીમાં લઈએ છીએ. તેથી, અમારે આ હકીકતને ધ્યાનમાં લેવી પડશે કે સામાન્ય રીતે 5000 ઘટકો કરતાં મોટી યાદી માટે આ બન્ને ઇચ્છાઓ ખૂબ જ સારી રીતે કામ કરશે નહીં. આગામી અઠવાડિયે આપણે શું જોશું તે છે કે ત્યાં વધુ કાર્યક્ષમ સોર્ટિંગ એલ્ગોરિધમ્સ છે જે અમને મોટાભાગની યાદીઓને સોર્ટ કરવા માટે પરવાનગી આપે છે, કારણ કે પસંદગી સોર્ટ અથવા ઇન્સર્શન સોર્ટનો ઉપયોગ કરીને આપણે સોર્ટ કરી શકીએ છીએ.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રો. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
અઠવાડિયું - 04 લેક્ચર - 01 મર્જ કરો સોર્ટ કરો

(સ્લાઇડ ટાઇ નો ઉલ્લેખ કરો: 00:02)

ગયા અઠવાડિયે, અમે બે સરળ સોર્ટિંગ એલ્ગોરિધમ્સ જોયાં, પસંદગી સોર્ટ અને ઇન્સ્ટર્શન સોર્ટ. આ આકર્ષક હતા, કારણ કે તેઓ માર્ગદર્શિકાને અનુસરતા હતા જેમાં આપણે હાથ દ્વારા વસ્તુઓને સોર્ટ કરીશું. બીજી બાજુ, અમે આનું વિશ્લેષણ કર્યું છે કે સૌથી ખરાબ કેસની જટિલતા એ ઓર્ડર n સ્ક્વેર્ડ છે જ્યાં n એ છટણી કરવા માટેની ઇનપુટ સૂચિની લંબાઈ છે. અને કમનસીબે, n સ્ક્વેર્ડ સોર્ટિંગ એલ્ગોરિધમ્સ આશરે 5000 થી એન માટે અયોગ્ય છે, કારણ કે તે માત્ર ખૂબ લાંબો સમય લેશે અને બીજી બાજુ, 5000 એ ખરેખર નાની સંખ્યા છે જ્યારે અમે વાસ્તવિક ડેટા સાથે વ્યવહાર કરીએ છીએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 00:37)

ચાલો આપણે બધા સાથે એક જુદી જુદી વ્યૂહરચનાનું પરીક્ષણ કરીએ. ધારો કે અમારી પાસે એવી ઉદાહરણ છે જ્યાં તમે સહાયક શિક્ષણ આપતા હતા અને તમે પ્રશિક્ષક માટે જવાબના કાગળોને સોર્ટ કરવાના હતા અને એવું માનતા હતા કે પ્રશિક્ષક એક ઉપદેશક નથી, પરંતુ બે સહાયક સહાયકો છે. અને આ કામ બે શિક્ષણ સહાયકોને વહેંચવામાં આવે છે, તેથી દરેકને કાગળને છૂટા કરીને, અલગથી સોર્ટ કરવા અને પાછા આવવા કહેવામાં આવે છે અને પછી પ્રશિક્ષકે આ બે યાદીઓને એકસાથે મૂકવી પડશે. બીજા શબ્દોમાં કહીએ તો, તમે શરૂઆતમાં એરેને વહેંચી શકો છો, બે ભાગમાં ક્રમમાં ગોઠવેલ એરે અથવા સૂચિ અને પછી તમે આ બે ભાગોને બે જુદા જુદા લોકો અથવા બે જુદા જુદા પ્રોગ્રામો પર આપો છો જો તમે સોર્ટ કરવા માંગતા હોવ તેથી, તમે આ બંને છિદ્ર અલગથી સોર્ટ કરો અને હવે કી એક જ સોર્ટ કરેલ સૂચિમાં આ બંને સોર્ટ કરેલ ચીજો કાર્યક્ષમ રીતે સંયોજિત કરવા માટે સક્ષમ છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 01:27)

ચાલો છેલ્લા ભાગ પર ધ્યાન કેન્દ્રિત કરીએ, આપણે કેવી રીતે એક સોર્ટ કરેલી સૂચિમાં બે સોર્ટવાળી સૂચિને ભેગા કરીએ છીએ. હવે આ ફરી કંઈક છે જે તમે તદ્દન સ્વાભાવિક રીતે કરશો. ધારો કે તમારી પાસે બે શિક્ષણ મદદનીશોમાંથી બે આઉટપુટ છે, તો તમે શું કરશો તે તમે બન્નેમાં ટોચના કાગળના અભ્યાસક્રમનું પરીક્ષણ કરશો. હવે, ડાબા હાથની બાજુમાં આ ટોચનો કાગળ ડાબી બાજુ પર સૌથી વધુ ચિહ્ન છે. જમણા બાજુ પર ટોચનો કાગળ જમણી બાજુ પર સૌથી વધુ ચિહ્ન છે. આ બેમાંથી એક સૌથી વધુ એકંદર ટોચ છે. તેથી, તમે આને મહત્તમ કહી શકો છો અને તેને એકાંતે ખસેડી શકો છો હવે તમારી પાસે જમણી બાજુ પર બીજો સૌથી મોટો અને ડાબી બાજુ પર સૌથી વધુ પ્રથમ છે. ફરીથી, તમે મોટા એક જુઓ અને તે અહીં અને તેથી આગળ ખસેડો. તેથી, દરેક સમયે, તમે વર્તમાન યાદીઓ અથવા દરેક સૂચિની ટોચ પર જુઓ અને મોટા એકને આઉટપુટમાં ખસેડો, જમણે. અને જો તમે આ બધા ઘટકો સમાપ્ત ન થાય ત્યાં સુધી પુનરાવર્તન કરો છો, તો તમે તેને એકસાથે ક્રમમાં ગોઠવવાનું જાળવી રાખ્યું છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 02:24)

ચાલો આપણે જોઈએ કે આ એક સરળ ઉદાહરણમાં કેવી રીતે કામ કરશે. તેથી, અમારી પાસે 2, 32, 74, 89 અને 21, 55, 64 ની સૂચિબદ્ધ યાદીઓ છે. તેથી, આપણે ડાબેથી શરૂ કરીએ છીએ અને શરૂઆતમાં આ બે ઘટકોનું પરીક્ષણ કરીએ છીએ અને બેમાંથી નાના પસંદ કરીએ છીએ કારણ કે અમે યડતા ક્રમમાં સોર્ટ કરીએ છીએ. તેથી, આપણે 21 નાં બેમાંથી નાનાને પસંદ કરીએ છીએ અને હવે બીજી સૂચિ બે ઘટકોને ઘટાડી દીધી છે. આગળના તબક્કે, અમે પ્રથમ યાદીમાં પ્રથમ ઘટક અને બીજી લિસ્ટમાં બીજા તત્વની તપાસ કરીશું કારણ કે તે બાકી છે. આ બંને 32 માં નાની છે, તેથી આપણે 32 ખસેડીશું. હવે 55 ઓવરને અંતે બે ના નાનું છે, હવે 64 ઓવરને અંતે બે નાના છે. નોંધ કરો કે આપણે બીજી સ્થાને ખાલી છે તેવી સ્થિતિ પર પહોંચી ગયા છીએ, તેથી બીજી સૂચિ ખાલી છે, તેથી આપણે પ્રથમ લિસ્ટની નકલ કરી શકીએ કારણ કે તેની તુલના કોઈની સરખી નથી કારણ કે તે બધા બાકી તત્વો છે કે જે મજૂ કરવાની જરૂર છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 03:23)

આ કરવાથી, હવે અમારી પાસે એક-સોર્ટ કરેલી સૂચિમાં બે-સોર્ટ કરેલ સૂચિને મજૂ કરવાની એક પ્રક્રિયા છે; હવે મજૂ સોર્ટનો ઉપયોગ કરીને વસ્તુઓને મજૂ કરીને અમે તેનો અર્થ શું કરીએ તે સ્પષ્ટ કરી શકીએ છીએ. તેથી, મજૂ કરો સોર્ટ એ આ અલ્ગોરિધમનો છે જે સૂચિને બે ભાગોમાં વિભાજિત કરે છે અને પછી એક જ ભાગમાં છટણી છિદ્રને જોડે છે. તો આપણે શું કરીએ, આપણે સૌ પ્રથમ ડાબા હાથને સોર્ટ કરીએ છીએ. તેથી, આપણે 0 થી n ની 2 ઘાત 1 વડે લઈએ છીએ જ્યાં n એ લંબાઈ છે. આ બાકી છે અને આ અધિકાર છે હવે નોંધવું કે એક વસ્તુ પાંચથોન સંકેતલિપીમાં છે, આપણે અહીં અને અહીં જ સબસ્ક્રિપ્ટનો ઉપયોગ કરીએ છીએ, કારણ કે આ આપણને 2 ની 1 ઘાત દ્વારા પદ n પર લઈ જશે અને આ 2 દ્વારા n ની શરૂઆત થશે. તેથી, આપણે કાંઈ ચૂકીશું નહીં અને ન પણ કરીશું કંઈપણ ડુપ્લિકેટ, તેથી તે ખૂબ અનુકૂળ છે. આ એક બીજું કારણ છે કે પાંચથોન ને સંમેલન શા માટે છે કે સ્લાઇસની જમણા હાથની બાજુએ સ્લાઇસ -1 ની નીચે જાય છે. જો આપણે આના જેવું કંઈક લખીએ તો અમને ચિંતા કરવાની જરૂર નથી કે આપણે વત્તા 1, ઓછા 1 કરવું જોઈએ, અમે ફક્ત જમણી બાજુ અને ડાબી બાજુના ઇન્ડેક્સનું ડુપ્લિકેટ કરી શકે છે અને તે સંપૂર્ણ સૂચિની સંપૂર્ણ સૂચિ છે. તેથી, આપણે શું કરીએ તે આ એક કુદરતી પુનરાવર્તન અલ્ગોરિધમ છે; આપણે પ્રથમ અર્ધ અને બીજા અર્ધને સોર્ટ કરવા માટે આ અલ્ગોરિધમનો ઉપયોગ કરીએ છીએ અને પછી આપણે આ બે છટણીના વિભાગોને આઉટપુટમાં મજૂ કરીએ છીએ. મહત્વની બાબત એ છે કે આપણે વારંવાર એક જ વસ્તુ કરી રહ્યા છીએ; આપણે અડધાને અડધું કરીએ, બીજા અડધા સોર્ટ કરો અને મજૂ કરીએ અને ક્યારે અમે બેઝ કેસ સુધી પહોંચીએ છીએ જ્યારે આપણે એક સૂચિ પર પહોંચીએ છીએ જેમાં ફક્ત એક જ તત્વ અથવા શૂન્ય તત્વો હોય છે જે સોર્ટ કરવા જેવું નથી. જ્યારે આવી સ્થિતિ, અમે ફક્ત તે સૂચિને પાછા આપી શકીએ છીએ અને પછી આગળ વધવા મજૂ પર આધાર રાખી શકીએ છીએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 04:58)

ફરી એકવાર ચાલો આપણે ઉદાહરણરૂપે ઉદાહરણ જોઈએ. ધારો કે અમારી પાસે આઠ વસ્તુઓ છે જે આ રીતે ગોઠવાય છે. પ્રથમ પગલું એ તેને બે ભાગમાં વિભાજિત કરે છે અને દરેક અલગથી સોર્ટ કરે છે. તેથી, આપણે તેને બે જૂથોમાં વિભાજિત કરીએ છીએ; અમારી પાસે ડાબી અડધા અને જમણી અર્ધ છે હવે તે હજુ પણ વસ્તુઓ છે, જે અમને ખબર નથી કે કેવી રીતે સીધા સોર્ટ કરવી, તેથી ફરી આપણે બે ભાગમાં વહેંચીએ છીએ. ડાબા અડધા બે વધુ પેટાવિભાગોમાં વહેંચાયેલો છે અને તેથી જ અધિકાર છે. હવે આપણી પાસે લંબાઈ 2 ની સૂચિ છે, આપણે તેને હાથ દ્વારા ગોઠવી શકીએ છીએ, પરંતુ અમે કહીએ છીએ કે આપણે લંબાઈ 1 અથવા 0 ની સૂચિ સિવાય કંઈપણ સોર્ટ કરવું નથી, તેથી અમે તેને વધુ તોડી નાખીશું. હવે, આપણી પાસે ડાબેરી, 22 78 અને તેથી પર તુચ્છ યાદીઓ 43 અને 32 છે, તેથી અમે આપમેળે છટણી કરવામાં આવે છે, જે લંબાઈ એક આઠ યાદીઓ સાથે અંત. આ તબક્કે, પુનરાવર્તન પાછું આવે છે અને કહે છે, તમે સબ સૂચિ 43 ને અલગ કરી છે, ઉદાહરણ તરીકે, આ સૂચિ અમે ડાબી

બાજુએ 43 માં અને સયોટ રૂપે 32 માં સોર્ટ કરી છે, તેથી અમે તેમને મર્જ કરીને ભેગા કરવાની જરૂર છે. તેથી, અમે મર્જ પ્રક્રિયાને લાગુ કરીને 43 અને 32 મર્જ કરીએ છીએ અને આપણને 32 પહેલા 43 મળે છે જ્યારે આપણે 22 અને 78 માં મર્જ કરીએ છીએ તો તે એક જ ક્રમમાં રહે છે. અહિંયાં 57 પહેલા આવે છે 63 અને છેવટે, 13 ની પહેલા 91 ની છે. હવે, આ સ્તરે, આપણી પાસે બે યાદીઓ છે, જે બે અલગ અલગ હોય છે અને તેથી તેમને મર્જ કરવામાં આવે છે અને તે જ રીતે અહીં આપણે બે લિનીઓની બે યાદીઓ છે જે છટણી કરવામાં આવે છે અને તેઓ મર્જ હોવું જોઈએ તેથી, અમે પ્રથમ જોડીને મર્જ કરીએ છીએ, ત્યારબાદ 22, ત્યારબાદ 32, ત્યારબાદ 43, ત્યારબાદ 78, અને ત્યારબાદ આપણે 13, 57, અને 63 પછી 91 મેળવી શકીએ છીએ. તેથી, આ બંને મર્જેન્સ કર્યા પછી આપણે હવે લાંબી ચારની સૂચિ, જેમાંથી દરેકને અલગ પાડવામાં આવે છે. અને હવે આપણે આ 13 માંથી પસંદ કરવાનું સમાપ્ત કરીશું અને પછી આ 13 છે અને પછી આપણે 22 પસંદ કરીશું તો 32 પસંદ કરીશું અને પછી આપણે 43, 57, પછી 63, 78 અને પછી 91 બરાબર જ કરીશું. આ રિકર્ઝન કેવી રીતે જાય છે, તમે સૌ પ્રથમ તેને બેસ કેસ સુધી તોડી પાડો છો અને પછી તમે મર્જનો ઉપયોગ કરીને પાછળથી સંરેખિત કરો છો.

(સ્લાઇડ સમયનો સંદર્ભ લો: 07:08)

આ વ્યૂહરચના જે અમે મર્જ સોર્ટથી દર્શાવેલ છે તે એક સામાન્ય નમૂનારૂપ છે જેને વિભાજન અને જીતી કહેવામાં આવે છે. તેથી, જો તમને કોઈ સમસ્યા હોય, તો તમે સમસ્યાનું પેટા સમસ્યાઓમાં તોડી શકો છો, જેમાં એકબીજા સાથે કોઈ દખલગીરી નથી. તેથી, દાખલા તરીકે, અહીં સૂચિના પ્રથમ ભાગને સોર્ટ કરો અને સૂચિના બીજા ભાગને સોર્ટ કરો સ્વતંત્ર રીતે કરી શકાય છે તમે પ્રશિક્ષક દ્વારા નિયુક્ત કરેલ કાગળો લઈ શકો છો, તેમને બે જુદા જુદા શિક્ષણ સહાયકો આપો, તેમને બે અલગ રૂમમાં જવા માટે કહો; તેઓ તેમના છિદ્રને સોર્ટ કરવાનું સમાપ્ત કરવા માટે એકબીજા સાથે વાતચીત કરવાની જરૂર નથી. આવી પરિસ્થિતિઓમાં, તમે સમસ્યાનું વિતરણ સ્વતંત્ર પેટા સમસ્યાઓમાં કરો છો અને પછી તમારી પાસે ઉકેલાયેલી પેટા સમસ્યાઓને સંયોજિત કરવાની કાર્યક્ષમ રીત છે. તેથી તે ત્યાં કી છે, કેવી રીતે અસરકારક રીતે તમે સમસ્યાઓનો સંયોજિત કરી શકો છો. જો તમે સમસ્યાઓને ભેગા કરવા માટે ખૂબ જ લાંબો સમય લાવો છો તો તે તમને મદદ કરશે નહીં. પરંતુ, જો તમે તેને આ મર્જ સોર્ટ જેવા સરળ રીતમાં કરી શકો છો જ્યાં અમે ફક્ત બે યાદીઓને શરૂઆતથી અંત સુધી સ્કેન કરીને અને તેમને દરેક વસ્તુને અંતિમ વસ્તુ તરીકે સોંપણી કરીને મર્જ કરીએ છીએ, પછી તમે ખરેખર તે મેળવી શકો છો વિભાજન અને જીતી લાભ ઘણો

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 08:08)

ચાલો આપણે કેવી રીતે આ કરવા જઈ રહ્યા છીએ તેના વાસ્તવિક અલ્ગોરિધમનો પાસા પર વિગતવાર વધુ જુઓ. પ્રથમ, કારણ કે અમે મૂળભૂત પ્રક્રિયા તરીકે મર્જ કરવાની જોગવાઈ કરી હતી, અમે બે સોર્ટ કરેલી સૂચિને કેવી રીતે મર્જ કરીએ છીએ. આધાર કેસ તરીકે, જો આપણે ઉદાહરણમાં જોયું તેમાંથી કોઈપણ ખાલી છે, તો અમને કાંઈ કરવાની જરૂર નથી; અમે ફક્ત એક જ નકલ કરીએ છીએ. તેથી, અમે બે ઇનપુટ લીસ્ટ A અને B લઈ રહ્યા છીએ, જે બંને છટણી કરવામાં આવે છે અને અમે એક સોર્ટ કરેલ સૂચિ પરત કરવાનો પ્રયાસ કરી રહ્યા છીએ. તેથી, જો A ખાલી હોય, તો આપણે ફક્ત B માં C ની નકલ કરીએ છીએ; જો બી ખાલી છે, તો આપણે ફક્ત એક માં સી નકલ કરીશું. નહિતર, આપણે શું કરીએ જો બંને ખાલી ન હોય તો, આપણે A ના માથાના નાના એકને અને B નું માથું લેવું અને તેને સી પર ખસેડો, કારણ કે તે શું બાકી છે તે એકંદર સૌથી નાનું હશે. તેથી, આપણે A અને B ના પ્રથમ ઘટકની તુલના કરીએ છીએ અને આપણે નાના એકને સીમાં ખસેડીએ છીએ અને આપણે એ પુનરાવર્તન કરીએ છીએ જ્યાં સુધી A અને B ના બધા તત્વો ખસેડવામાં ન આવે ત્યાં સુધી.

(સ્લાઇડ સમય જણાવો: 09:05)

આ સામાન્ય રીતે આ મજ કર્યને અજમાયશી છે, તો બે સૂચિની સમાન લંબાઈની જરૂર નથી. તેથી, અમે લંબાઈ મીટરના લંબાઈ મીટર અને લંબાઈના A ની મજ કરી રહ્યા છીએ. શરૂઆતમાં, અમે મીટર અને એન સેટ કરીએ છીએ, કારણ કે આપણે નક્કી કરીએ છીએ કે આપણે કેટલા તત્વો શોધી કાઢ્યા છે કે ક્યારે સમાપ્ત તેથી, અમે m અને n એ એ અને બી ની લંબાઈને અનુક્રમે નિર્દેશિત કરીએ છીએ અને અમે ખાલી સૂચિ થવા માટે આઉટપુટ સૂચિને પ્રારંભ કરીએ છીએ, કારણ કે યાદ રાખો કે પાચથોન માં C નો પ્રકાર જ મૂલ્યને સોંપવામાં આવે તે પછી જ ઓળખવામાં આવશે. તેથી, આપણે તે કહેવું જોઈએ કે શરૂઆતમાં આઉટપુટ મજ લિસ્ટ એ ખાલી યાદી છે, જેથી અમે તેના પર વસ્તુઓ ઉમેરીને જોડણીનો ઉપયોગ કરી શકીએ. હવે આપણે જે કરવા જઈ રહ્યા છીએ તે એ અને બી બંનેની ડાબી બાજુથી આવશ્યકપણે શરૂ થાય છે, તેથી અમે અહીંથી જઈ રહ્યા છીએ અને જમણી તરફ જઈ રહ્યા છીએ; જેમ આપણે લાંબા સમય સુધી જઈએ છીએ ત્યાં આપણે બે ઘટકોમાંના એકને ખસેડવા જઈ રહ્યા છીએ. તેથી, આપણને અહીં નિર્દેશ કરવા માટે ઇન્ડેક્સની જરૂર છે, તેથી આપણે અનુક્રમે A અને B માં નિર્દેશ કરવા માટે ઇન્ડેક્સ i અને j નો ઉપયોગ કરીએ છીએ. અને શરૂઆતમાં, આ સૂચકાંકો પ્રારંભિક તત્વ પર નિર્દેશ કરે છે જે 0 છે. તેથી, જ્યારે આપણે 0 થી 1 માં ખસેડીએ છીએ ત્યારે તે સાથે ચાલો આપણે એમાં એક ઘટક પ્રક્રિયા કરી છે; જો આપણે તેને 3 માં ખસેડીશું; અમે એમાં ત્રણ ઘટકો પર પ્રક્રિયા કરી છે અને તેથી. તેથી, કોઈપણ સમયે હું વત્તા જે અમને જણાવશે કે અત્યાર સુધી કેટલા તત્વોને આઉટપુટમાં ખસેડવામાં આવ્યા છે; છેવટે, એમાં બધું જ અને બધું તમામમાં આઉટપુટમાં ખસેડવું આવશ્યક છે. આ તેટલા લાંબા સમય સુધી ચાલશે કારણ કે હું વત્તા જે કુલ મીટર વત્તા n સુધી પહોંચી નથી કારણ કે જે કુલ તત્વોની કુલ સંખ્યા હતી જેની સાથે આપણે શરૂ થવું પડ્યું હતું. જ્યારે હું વત્તા જે મીટર વત્તા એન કરતાં ઓછું હોય ત્યારે, આપણે જુદા જુદા કિસ્સાઓમાં જોવાની જરૂર છે. પ્રથમ બે કિસ્સાઓ છે જ્યાં બે યાદીઓમાંથી એક ખાલી છે; ક્યાં તો અમે એ અંત થાય છે, તેથી હું ખરેખર પહોંચી છે. તેથી, યાદ રાખો કે સૂચકાંકો 0 થી મા ઓછા 1 અને 1 થી n ની 1 ઘાત થાય છે. તેથી, જો હું વાસ્તવમાં મીટર પર ગયો છું; તેનો અર્થ એ કે, આપણે એ એલિમેન્ટ્સમાં થાકેલી છે. તેથી, આપણે બી માં આગામી તત્વ ઉમેરવું અને આપણે પોઇન્ટરને બીમાં અથવા ઇન્ડેક્સમાં વધારીને ચાલુ રાખીએ. એ જ રીતે, જો આપણે બી ના અંત સુધી પહોંચી ગયા છીએ, તો આપણે ઉમેર્યું આગામી તત્વ એ અને અમે પાછા જાઓ. હવે યાદ રાખો કે આ બિંદુએ, કારણ કે હું વત્તા જે મીટર વત્તા n કરતાં ઓછી છે, જો આપણે m તત્વો પૂર્ણ કરી છે, તો baટીએમ પ્લસ n સુધી પહોંચી શકાતું નથી, બીમાં કેટલાક તત્વ હોવું જોઈએ. એ જ રીતે, જો આપણે પૂર્ણ કર્યું છે તો મારો અર્થ છે કે આપણે B માં n એ તત્વો પૂર્ણ કર્યા છે, પરંતુ હું વત્તા j હજુ મીટર વત્તા n નથી, ઓછામાં ઓછા એક ઘટક હોવો જોઈએ તેમાં છોડી દીધું આ બે બાબતો ચોક્કસપણે એ હકીકતને ચકાસીને કામ કરશે કે અમે બધા ઘટકો સમાપ્ત કર્યા નથી, પરંતુ સૂચિમાંની એક ખાલી છે. હવે, ન તો સૂચિ થાકેલી હોય તો આપણે સરખામણી કરવી પડશે. તેથી, આપણે આ કેસમાં આવીએ છીએ અને અમે એ તપાસીએ છીએ કે A માં તત્વ બી માં તત્વ કરતાં નાના અથવા બરાબર છે. તેથી, આ બિંદુએ આપણે સામાન્ય રીતે કેટલાક એ i અને કેટલાક બી જોશે. તેથી, આપણે નક્કી કરવું પડશે કે આ બેમાંથી કઈ સી આગળ જાય છે. જો તે A માં આવે તો બેમાંથી નાનો, અમે તે ઇન્ડેક્સ આઇ પોઇન્ટર પર C ને ઉમેરીએ છીએ; નહિતર, અમે B મૂલ્યને j પોઇન્ટરને વધારીએ છીએ. આ લૂપના અંતમાં, આપણે શું કર્યું હોત તો m અને n તત્વોને યોગ્ય ક્રમાંકમાં A અને B માં સીમાં બદલ્યા હોત.

(સ્લાઇડ ટાઇ નો ઉલ્લેખ કરો: 12:29)

ચાલો આપણે ચકાસીએ કે આ કોડ છે અમે આશા રાખીએ છીએ કે પાચથોન તો, અહીં એક ફાઇલ મજ ડોટ પીઇ છે, જેમાં આપણી પાસે સ્લાઇડ પર જેવો જ કોડ હતો. તેથી, તમે ચકાસી શકો છો કે કોડ બરાબર એ જ છે જે આ લૂપમાંથી પસાર થાય છે જ્યારે હું વત્તા જે મીટર વત્તા એન કરતાં ઓછું હોય છે અને તે ચાર કેસોની તપાસ કરે છે અને તે મુજબ તે મુજબ A થી C અથવા B થી સી અને આખરે યાદી આપે છે C.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 13:00)

આ કરવા માટેનો સૌથી સરળ માર્ગ છે બે યાદીઓ અજમાવવા અને બાંધવા; ધારવું, અમે સંખ્યાઓની સૂચિ લઈએ છીએ જ્યાં પણ 0 થી 100 ની સંખ્યા. તેથી, આપણે 0 થી શરૂ કરીએ અને 2 ના પગલાઓમાં 100 પર જઈએ. અને અમે 1 થી 75 કહે તે વિચિત્ર સંખ્યાઓ લઈ શકીએ છીએ, તેથી આપણી પાસે નથી અહીં જ લંબાઈ જમણી. એકની લંબાઈ 50 છે, b ની લંબાઈ 37 છે. અને તે ચઢતા ક્રમમાં છે 2 થી 2 ના પગલાઓમાં; બી 1 થી 1 ના તબક્કામાં અને ફરીથી 2 ના પગલામાં છે. હવે જો આપણે કહીએ કે, a, b ને મર્જ કરો, અને પછી આપણને કંઈક મળે છે જે વાસ્તવમાં આ મર્જ લિસ્ટ રીટર્ન કરે છે. નોંધ લો કે 73 સુધી, જે b માં છેલ્લો ઘટક છે, આપણને બધી સંખ્યાઓ મળે છે. અને પછી અહીંથી, આપણે ફક્ત સંખ્યાઓ જ મેળવીએ છીએ કારણ કે આપણે ફક્ત એકની નકલ કરીએ છીએ. અને જો તમે મર્જ લેનલની લંબાઈ તપાસવા માંગો છો તો તે યોગ્ય રીતે છે 37 વત્તા 50 છે 87.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 14:12)

જો આપણે પાછા જઈએ અને આ કોડને ફરી જોવો, તો એવું લાગે છે કે આપણે ડુપ્લિકેટ કર્યું છે કેટલાક સ્થળોએ કોડ તેથી, આપણી પાસે બે પરિસ્થિતિઓ હોય છે કેસ 1 અને કેસ 4 જ્યાં આપણે તત્વને B માં C માં ઉમેરતા હોઈએ છીએ અને અમે j વધારી રહ્યા છીએ. અને એ જ રીતે, અમારી પાસે બે અલગ અલગ પરિસ્થિતિઓ છે- કેસ 2 અને કેસ 3, જ્યાં આપણે એમાંથી C માં એજિન ઉમેરી રહ્યા છીએ અને પછી i ને વધારીએ છીએ. તેથી, એવી દલીલ કરવાની પ્રેરણા છે કે આ અલ્ગોરિધમનો વધુ સઘન સંસ્કરણ હશે, જો આપણે આ કેસોને ભેગા કરીએ

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 14:57)

જો આપણે આ કેસોને ભેગા કરીએ તો આપણે કેસ 1 અને 4. એકને યાદ રાખી શકીએ. યાદ રાખો એક અને ચાર તે છે જ્યાં આપણે બીમાંથી મૂલ્ય લે છે. તેથી, આપણે 1 અને 4 ભેગા કરીએ છીએ અને કહીએ છીએ કે જો A ખાલી છે અથવા જો બી પાસે એક નાનું મૂલ્ય છે તો તમે બી માંથી મૂલ્ય લે અને તેને સી પર ઉમેરો અને j બરાબર j વત્તા 1 લખો. બીજી બાજુ, ક્યાં તો B ખાલી હોય અથવા A ની નાની કિંમત હોય તો તમે A થી મૂલ્ય અને તે જમણી બાજુના ઇન્ડેક્સને જોડો. ચાલો જોઈએ કે જો આપણે આ કોડ ચલાવવાનો પ્રયત્ન કરીએ તો શું થશે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 15:20)

અહીં આપણી પાસે ફાઇલ મર્જ ખોટો છે, નામ સૂચવે છે કે તે એક સમસ્યા છે, જ્યાં અમે કેસ 1 અને 4 સંયુક્ત કર્યો છે, જ્યાં આપણે તત્વને B માં C અને 2 માં જોડીએ છીએ અને 3 જ્યાં આપણે ભેગા કરીએ છીએ તત્વ તત્વને A માં C માં ઉમેરીએ. ચાલો આપણે આ રન કરીએ અને જુઓ. હવે, અમે પ્રારંભ બિંદુ પર મર્જ ખોટીએ છીએ.

(સ્લાઇડ સમય જણાવો: 15:46)

અને ચાલો આપણે સરળ કેસ કરીએ ધારો કે અમે 2, 4, 6 અને b એ 1, 3, 5 તરીકે લઈએ છીએ તો આપણે 1, 2, 3, 4, 5, 6 ની અપેક્ષા રાખવી જોઈએ. ચાલો એ અને b મર્જ કરવાનો પ્રયાસ કરીએ અને હવે અમને કોઈ ભૂલ મળી છે. જે કહે છે કે અમારી પાસે રેંજની સૂચિ ઇન્ડેક્સ છે ઇન્ડેક્સ શ્રેણીની બહાર સૂચવે છે કે અમે કોઈ ઘટકને એક્સેસ કરવાનો પ્રયાસ કરી રહ્યાં છીએ જે હાજર નથી અને તે આવું બને છે કે આ તે કિસ્સામાં છે જો હું બી અથવા ઇ બી બરાબર, બી j કરતા વધારે ચાલો જોઈએ કે આપણે શું ચાલી રહ્યું છે તેનું નિદાન કરી શકીએ છીએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 16:25)

શું ચાલી રહ્યું છે તે નિદાન કરવાનો એક સરળ રસ્તો એ છે કે કેટલાક યોગ્ય બિંદુઓ પર નામોનાં મૂલ્યોને છાપવા માટે અમુક નિવેદનો દાખલ કરો. હવે અહીં આપણી પાસે જ્યારે લૂપની અંદર કોઈ ભૂલ આવી છે ત્યારથી આપણે જે કર્યું છે તે અમે સ્ટેટમેન્ટ પ્રિન્ટમાં ઉમેર્યું છે, જેમ મેં કહ્યું હતું કે અમે ઔપચારિક રીતે જોઈ નથી અમે તે આગામી સપ્તાહમાં જોશું. પરંતુ તે સાહજિક વસ્તુ છે કે જે નામો એમ, એન, આઈ અને જો લે છે અને સ્ક્રીન પર તે અનુક્રમમાં તેમને છાપે છે, જેથી આપણે શું થઈ રહ્યું છે તે જોઈ શકીએ. ચાલો હવે આ ફરીથી રન કરીએ. તેથી, અમે દુભાષિયાને આ બદલાયેલ આવૃત્તિને ખોટી રીતે મર્જ કરો ચલાવો.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 17:03)

પહેલાં અને પહેલા તરીકે સેટ કરો. તેથી, એ 2, 4, 6 છે; b એ 1 3 5 છે. અને હવે અમે મર્જ રન કરીએ છીએ. અને હવે આપણે શું થઈ રહ્યું છે તે જુઓ. તેથી, m અને n એ પ્રારંભિક લંબાઈ 3 અને 3 છે. અને આ 0 અને 0 ની કિંમતો છે i અને j પોઇન્ટરો. તેથી, જો 1 બની જાય તો હું 1 અને તેથી આગળ વધું છું. આ બિંદુએ, આ તે છે જ્યાં સમસ્યા અધિકાર છે. આપણે જે શોધી કાઢ્યું છે તે છે કે જો હું બરાબર n અથવા ai બીજે કરતા વધારે હોય, તો હું મીટર બરાબર નથી. તેથી, હું હજી 3 નથી, તેથી તે એ બી ની તપાસ કરતાં મોટી છે કે નહીં તે તપાસવાનો પ્રયાસ કરી રહ્યો છે, પરંતુ આ બિંદુએ કમનસીબે જ એ એન છે. જો આપણી પાસે આ કેસો હોત તો અમે પહેલીવાર તપાસ કરી હોત જો હું 3 હોય તો જો જો 3 છે અને માત્ર જો તેમાંથી ત્રણ તો તેમની સરખામણી કરવા માટે પ્રયત્ન કરશે. કારણ કે હવે અમે માત્ર 3 જ છે તે તપાસ કરી રહ્યા છીએ, કારણ કે હું 3 નથી અને અમે બીજે સામે ai પર વેલ્યુ ચકાસી રહ્યા છીએ, પરંતુ દુભાગ્યે જ 3 પહેલાથી જ બની ગયું છે અને અમે તેને હજી સુધી ચકાસ્યું નથી. આ બે કેસોને સંયોજિત કરીને, અમે એવી પરિસ્થિતિને મંજૂરી આપી છે કે જ્યાં અમે ai અને bj ની સરખામણી કરવાનો પ્રયાસ કરી રહ્યા છીએ, તેમાંના એક માન્ય ઇન્ડેક્સ છે અને અન્ય માન્ય ઇન્ડેક્સ નથી. આ બે કેસોને જોડવાનું આકર્ષિત થતું હોવા છતાં, જ્યારે ખાસ કરીને જ્યારે આ સીમા પરિસ્થિતિઓ હોય ત્યારે જ્યારે આપણે અનુક્રમિત સૂચિ હોય ત્યારે, આપણે તે સુનિશ્ચિત કરવું જોઈએ કે જે ઇન્ડેક્સ અમે મેળવવાનો પ્રયાસ કરી રહ્યા છીએ તે એક માન્ય ઇન્ડેક્સ છે. અને કેટલીકવાર તે ગર્ભિત છે અને ક્યારેક આપણે સાવચેત રહેવું જોઈએ અને આ એવા કિસ્સાઓ પૈકી એક છે જ્યાં તમારે સાવચેત રહેવું જોઈએ અને આ વસ્તુઓને ઓપ્ટિમાઇઝ ન કરવું જોઈએ. નહિંતર, અમારું અલગ શરત હોવું જરૂરી છે કે જો હું મીટર કે જ બરાબર છું તો n ની સંખ્યા ઓછી છે અને જે ચાર સ્પષ્ટ કેસો સાથે આપણી આવૃત્તિ કરતાં વધુ જટિલ બની જાય છે. તેથી, અમે ચાર સ્પષ્ટ કેસ સાથે આવૃત્તિમાં પાછા જઈ શકીએ છીએ.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 18:52)

હવે આપણે જોયું છે કે કેવી રીતે યાદીને મર્જ કરવી, ચાલો આપણે તેમને સોર્ટ કરીએ. તેથી, આપણે શું કરવા ઇચ્છીએ છીએ એ એ તત્વો A ની સૂચિ લઈએ છીએ અને તેને આઉટપુટ સૂચિ B માં સોર્ટ કરીએ. તેથી, જો n ખરેખર 1 અથવા 0 છે, તેથી જો તે ખાલી છે અથવા તે લંબાઈ 1 મળી છે, તો અમારું કંઈ નથી; અન્યથા, આપણે પ્રથમ અર્ધને એક નવી સૂચિ એલમાં સોર્ટ કરીશું અને બીજા અડધાને ડાબી બાજુની એક નવી સૂચિ આર. એલ માં અને જમણી તરફ આર માટે સોર્ટ કરીશું. અને પછી આપણે પહેલાની મર્જ ફંક્શનને આઉટપુટ લિસ્ટ મેળવવા માટે અરજી કરીશું.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 19:31)

આ એક સરળ કાર્ય છે, સિવાય કે આપણે આપણી યાદીના જુદા ભાગો અથવા સ્લાઇસેસને સોર્ટ કરી રહ્યા છીએ. તેથી, આપણે વાસ્તવમાં ઈનપુટ સૂચિ અને સોર્ટ કરવાની સ્લાઇસના ડાબી અને જમણી અંતિમ

બિંદુઓમાં મર્જ સોર્ટ કરીશું. જો લંબાઈ 1 અથવા 0 ની સ્લાઇસ હોય તો આપણે સ્લાઇસને ફક્ત તે જ પાછું આપીએ છીએ. તે મહત્વનું છે કે આપણે સ્લાઇસના તે ભાગને પરત કરવો પડશે અને સમગ્ર ભાગ એ નહીં, કારણ કે તે ફક્ત સોર્ટિંગ છે. યાદ રાખો કે જ્યારે અમે કંઈક બે ભાગોમાં તોડ્યું, ઉદાહરણ તરીકે, જમણે, તેથી આ બિંદુએ અમને આ સ્લાઇસની સોર્ટ કરેલ સંસ્કરણ પરત કરવાની રહેશે, સમગ્ર સ્લાઇસ નહીં. તેથી, જો આપણે એ મૂળ કેસ છે તો A ને ડાબેથી જમવાની જરૂર છે; અન્યથા, આપણે મિડપોઇન્ટને શોધી કાઢીએ છીએ તો આપણે વર્તમાન સ્લાઇસથી ડાબા હાથની બાજુથી મિડપોઇન્ટ સુધીના સોર્ટને સોર્ટ કરીએ છીએ, અમે તેને એલ માં મુકીએ છીએ. પછી આપણે મધ્યમાં જમણી બાજુએ લઈએ છીએ, તેને આર માં મુકો અને આપણે અગાઉ ઉપયોગ કરીએ છીએ. ફક્શન આ બે ભાગો એલ અને આરમાંથી સોર્ટ કરેલી સૂચિ મેળવવા માટે મર્જ કરે છે અને આને પાછું આપો. આ એક અત્યંત સીધા આગળ અમલીકરણ છે; અહીં કોઈ યુક્તિ અથવા મુશ્કેલીઓ નથી. યાદ રાખવાની એકમાત્ર વસ્તુ એ છે કે આપણે આ બે વસ્તુઓ, ડાબા અને જમણી બિંદુ સાથે અમારી મર્જ સોર્ટ કાર્યને વધારવું પડશે. જો તમે દ્વિસંગી શોધ માટે યાદ હોવ તો અમારી પાસે એક સમાન બાબત હતી, જ્યાં અમે શોધ માટે અંતરાલ રાખીએ છીએ. તેથી, આપણે તે કહેવું રાખવું પડશે કે કયા અંતરાલ અમે શોધી રહ્યા છીએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 20:53)

ચાલો મર્જ સોર્ટની પાયથોન ના અમલીકરણને જોઈએ. તેથી, અહીં આપણી પાસે ફાઈલ મર્જ સોર્ટ ડોટ પીઈ છે. અમે ફક્શન મર્જથી શરૂઆત કરીએ છીએ, જે આપણે પહેલા અને એથી બી એ સીમાં ખસેડવા માટે ચાર માર્ગ કેસ સ્પ્લિટ સાથે જોયાં છીએ

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 21:06)

અને પછી આપણે ફાઇલના તળિયે ઉમેરો નવી ફક્શન મર્જ સોર્ટ જે અમે પહેલાની સ્લાઇડ પર જોયું છે જે A ના સ્લાઇસને ડાબેથી જમણી તરફ લઈ જાય છે અને તેને ગોઠવે છે. જો તે તુચ્છ સ્લાઇસેસ છે, તો તે સ્લાઇસને પાછું આપે છે કારણ કે તે છે. નહિતર, તે તેને બે ભાગોમાં તોડે છે અને તે છુપાવે છે. ચાલો જોઈએ કે આ ખરેખર કેવી રીતે ચાલશે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 21:25)

આપણે પાયથોન લઈએ છીએ અને મર્જ સોર્ટમાંથી કહીએ છીએ, તમામ કાર્યો આચાર કરીએ છીએ. અને હવે ચાલો એક મોટી શ્રેણી લઈએ. ધારો કે, જો આપણે બધા પણ સંખ્યાઓ દ્વારા અનુસરતી તમામ વિચિત્ર સંખ્યાઓ લઈએ છીએ. તેથી, આપણે 2 થી 2 ના રેન્જમાં 1 થી 100 શ્રેણી કહીએ છીએ. તે વિચિત્ર સંખ્યાઓ છે અને તે પછી એક જ શ્રેણીના બધા પણ નંબરો કહે છે. તેથી, એ હવે સંખ્યાઓ દ્વારા અનુસરતી વિચિત્ર નંબરો છે. તમે કલ્પના કરો કે જો હું 0 થી A ની લંબાઈને સોર્ટ કરું છું, તો પછી તમે ક્રમાંકમાં ક્રમાંકિત નંબરો મેળવશો.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 22:12)

હવે જો હું મોટી સૂચિ 1000 લઈશ તો હું ફરીથી વિચારું છું, ફરીથી બધું સોર્ટ કરે છે.

(સ્લાઇડ સમય જણાવો: 22:16)

હવે અમારો દાવો છે કે આ એક ઓર્ડર એન લોગ એન એલ્ગોરિધમ છે. તે પણ મોટી યાદી માટે સારી રીતે કામ કરીશું જો હું કહું કે 10000 જે યાદ રાખશે તો દાખલ સોર્ટ અથવા પસંદગી સોર્ટ સાથે ખૂબ લાંબો સમય લેશે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 22:26)

પ્રશ્ન એ છે કે તે અહીં કેટલો સમય લે છે અને તે ખૂબ ઝડપી આવે છે.

(સ્લાઇડ સમય જણાવો: 22:29)

આપણે આગળ વધીએ છીએ અને ઉદાહરણ તરીકે 100000 કહીએ છીએ, અને તે અહીં પણ વ્યાજબી ઝડપી આવે છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 22:37) તેથી, આપણે જોઈ શકીએ છીએ કે આપણે ખરેખર લિંક્સની શ્રેણીમાં મોટા પ્રમાણમાં વિસ્તૃત કરી છે કે જે આપણે $n \log n$ એલ્ગોરિધમ તરફ જઈને સોર્ટ કરી શકીએ છીએ કારણ કે હવે મર્જ કરો સોર્ટ વસ્તુઓને સંભાળી શકે છે જે 100% જેટલી મોટી 100,000 છે. થોડા 1000 પછી દાખલ સોર્ટ અથવા પસંદગી સોર્ટ કરો. ધ્યાનમાં રાખવાનું બીજો નાનો મુદ્દો એ નોંધવું છે કે અમે તેની રિકર્ઝન મર્યાદાની સમસ્યામાં ચલાવી શકીએ છીએ કે જેણે ઇન્સેશન સોર્ટ સાથે અમારી પાસે છે જે અમે ફરીથી આવર્તક રીતે વ્યાખ્યાયિત કર્યું છે. સૂચિમાં દરેક ઘટક માટે, અમે ફરી યાદ આવવું કોલ કરી રહ્યાં છીએ. જો આપણી પાસે 1000 તત્વો છે, તો આપણે 1000 ફરી યાદ આવવું કોલ કર્યાં છે અને પછી આપણે મર્યાદા વધારવી પડશે. હવે અહીં પણ 100,000 માટે અમારી પાસે સમસ્યા નથી અને કારણ એ છે કે અહીં ફરી આવર્તક કોલ્સ એ એક ઘટક દીઠ એક નથી, પરંતુ એક અડધી યાદી દીઠ. તેથી, અમે ફક્ત લોગ અને ફરી યાદ આવવું કોલ્સ બનાવી રહ્યા છીએ તેથી, 100,000 તત્વોને માત્ર 100,000 લોગ કરવાની જરૂર છે. યાદ રાખો લોગ 1000 એ આશરે 10 છે. તેથી, અમે 20 થી ઓછા ફરી યાદ આવવું કોલ કરી રહ્યા છીએ, તેથી અમને રિકર્ઝનની મર્યાદામાં કોઈ સમસ્યા નથી, અમારી પાસે આ કાર્યની ઊંડાઈમાં કોઈપણ બાકી પુનરાવર્તન નથી.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 23:45)

અમે મર્જ સોર્ટ ક્રિયામાં જોયું છે અને અમે કોઈપણ દલીલ વગર એવો દાવો કર્યો છે કે વાસ્તવમાં ઓર્ડર એન લોગ એન છે અને દર્શાવ્યું છે કે તે કદ 100,000 ની ઇનપુટ માટે કાર્ય કરે છે. આગામી વ્યાખ્યાનમાં, અમે ખરેખર ગણતરી કરીશું કે શા માટે મર્જ કરો સોર્ટ ઓર્ડર એન લોગ એન છે.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રોફેસર. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ
ઇન્ડિયન ઇન્સ્ટીટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 04 લેક્ચર - 02

મર્જ કરો સોર્ટ, એનાલિસિસ છેલ્લા પ્રવચનમાં અમે મર્જ સોર્ટ પર જોયું અને અમે અનૌપચારિક રીતે દાવો કર્યો કે તે નિવેશ સોર્ટ અથવા પસંદગી સોર્ટ કરતા વધુ કાર્યક્ષમ અને અમે દાવો કર્યો છે કે તે સમય ક્રમમાં $n \log n$ માં ચલાવે છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 00:03)

યાદ રાખો કે તેના આધારને એક મર્જ એલ્ગોરિધમ તરીકે મર્જ કરો, જે બે સોર્ટ કરેલ યાદીઓ, એ અને બી લે છે અને બધા ઘટકો પર એક સ્કેન કરીને તેમને એક સાથે જોડે છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 00:29)

મર્જ સોર્ટ વિશ્લેષણ કરવા માટે, પ્રથમ વસ્તુ જેને અમે કરવાની જરૂર છે તે મર્જ કાર્યનું વિશ્લેષણ આપવાનું છે. બે યાદીઓ, એ અને બીનાં ઇનપુટ માપોના સંદર્ભમાં કેટલા સમયનો મર્જ કરવામાં આવે છે, તેથી ધારો કે એ પાસે એમ ઘટકો છે અને B નો n એલિમેન્ટ છે અને આપણે આ તમામ ઘટકોને સીમાં એક સોર્ટ કરેલ સૂચિમાં મૂકવા માગીએ છીએ. યાદ રાખો કે અમારી પાસે પુનરાવર્તન થયું છે, જ્યાં પુનરાવર્તનના દરેક પગલામાં અમે જોયેલી, એ અને બીમાં પ્રથમ ઘટક અને બેથી નાનાને સી પર ખસેડો. તેથી સ્પષ્ટ રીતે સી દરેક પુનરાવર્તન સાથે એક ઘટક દ્વારા વધે છે અને ત્યારથી અમે A અને B થી C ના બધા મીટર વત્તા n તત્વો, C નું માપ મીટર વત્તા n છે. દરેક પુનરાવર્તનમાં આપણે શું કરીએ છીએ? સારું આપણે સરખામણી કરીએ છીએ અને પછી, આપણે કોઈ અસાઇનમેન્ટ કરીએ છીએ અને પછી કેટલાક સૂચકાંક વધારીએ છીએ. તેથી, આ એક નિશ્ચિત કામગીરી છે. તેથી, તે સતત છે તેથી, કામની કુલ રકમ m વત્તા n નો પ્રમાણસર છે. નોંધ લો કે m વત્તા n એ મહત્તમ મહત્તમ બાયનરી વધુ વત્તા n છે. તેથી, મીટર 7 છે અને n એ 15 છે, પછી 5 વત્તા 7 વત્તા 15 બરાબર બે ગુણ્યા 15 ની હશે. આપણે કહી શકીએ કે વિધેય તરીકે મર્જ કરો મહત્તમ મીટર અને n ના ક્રમમાં અને ખાસ કરીને ઘણી વાર જેમ સોર્ટ મર્જ કરો, અમે બે યાદીઓ લગભગ સમાન કદ લઈ રહ્યા છીએ જેમ કે આપણે એક યાદીને બે છિદ્રમાં વિભાજીત કરીએ છીએ અને ત્યારબાદ આપણે તેમને મર્જ કરીએ છીએ. જો એમ અને એન બન્ને સમાન છે, તો લગભગ સમાન કદ, પછી મહત્તમ અને મીટર એ તેમાંના એક છે. અનિવાર્યપણે મર્જ, ઇનપુટ સૂચિનાં કદમાં રેખીય છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 02:06)

હવે, મર્જનું વિશ્લેષણ કર્યા પછી, ચાલો સોર્ટ કરો. તેથી, મર્જ કરો સોર્ટ કહે છે કે જો સૂચિ નાની છે, તો શૂન્ય અથવા એક ઘટક હોય છે, કંઈ કરવાનું નથી. અન્યથા તમારે સમસ્યાને બે અર્ધ સૂચિ માટે હટાવવી પડશે અને પછી તેમને મર્જ કરવો પડશે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 02:23)

કોઈપણ પુનરાવર્તિત કાર્યની જેમ આપણે પુનરાવૃત્તિના સંદર્ભમાં આવા કાર્ય દ્વારા લેવામાં આવેલા સમયને વર્ણવવું પડશે. તેથી, ચાલો આપણે ધારીએ, કારણ કે આપણે 2 દ્વારા ભાગાકાર કરવાનું ચાલુ રાખીએ છીએ જેથી આપણે 2 વચ્ચે ભાગાકાર રાખી શકીએ છીએ, વચ્ચે કોઈ વિચિત્ર નંબર મેળવ્યા વગર. ચાલો ધારો કે ઇનપુટ n એ 2 ની કેટલીક સંપૂર્ણ શક્તિ છે. તે n એ 2 છે જે k નો છે. જ્યારે આપણે વિભાજીત રીતે બે યાદીઓમાં મર્જ કરીએ છીએ, ત્યારે આપણી પાસે બે યાદીઓની સંખ્યા n દ્વારા 2 હોય છે. N તત્વો માટે લેવામાં સમય 2 ગુણ્યા 2 ગુણ્યા 2 માટે લેવાય છે અને આ મર્જ ઘટક છે. આપણી પાસે ઓર્ડર એન સ્ટેપ છે, અમને બે યાદીઓની સંખ્યા 2 દ્વારા મર્જ કરવાની જરૂર છે અને યાદ રાખો કે અમે ફક્ત મર્જને ઇનપુટના કદમાં લીનિયર છે. તેથી, આપણી પાસે બે ઉપનામોની બે પેટા સમસ્યા છે, જે 2 દ્વારા 2 ગુણ્યા n છે અને અમે મર્જ કરીએ છીએ જેના માટે ઓર્ડરની જરૂર છે. દ્વિસંગી શોધ અને પુનરાવર્તિત નિવેશ સૉર્ટ સાથે, અમે આ પુનરાવૃત્તિને ઉકેલવાથી તેને ઉકેલવા કરી શકીએ છીએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 03:33)

અમે બેઝ કેસથી શરૂ કરીએ છીએ. જો અમારી પાસે કદ 1 ની સૂચિ હોય, તો અમારી પાસે કંઈ નથી. તેથી, ટી નો n એ 1 છે અને T નો n એ સામાન્ય રીતે 2 ગુણ્યા n ની 2 વત્તા n છે. જો આપણે આ વિસ્તરણ કરીએ તો, આપણે 2 ને ટીએન ને અસ્પષ્ટતા વાંચીશું અને આપણે બે વખત ટી n ને 4 વત્તા n 2 મળશે, કારણ કે જો આપણે આને એક નવું ઇનપુટ તરીકે લઈએ, તો તે જ વ્યાખ્યાનો ઉપયોગ કરીને વિસ્તરે છે અને જો આપણે તેને ફરીથી લખીશું. તેથી, આપણે 2 ગુણ્યા 2 ની 2 ઘાત લખીશું અને આપણે આ 4 ને બે સ્ક્વેર્ડ લખીશું. અમે શોધીશું કે આ તે આ ફોર્મ 2 માં 2, 2 સ્ક્વેર્ડ ટી n ને 2 ચોરસમાં લખવાનું છે અને હવે નોંધ લો કે તમારી પાસે અહીં બે વાર 2 ગુણ્યા n છે. આ 2 અને આ 2 રદ કરશે. તેથી, આપણી પાસે એક પરિબલ n અને અન્ય પરિબલ n છે. અગત્યની બાબત એ છે કે તમારી પાસે ઘાત માં 2 છે અને તમારી પાસે 2 ની પહેલા n છે. હવે, આગળના પગલામાં આપણે શું કરીશું તે મુજબ આ બે વખત ટીએન 4 નો વિસ્તૃત કરીએ. તેથી, આપણે બે વખત ટીએન 4 વડે વિસ્તૃત કરીએ અને તે આપણને બીજા n ને 8 આપશે જે તમે 2 2 ક્યુબ તરીકે લખી શકો છો. 2 સ્ક્વેર્ડ 2 એન 2 સ્ક્વેર વત્તા 2 n થાય તે 2 ઘનની 2 ઘન 2 ની ઘન વત્તા 3 n થાય. તેથી, નોંધ લો કે બન્ને થ્રીસ એક્સરખી બની ગયા છે. તેથી, આ રીતે જો આપણે k પગલાંઓ અથવા j પગલાંઓ પછી ચાલુ રાખીએ તો આપણી પાસે 2 વખત j ગુણ્યા n ને 2 થી j વત્તા j વખત n થશે. હવે, અમે કેટલો સમય આ કરી રહ્યા છીએ? જ્યાં સુધી અમે બેઝ કેસ નહીં કરીએ ત્યાં સુધી અમે આ કરવાનું ચાલુ રાખીએ છીએ. તેથી જ્યારે j નો લોગ થાય છે, જ્યાં લોગ દ્વારા લોગ થાય છે, તો આપણે સામાન્ય રીતે બેઝ 2 માં લોગ થાય છે, પછી n એ 2 થી j એ 1 થશે. તેથી, ટી દ્વારા ટી2 થી j પણ હશે 1. લોગ એન પગલાંઓ પછી, આ સમીકરણ લોગ n વત્તા લોગ n વખત n ને દરેક જગ્યાએ સરળ બનાવે છે, જ્યારે દરેક પાસે એજે છે, આપણે લોગ એન મૂકી અને તે 1 બની ગયું છે, તેથી તે અદ્રશ્ય થઈ ગયું છે. આપણી પાસે 2 છે જે j નો 2 છે લોગ એ વત્તા આ j એ લોગ n છે અને પછી, આપણી પાસે n છે અને આ વ્યાખ્યા પ્રમાણે લોગ n માં 2 છે તે ફક્ત n છે. તેથી, 2 નો લોગ એ n છે અને આપણી પાસે n લોગ એન છે અને અમારા નિયમ પ્રમાણે આપણે જ્યારે ઉચ્ચ પદ ઉપર રાખીએ છીએ, ત્યારે આપણે n લોગ એન એ n કરતાં મોટી છે. મર્જ સૉર્ટ માટે અમે ઓ n લોગ એન નું અંતિમ મૂલ્ય મેળવીએ છીએ.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 06:00)

મર્જ ખૂબ જ ઉપયોગી ઓપરેશન થાય છે. જો આપણે ડુપ્લિકેટ્સ ધરાવીએ છીએ તો આપણે શું જોયું તે બે સૂચિ વિશ્વાસુપણે એક સૉર્ટ કરેલી સૂચિમાં અમારી યાદી વિહંગાવલિ કરશે. તેથી, જો આપણે મર્જ કરીએ 1, 3 અને 2, 3, તો આપણે ફોર્મ 1, 2, 3, 3 ની યાદી સાથે અંત કરીશું. આ રીતે મર્જ કાર્ય કરશે. તે કોઈ પણ માહિતી ગુમાવતું નથી. તે ડુપ્લિકેટ્સ રાખે છે અને અંતિમ યાદીમાં વિશ્વાસુ રૂપે કોપી કરે છે. બીજી બાજુ, અમે એવી પરિસ્થિતિ ધરાવીએ છીએ કે જ્યાં આપણે યુનિયન ઇચ્છતા હોય. અમે બહુવિધ કોપીઓ રાખવા નથી માંગતા અને અમે માત્ર એક જ નકલ રાખવા માંગીએ છીએ. યુનિયન કેસમાં અહીં

આપણે શું કરીશું. ચાલો ધારો કે અમારી પાસે બે યાદીઓ છે અને સામાન્ય રીતે, આપણે સૂચિમાં પહેલાથી ડુપ્લિકેટ્સ મેળવી શકીએ છીએ. ચાલો ધારો કે અમારી પાસે 1, 2, 2, 6 અને 2, 3, 5 છે, તો આપણે સામાન્ય મર્જ કરીએ છીએ. તેથી, આપણે અહીં અને હવે એકને ખસેડીએ છીએ, જ્યારે આપણે બે ઘટકોને હરાવીએ છીએ, જે બરાબર છે, તો પછી આપણે મૂળભૂત રીતે સ્કેન કરવાની જરૂર છે જ્યાં સુધી આપણે આ સમાન વસ્તુ સમાપ્ત ન કરીએ અને તેની એક કોપી નકલ કરીએ અને પછી છેલ્લે, આપણે 3 અને પછી, 5 અને પછી 6. જ્યારે A અને એ i બરાબર બી j બંને બાજુઓને વધારશે અને ખાતરી કરો કે આપણે તે બ્લોકના અંતમાં જઈશું. બીજો વિકલ્પ છે આંતરછેદ કરવા. ધારો કે આપણે 1, 2, 6 અને 2, 6, 8 લેવા માંગીએ છીએ અને જવાબ 2, 6 સાથે સામાન્ય ઘટકો તરીકે બહાર નીકળવા માંગીએ છીએ, તો જો એક બાજુ બીજી બાજુ કરતા નાની હોય, તો આપણે તે તત્વ છોડી દઈએ છીએ કારણ કે તે નથી ત્યાં બંને યાદીઓમાં તેથી, જો A i એ B કરતાં ઓછું છે તો હું ઇન્ડેક્સ કડું છું, જો B j એ I કરતાં ઓછું છે, તો આપણે j વધારી છે અને જો તે સમાન છે તો આપણે યુનિયન લેશે, આપણે સામાન્ય ઘટકની એક નકલ રાખીશું. તેથી, મર્જનો ઉપયોગ વિવિધ સંયોજનોને અમલમાં મૂકવા માટે કરી શકાય છે. તેનો ઉપયોગ બે યાદીઓનું જોડાણ લેવા અને ડુપ્લિકેટ્સ કાઢી નાખવા માટે થઈ શકે છે. તેનો ઉપયોગ બે યાદીઓના આંતરછેદને લેવા માટે અને છેલ્લે, પરીક્ષણ માટે એક કસરત તરીકે કરી શકાય છે કે તમે તેને સમજો છો અને જુઓ કે તમે સૂચિ તફાવત કરવા માટે મર્જનો ઉપયોગ કરી શકો છો. સૂચિ તફાવત એ નીચેની કામગીરી છે. જો હું 1, 2, 3, 6 અને મારી પાસે 2, 4, 6, 8 હોય તો આ તફાવત એ પ્રથમ યાદીમાંના બધા ઘટકો છે જે બીજી સૂચિમાં નથી. બે અહીં છે અને તે અહીં છે, તમે 2 દૂર કરો, 6 અહીં છે અને અહીં 2 દૂર કરો. તેથી, તમારે 1, 3 મેળવવું જોઈએ. તેથી, જો તે A છે, અને આ બી છે, તો આ કહેવાતી યાદી છે તફાવત એ બાદ બી. તેથી, જો તમે મર્જનું સંસ્કરણ લખી શકો છો કે જે તમને એમાંના તમામ ઘટકો આપે છે જે બીમાં નથી પણ યાદી તફાવત તરીકે પણ ઓળખાય છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 08:49)

હવે, મર્જ કરો સોર્ટ સ્પષ્ટપણે સોર્ટ અને પસંદગી સોર્ટ કરતાં શ્રેષ્ઠ છે કારણ કે તે ઓર્ડર એન લોગ એન છે, તે સૂચિને હેન્ડલ કરી શકે છે કારણ કે જેમ આપણે જોયું છે કે થોડા હજાર જેટલા કદની સંખ્યા 100,000 છે, પરંતુ તે તેનો અર્થ એ નથી કે મર્જ કરો સોર્ટમાં મર્યાદાઓ નથી. મર્જ સોર્ટની મર્યાદાઓમાંની એક એ છે કે અમને નવી એરે બનાવવા માટે ફરજ પાડવામાં આવે છે, જ્યારે અમે બે યાદીઓને મર્જ કરીએ છીએ. નવી સૂચિ બનાવી વગર બે યાદીઓને કાર્યક્ષમ રીતે મર્જ કરવાની કોઈ સ્પષ્ટ રીત નથી. તેથી, વધારાની સ્ટોરેજની દ્રષ્ટિએ દંડ છે. અમે આ જગ્યાને ડબલ કરવી પડશે કે જ્યારે અમે યાદીઓ સાથે શરૂ કરીએ અને અમે મર્જ સોર્ટમાં સોર્ટ કરવા માંગીએ છીએ. મર્જ સોર્ટ સાથેની અન્ય સમસ્યા એ છે કે તે સ્વાભાવિક રીતે ફરી યાદ આવવું છે અને તેથી, મર્જ કરો સોર્ટને પોતે પ્રથમ અર્ધ અને બીજા ભાગમાં હવે, આ વિભાવનાત્મક રીતે ખૂબ સરસ છે. અમે જોયું છે કે રિકર્સિવ વ્યાખ્યાઓ recursive કાર્યો ખૂબ સ્વાભાવિક રીતે પ્રત્યક્ષ પ્રમાણ પર આધારિત વ્યાખ્યાઓ સાથે સંકળાયેલા છે અને તે એક જ પ્રકારની નાની સમસ્યાઓના સંદર્ભમાં સમસ્યાના માળખાને સમજવા માટે અમને સહાય કરે છે. હવે, કમનસીબે, પ્રોગ્રામિંગ ભાષામાં ફરી યાદ આવવું કોલમાં વર્તમાન વિધેયને સસ્પેન્ડ કરવું, નવી ગણતરી કરવી અને પછી, વર્તમાન કાર્ય માટે અમે સસ્પેન્ડ કરેલી મૂલ્યોને પુનર્સ્થાપિત કરવો. તેથી, જો અમારી પાસે હાલમાં આઈ, જે, કે જેવી સ્થાનિક નામો માટે મૂલ્યો છે, તો આપણે તેમને ક્યાંક સ્ટોર કરવો પડશે અને પછી તેમને પાછો પાડો અને ફરી યાદ આવવું હોય ત્યારે જૂના મૂલ્યો ચાલુ રાખો. આના માટે વધારાની કાર્ય ચોક્કસ રકમની જરૂર છે રિકર્સિવ કોલ્સ અને રીટર્ન તેમની પોતાની સમય મર્યાદા પર ખર્ચાળ બનવાનું ચાલુ કરે છે. તેથી, તે સરસ હોઈ શકે જો આપણે મર્જ સોર્ટના ઓર્ડર n લોગ એન વર્તન બંને હોઈ શકે. અને અમે આ ફરી યાદ આવતી વસ્તુ સાથે દૂર કરી શકીએ છીએ, પરંતુ આ માત્ર એક નાની ટિપ્પણી છે. પરંતુ સોર્ટિંગની મર્જ કરો સોર્ટ મૂળભૂત ઓર્ડર એન લોગ એન સોર્ટિંગ એલ્ગોરિધમ છે અને તે જાણવું ખૂબ જ ઉપયોગી છે કારણ કે તે આડકતરી રીતે અથવા સીધી અન્ય ઘણી વસ્તુઓમાં ભૂમિકા ભજવે છે.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રોફેસર. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કોમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટીટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 04 લેક્ચર - 03

ક્લિકસોર્ટ આપણે જોયું કે મર્જ સોર્ટ એ ઓર્ડર એન લોગ એન સોર્ટિંગ એલ્ગોરિધમ છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 00:01)

પરંતુ તેની કેટલીક ખામીઓ છે જે તેને કેટલીક વાર અવ્યવહારુ બનાવે છે. મુખ્ય સમસ્યા એ છે કે તેમને મર્જ કરવા માટે વધારાની જગ્યા જરૂરી છે. અમે એ પણ જોયું કે રિકર્ઝનનો ઉપયોગ કર્યા વિના મર્જ સોર્ટ અમલીકરણ કરવું મુશ્કેલ છે અને પ્રોગ્રામિંગ ભાષામાં રિકર્ઝન તેની પોતાની કિંમત ધરાવે છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 00:23)

ચાલો આપણે જગ્યા સમસ્યાને સંબોધિત કરીએ. મર્જ ફંક્શનનો અમલ કરવા માટે મર્જ સોર્ટ દ્વારા જરૂરી વધારાની જગ્યા જરૂરી છે અને શા માટે અમને મર્જ કરવાની જરૂર છે? અમે મર્જ કરવાની જરૂર છે તે છે કે જ્યારે આપણે મર્જ સોર્ટ કરીએ છીએ, ત્યારે અમારી પાસે પ્રારંભિક સૂચિ છે અને પછી અમે તેને બે ભાગમાં વિભાજિત કરી છે, પરંતુ સામાન્ય રીતે જમણી બાજુની વસ્તુઓ કરતાં મોટી છે જે ડાબી બાજુમાં આઇટમ્સ હોઈ શકે છે. દાખલા તરીકે, જો આપણે ડાબી બાજુની સંખ્યા અને જમણી બાજુએ સંખ્યાઓ કહીએ તો આપણે ક્યાં તો બંને બાજુથી નંબરો લઈને મર્જ કરવો પડશે. તેથી, જો આપણે વ્યવસ્થા કરી શકીએ તો આપણી વિભાજિત સમસ્યાની ડાબી બાજુએ જે બધી વસ્તુ વિભાજિત સમસ્યાના જમણા બાજુથી નીચી હોય તેના કરતાં ઓછી હોય, તો પછી અમને બધામાં મર્જ કરવાની જરૂર નહીં હોય અને આ કદાચ આપણને આ સમસ્યાની જરૂર પડતી હોય મર્જ માટે વધારાની જગ્યા

(સ્લાઇડ સમયનો સંદર્ભ લો: 01:19)

અમે કેવી રીતે મર્જ કર્યા વગર વિભાજિત અને જીતીશું. ધારે છે કે અમે મધ્ય મૂલ્ય જાણતા હતા; મધ્યસ્થ મૂલ્યને યાદ રાખવું એ મૂલ્ય છે જે અડધા તત્વો નાની છે અને અડધા મોટા છે. અમે મધ્યમથી ડાબી બાજુથી તમામ મૂલ્યોને નાનામાં ખસેડી શકીએ છીએ અને મધ્યમથી જમણી અડધા ભાગ જેટલા મોટા ભાગના જેમ આપણે જોશું કે આ યાદીની લંબાઈના પ્રમાણમાં નવો એરે બનાવ્યાં વિના કરી શકાય છે. ડાબા અડધા બધા નાના મૂલ્યો ખસેડવાની અને આ જમણી અડધાથી મોટી કિંમતો કર્યા પછી આ પુનઃ ગોઠવણી કર્યા પછી આપણે આ વિભાજનને લાગુ પાડીએ છીએ અને વ્યૂહરચના પર વિજય મેળવી શકીએ છીએ અને જમણી અને ડાબી અડધા અલગથી સોર્ટ કરી શકીએ છીએ અને કારણ કે અમે બાંધકામ આપી છે કે ડાબી બાજુમાં બધું અડધો ભાગ જમણી અડધાથી નાની છે, આ આપમેળે અર્થ થાય છે કે આ વિભાજન અને પગલાને જીતીને પછી આપણે કોઈ પણ તુચ્છ નજીવી રીતે જવાબોને જોડવાની જરૂર નથી કારણ કે ડાબા અડધા પહેલાથી જ અડધા જ નીચે છે તેથી, આપણને મર્જ કરવાની જરૂર નથી. જો આપણે આ રણનીતિ લાગુ કરીએ તો આપણે ફરી યાદ આવવું સમીકરણ જેમ કે મર્જ સોર્ટ કરીશું. તે કહે છે કે લંબાઈની સૂચિને સોર્ટ કરવા માટેનો સમય જરૂરી છે, અમને પ્રથમ 2 ની સૂચિની 2 યાદીઓની યાદી કરવાની જરૂર છે અને અમે મર્જ કરવા માટે ક્રમમાં નથી, પરંતુ સૂચિને સડવું જેથી તે બધા નાના મૂલ્યોમાં હોય. ડાબી અને જમણે તેથી, ફરી યાદ આવવું તે પગલું છે તે પહેલાં આપણે પુનઃ ગોઠવણી કરવી એ ઓર્ડર n છે, જ્યારે મર્જ પુનરાવર્તિત પગલા પછીનું પગલું હતું, જે અગાઉના કેસમાં

ઓર્ડર હતું, પરંતુ જો આપણે પુનરાવૃત્તિને હટાવીએ, તો તે એક જ છે, આપણને બીજી ઓર્ડર મળે છે n લોગ એન એલોરિધમ આ અભિગમ સાથે મોટા અંતરાય મધ્યમાં શોધવાનો છે. યાદ રાખો કે આપણે અગાઉ જણાવ્યું હતું કે યાદીને સોર્ટ કરવાનાં ફાયદા પૈકીની એક છે કે આપણે સોર્ટ પછી મધ્યમ તત્વ તરીકે મધ્યમ ઓળખી શકીએ. હવે અહીં, અમે સોર્ટ કરતા પહેલા મધ્યસ્થી માટે કહીએ છીએ, પરંતુ અમારું લક્ષ્ય સોર્ટ કરવાનું છે, તે વિરોધાભાસી છે. જો સોર્ટિંગના આઉટપુટને સોર્ટિંગ માટે ઇનપુટ તરીકે રાખવાની જરૂર હોય તો આનો મતલબ એ છે કે આપણે યાદીને વિભાજિત કરવા માટે તત્વના વધુ સરળ પસંદગી સાથે વ્યૂહરચનાને અજમાવી જુઓ. મધ્યસ્થની શોધ કરવાને બદલે અમે ફક્ત એમાંના અમુક મૂલ્યને પસંદ કરીએ છીએ અને તેનો ઉપયોગ પીવોટ તત્વ તરીકે ઓળખાય છે. અમે આ પીવટને ધ્યાનમાં રાખીને A ને વિભાજિત કરી છે જેથી તમામ નાના તત્વો ડાબા હોય અને બધા મોટા તત્વો જમણી તરફ છે

(સ્લાઇડ સમય જણાવો: 03:58)

આ અલ્ગોરિધમનો ક્વિકોર્ટ કહેવામાં આવે છે, તે 1960 ના દાયકામાં કાર હોરે નામના વ્યક્તિ દ્વારા શોધાયું હતું અને તે સૌથી પ્રસિદ્ધ સોર્ટિંગ એલ્ગોરિધમમાંનું એક છે. તેથી, આપણે પીવટ તત્વ પસંદ કરીએ છીએ જે સામાન્ય રીતે એરેની સૂચિમાં પ્રથમ ઘટક છે. આ પાઇવટ એલિમેન્ટની બાબતમાં, અમે ભાગ, નીચલા ભાગમાં અને ઉપલા ભાગમાં છીએ. તેથી, આપણે બધા નાનાં તત્વોને ડાબી બાજુએ ખસેડીએ છીએ અને પીવટ એલિમેન્ટની પસંદગીના સંદર્ભમાં જમણી બાજુએ બધા મોટા તત્વોને ખસેડીએ છીએ અને અમે ખાતરી કરી શકીએ કે ધરી બે વચ્ચે આવે છે કારણ કે અમે એરેમાં પ્રથમ એલિમેન્ટ પસંદ કર્યું છે પીવટ તેથી, આ પછી આપણે તેને નીચલા અને ઉપલા ભાગ વચ્ચે મધ્યમાં ખસેડવા માગીએ છીએ અને પછી, અમે ફરીથી બન્ને ભાગોને સોર્ટ કરીએ છીએ. અહીં એક ઉચ્ચ સ્તરનું દૃશ્ય છે કે કેવી રીતે ઝડપી સૂચિ સામાન્ય સૂચિ પર કામ કરશે. ધારો કે આ અમારી સૂચિ છે, આપણે સૌ પ્રથમ યાદીની શરૂઆતને ઓળખીએ છીએ, પ્રથમ ઘટક તરીકે પીવટ એલિમેન્ટ. હવે, બાકીના ઘટકો માટે આપણે જાણવું પડશે કે જે નાના છે અને જે મોટા છે. તેથી, અમે આ કેવી રીતે કરી શકીએ તે વગર, અમે 32, 22 અને 13 ની ઓળખી કાઢીએ છીએ, જે ત્રણ ઘટકો છે જે નાના હોય છે અને પીળા રંગના હોય છે અને અન્ય ચાર ઘટકો જે લીલા રંગમાં ચિહ્નિત થયેલ છે તે મોટા છે. આ માપદંડના સંબંધમાં વાસ્તવમાં પ્રથમ પગલું છે. તેથી, આપણે આ તત્વોને આસપાસ ખસેડવાનું છે જેથી તેઓ બે બ્લોક્સમાં આવે. જેથી, 13, 32 અને 22 ડાબી બાજુ આવે; 63, 57, 91 અને 78 જમણી બાજુ આવે છે અને પીવટ તત્વ 43 મધ્યમાં આવે છે. આ ફરીથી ગોઠવણીનું પગલું છે અને હવે આપણે ફરી પીળા બિટ્સ અને લીલા બિટ્સને સોર્ટ કરીએ છીએ, એમ ધારી રહ્યા છીએ કે આપણે તે કરી શકીએ છીએ, અમારી પાસે એક સોર્ટ એરે અને નોટિસ છે કારણ કે તમામ પીળા વસ્તુઓ 43 કરતા ઓછી છે અને બધી લીલા વસ્તુઓ મોટી છે 43, કોઈ વધુ મર્ગીંગ આવશ્યક નથી.

(સ્લાઇડ સમયનો સંદર્ભ લો: 05:51)

તો ચાલો જોઈએ કે કેવી રીતે પાર્ટીશન કામ કરે છે. અહીં, આપણી પાસે અગાઉની સૂચિ છે અને આપણે 43 તરીકે આપણા પીવટ તત્વને ચિહ્નિત કર્યું છે અને અમે બાકીના ઘટકોનો સ્કેન કરવા માગીએ છીએ અને તેમને બે જૂથોમાં વહેંચીએ છીએ; 43 કરતાં ઓછી, પીળા રાશિઓ; તે 43 કરતાં મોટા, લીલા રાશિઓ અને તેમને ફરીથી ગોઠવવા. આપણે શું કરીશું, આપણે બે પોઇન્ટરો રાખીશું; એક પીળા નિર્દેશક અને લીલો પોઇન્ટર અને સામાન્ય નિયમ હશે કે કોઈ પણ સમયે આપણે અમુક અંતર પર હશે, પીળી પોઇન્ટર જે હું નારંગીમાં તેને વધુ દૃશ્યક્ષમ બનાવવા માટે અને લીલા પોઇન્ટર બનાવવા માટે દોરીશ. આ ક્રમમાં જશે; નારંગી પોઇન્ટર અથવા પીળા પોઇન્ટર હંમેશાં લીલા પોઇન્ટર અને ઇન્ડિવિવટી પ્રોપર્ટીની પાછળ હશે જે અમે જાળવીશું કે આ ઘટકો 43 કરતા ઓછા અથવા સમાન છે, આ ઘટકો 43 કરતાં મોટી છે અને આ ઘટકો અજ્ઞાત છે. આપણે શું કરવાનો પ્રયાસ કરીએ છીએ, આપણે ડાબેથી જમણે ખસેડવાનો પ્રયત્ન કરીએ છીએ અને તમામ અજાણ્યા તત્વોનું વર્ગીકરણ કરીએ છીએ; દરેક વખતે

જ્યારે આપણે એક અજ્ઞાત તત્વ જોઈશું, તો આપણે બે પોઇન્ટર પાળીશું જેથી અમે આ યોગ્ય જાળવીએ. કે 43 અને પ્રથમ પોઇન્ટર વચ્ચે આપણી પાસે 43 થી નાના અથવા સમાન તત્વો છે; પ્રથમ પોઇન્ટર અને બીજા પોઇન્ટર વચ્ચે, આપણી પાસે તત્વ 43 કરતાં વધારે છે અને લીલા પોઇન્ટરની જમણી બાજુ છે, આપણી પાસે હજુ સુધી સ્કેન કરાયેલું છે. શરૂઆતમાં કંઈ નહીં તો આપણે 32 ને જોઈએ, કારણ કે 32 એ 43 કરતા નાનું છે, આપણે પીળા પોઇન્ટર ખસેડીએ છીએ અને આપણે લીલા પોઇન્ટરને પણ દબાણ કરીએ છીએ. તેથી, અજાણ્યા વસ્તુઓ 22 થી શરૂ થાય છે, અને ત્યાં પીળા અને લીલા નિર્દેશક વચ્ચે કંઈ નથી જે સૂચવે છે કે આપણે હજુ 43 થી વધારે મૂલ્ય મેળવ્યું નથી, એ જ 22 માટે થાય છે. હવે, જ્યારે આપણે 78 જુઓ ત્યારે, આપણે નોંધ્યું છે કે 78 મોટા છે 43 કરતાં. હવે, આપણે ફક્ત લીલા નિર્દેશક ખસેડીએ છીએ અને પીળો નિર્દેશક નથી, અમારી પાસે આ ત્રણ અંતરાલો પહેલાની છે. યાદ રાખો કે આ તે ભાગ છે જે 43 કરતા ઓછું છે; આ તે ભાગ છે જે 43 કરતા વધારે છે અને આ ભાગ અજ્ઞાત છે. અમે આ રીતે આગળ વધીએ છીએ. હવે, આપણે 63 પર નજર કરીએ છીએ, 63 વાર ગ્રીન ઝોન વિસ્તરે છે, 57 ગ્રીન ઝોન વિસ્તરે છે, 91 એ ગ્રીન ઝોન વિસ્તરે છે. હવે, આપણે કંઈક શોધી કાઢીએ છીએ જ્યારે આપણે 13 શોધો. તેથી, 13 એ એક તત્વ છે જે પીળા ઝોનમાં મૂકવામાં આવે છે, એક રણનીતિ ઘણાં બધાં સ્થળાંતર કરવાનું છે. અમે 13 થી 22 કે પછી 22 પછી ખસેડીએ છીએ અને અમે બધું જ 78 થી આગળ જમણી બાજુએ ખસેડીએ છીએ, પરંતુ વાસ્તવમાં એક સ્પષ્ટ વ્યૂહરચના એ છે કે 13 અહીં જ જવું જોઈએ. તેથી, અમારે જગ્યા બનાવવાની જરૂર છે, પરંતુ જગ્યા બનાવવાને બદલે આપણે કહી શકીએ, તે અમને વાંધો નથી, અમે છેવટે લીલા વસ્તુઓને સોંટ કરવા જઈ રહ્યા છીએ. તે કેવી રીતે વાંધો છે, આપણે આ 78 ને લઈશું અને તેને 13 માં ખસેડીશું, તેથી, કોઈ પણ સ્થળાંતર કરવાને બદલે, આપણે ફક્ત ગ્રીન ઝોનના પ્રથમ તત્વનું તત્વ બદલીએ છીએ જે આપણે અત્યાર સુધી જોઈ રહ્યા છીએ, તે આપોઆપ બંને પીળા ઝોન અને લીલા ઝોન યોગ્ય રીતે વિસ્તરશે. તેથી, અમારું આગળનું પગલું એ 43 ની સરખામણીમાં 13 જેટલું નાનું છે અને તેને 78 સાથે સ્વેપ કરવું છે. હવે, અમે મધ્યવર્તી તબક્કે પહોંચી ગયા છે જ્યાં આપણે ધ્વજની જમણી બાજુએ બધું જ સ્કેન કર્યું છે અને અમે તેમને નાનામાંના વર્ગમાં વર્ગીકૃત કર્યા છે. અને મોટા લોકો જે છે હવે, તે પીળા વસ્તુઓને 43 ની ડાબી બાજુએ ખસેડવાનું રહે છે. ફરી એક વાર આપણે એ જ સમસ્યા જોઈ છે જ્યારે અમે પીળા ઝોનમાં 13 નો સમાવેશ કર્યો છે. જો આપણે 43 ને યોગ્ય સ્થાને ખસેડીએ તો આપણે અહીં બધું જ ડાબી બાજુએ ખસેડવું પડશે, પરંતુ તેના બદલે આપણે આ 13 ભાગને છેલ્લા ભાગમાં પીળા ઝોનમાં લઈ શકીએ છીએ અને તેને ત્યાં બદલી અને 32 અને 22 ના સ્થાનાંતરિત કરી શકીએ છીએ. આ ઓર્ડરને ખલેલ પહોંચાડે છે, પરંતુ કોઈપણ રીતે આ ક્રમમાં ગોઠવાયેલ નથી છે, તે માત્ર ક્રમમાં ગોઠવાયેલ નથી રહે છે. તેથી, આપણે આ કરીએ છીએ અને હવે આપણી પાસે એરે ફરીથી ગોઠવવાની જરૂર છે, કારણ કે આપણે ડાબી બાજુએ આ તમામ બાબતો ધરી કરતા નાના હોય છે અને ધુમ્મસ મધ્યમાં હોય છે અને જમણી બાજુ બધું જ ધરીથી મોટો છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 09:47)

અહીં પાયથોન માં અમલીકરણ છે. તેથી, યાદ રાખો કે ક્લોસરટ મર્જ સોર્ટ જેવી છે અને દ્વિસંગી શોધની જેમ છે, અમે તેને નાના અને નાના ભાગોમાં વારંવાર લાગુ કરીએ છીએ. સામાન્ય રીતે, આપણે તે યાદીને પાસ કરવી પડશે જે અમે A ને કહીએ છીએ, અને સેગમેન્ટમાં ડાબી અને જમણી તરફના અંતનો નિર્દેશ કરે છે. જો આપણી પાસે કંઈક છે જે આપણે r ઓછા 1 થી એક સ્લાઇસ કરી રહ્યા છીએ, જો આ સ્લાઇસ લંબાઈમાં 1 અથવા 0 હોય, તો અમે કાંઈ નથી બીજું આપણે આ પાર્ટીશનની વ્યૂહરચનાને અનુસરીએ છીએ, જે છે કે આપણે 1 થી r ની સોર્ટ કરી રહ્યા છીએ. 1. સ્થિતિ 1, આ ધ્વનિ છે. અમે શરૂઆતમાં પીળા પોઇન્ટરને અહીં મુકીશું, એમ કહીને કે પીળા ઝોનનું અંતર વાસ્તવમાં માત્ર ધ્વનિ છે, ત્યાં કંઈ નથી. તેથી, પીળા લી વત્તા 1 છે અને હવે આપણે લીલો આગળ વધીએ છીએ અને દર વખતે જ્યારે તમે ગ્રીનમાં એક તત્વ જુઓ છો જે એક નવું હરિયાળ છે જે એક જ ધુમ્મસ છે તેના કરતાં નાનું છે. યાદ રાખો કે આ ધ્વનિ છે, જો આપણે ક્યારેય લીલા જોશું તો તપાસ માટે આગામી મૂલ્ય એ એ [એલ] ની સરખામણીએ નાની છે અથવા તેના બરાબર છે જેથી અમે આ મૂલ્યને પીળા ઝોનના અંતમાં લાવીએ. આ આપણે 13 થી કર્યું છે અને પછી આપણે પીળા પોઇન્ટરને પણ ખસેડીએ છીએ, અન્યથા જો

આપણે મૂલ્યને જોયું જે કડક રીતે મોટી છે, તો આપણે ફક્ત લીલા પોઇન્ટર ખસેડીએ છીએ જે લૂપ માટે પૂર્ણપણે કરવામાં આવે છે અને અમે પીળાને ખસેડી શકતા નથી. આનાં અંતમાં, આપણી પાસે ધ્વનિ હોય છે, પછી અમારી પાસે પિવટની બરાબર છે અને ત્યારબાદ આપણી પાસે વધારે છે. તેથી, આ તે મધ્યવર્તી મંચ કે જે અમે આ લૂપના અંતમાં પ્રાપ્ત કર્યું છે. હવે, આપણે ધ્વનિ શોધી કાઢવું પડશે અને તેને યોગ્ય સ્થાને ખસેડો. યાદ રાખો કે પીળી, પીળો એ પોઇન્ટ કરતાં છેલ્લો એલિમેન્ટ હોય છે. તેથી, પીળો આની આગળ, પહેલાં એક મૂલ્ય છે. તેથી, આપણે પીળા બાદ 1 મૂલ્યને લે છે અને તેને ડાબી મૂલ્ય સાથે ફેરબદલ કરીએ છીએ અને હવે આપણે શું કરવાની જરૂર છે, આપણે હવે p કરતાં વધુ, p, p કરતાં વધારે છે અને આ તે છે જ્યાં પીળો છે તેથી, અમારે 0 થી પીળા ઓછા 1 ની જરૂર છે, આપણે ફરીથી સોર્ટ કરવા નથી માગતા. કારણ કે પૃષ્ઠ પહેલેથી જ યોગ્ય સ્થાને મૂકવામાં આવી છે, તેથી અમે એલથી પીળા બાદ 1 અને પીળાથી જમણી બાજુએ ઝડપી ક્રમાંક

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 12:11)

અહીં, આપણે પાચથોન કોડ લખ્યું છે જે આપણે એક ફાઇલમાં સ્લાઇડમાં જોયું છે. તમે તપાસ કરી શકો છો કે તે બરાબર એ જ કોડ છે જે સ્લાઇડમાં હતું. અમે તેનો પ્રયાસ કરી અને ચલાવી શકીએ છીએ અને ખાતરી કરી શકીએ છીએ કે તે કામ કરે છે. તેથી, આપણે પાચથોન ને કોલ કરીએ છીએ અને આ કાર્ય આયાત કરીએ છીએ. યાદ રાખો કે આ ફરી એક કાર્ય છે જે સ્થળે ગોઠવે છે. જો તમે કંઈક સોર્ટ કરવા માંગો છો અને અસર જુઓ તો આપણે તેને નામ આપવું પડશે અને ત્યારબાદ તે નામને સોર્ટ કરવું પડશે અને પછીથી નામ તપાસવું પડશે. દાખલા તરીકે, ચાલો 500 થી નીચે 0 સુધીના મૂલ્યોની શ્રેણી લઈએ, જો આપણે ક્લયુસોર્ટ (એલ) કહીએ તો આપણે અલબત્ત, તે અંત આપે છે, પછી હું યોગ્ય રીતે સોર્ટ કરીશ. તેથી, તમે તે બધા જોઈ શકતા નથી, પરંતુ તમે 83, 84 થી 102 સુધી 500 સુધી જોઈ શકો છો. હવે, અમારી પાસે તે જ સમસ્યા છે જેમાં અમારી પાસે નિવેશ સોર્ટ સાથે હતી. જો આપણે 1000 ને કહીએ અને પછી આપણે આને ઝડપી કરવાનો પ્રયાસ કરીએ, તો આપણે આ રિકર્ઝનની ઊંડાઈ મેળવીશું કારણ કે આપણે જોશું કે, ખરાબ કિસ્સામાં, ક્લોસોર્ટ નિવેશ સોર્ટ જેવા થોડી વર્તે છે અને આ એક ખરાબ કેસ છે. તેથી, આની આસપાસ જવા માટે અમારે સામાન્ય વસ્તુ કરવી પડશે - આપણે સીઆઈએસ મોડ્યુલને આયાત કરવું પડશે અને પુનરાવર્તનની મર્યાદાને એકદમ મોટી કંઈક બનાવવાનું કહેવું પડશે, 10000 કહેવું છે, કદાચ 100,000 અને પછી જો આપણે તેને ક્લોસ્ટોર્ટ કરવા કહીએ તો કોઈ સમસ્યા નથી. આ બીજું એક કેસ છે જ્યાં પાચથોન ને રિકર્ઝનની મર્યાદા મેન્યુઅલી સેટ કરવાની હોય છે અને એક વસ્તુ આપણે જોઈ શકીએ છીએ કે ક્લોસ્ટોર્ટ એટલું સારું નથી કારણ કે આપણે માનીએ છીએ કારણ કે જો આપણે જોઈએ તો દાખલા તરીકે કંઈક 7500 કહે છે. દૃશ્યમાન સમય આપણે જોયું કે મજ સોર્ટ જે એન લોગ એન હતું તે 5000 અને 10000 અને તત્કાલ 100,000 પણ કરી શકે છે. તેથી, સ્પષ્ટ રીતે ક્લિનસોર્ટ વગાડતા નથી અને સોર્ટ મજ કરે છે અને અમે હકીકતમાં જોશું કે, ક્લોસ્ટોર્ટમાં ઓર્ડર એન લોગ એન વર્તણૂક નથી હોતી કારણ કે અમે ગમ્યું હોત અને તે એટલા માટે છે કે આપણે સરેરાશનો ઉપયોગ કરી રહ્યાં નથી, પરંતુ પ્રથમ વેલ્યુ વાત કરવા માટે. આપણે જોશું કે આગલી ભાષણમાં કેમ ક્લોસર્ટ ખરેખર ખરાબ કેસ ઓર્ડર એન લોગ અલ્ગોરિધમનો નથી.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રોફેસર. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 04 લેક્ચર - 04 ક્વીક્સર્ટ એનાલિસિસ

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 00:01)

અગાઉના પ્રવચનમાં, નીચે પ્રમાણે કામ કરે છે તમે સૌ પ્રથમ તત્વોને એક પીવટ એલિમેન્ટ સાથે ફેરબદલ કરી શકો છો, જે સારમાં, એરેમાં પ્રથમ મૂલ્ય કહે છે. તમે પાર્ટિશન એ, નીચલા અને નીચલા ભાગોમાં, જે પીવટ કરતા નાના હોય છે, અને તે પીવટ કરતા મોટા હોય છે. તમે એરે ફરીથી ગોઠવો જેથી ધરી નાના અને મોટાભાગના ભાગો વચ્ચે રહે છે અને પછી તમે બન્ને ભાગોને સોર્ટ કરો છો.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 00:31)

અને આ ખરેખર પાયથોન કોડ છે જે અમે દોડ્યો અને અમે જોયું કે તે ખરેખર 7500 ની સાઈઝની સૂચિ માટે પણ મજા નહીં કરે, અમે જોયું કે તે પ્રશંસનીય રીતે લાંબો સમય લે છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 00:42)

ક્લોસર્ટના સૌથી ખરાબ કેસ વર્તન શું છે? ક્વિકસર્ટનું સૌથી ખરાબ કેસ વર્તન ત્યારે આવે છે જ્યારે તે મુખ્ય આદર્શ મૂલ્યથી ઘણું દૂર છે. અમે ઇચ્છીએ તે આદર્શ મૂલ્ય મધ્યસ્થ છે, મધ્ય એ સૂચિને બે ભાગોમાં વિભાજિત કરશે અને ત્યાંથી કામને બે ભાગોમાં વહેંચી દેશે, પરંતુ જો પીવટ એકંદરે એરેમાં ન્યૂનતમ અથવા મહત્તમ મૂલ્ય જેટલું થાય, તો પછી તે મહત્તમ છે ત્યારબાદ દરેક અન્ય કિંમત નીચલા ભાગમાં જશે અને કંઈ ઉચ્ચ ભાગમાં જશે નહીં. તેથી, પુનરાવર્તિત કોલમાં તે n બાદ 1 ઘટકોની સોર્ટિંગ ધરાવે છે. જો એક પાર્ટિશન ખાલી હોય, તો બીજો એક કદ n ઓછા 1 હોય, તો તે થાય કે જો ધરી ભારે તત્વો પૈકીનું એક છે અને જો આવું થાય અને આ સૌથી ખરાબ કેસ છે, તો n તત્વોને સોર્ટ કરવા માટે, આપણે ફરી આવવું જોઈએ સોર્ટ n ઓછા 1 ઘટકો અને એરેને ફરીથી ગોઠવવા માટે અમે હજી પણ આ કમમાં n વર્ક કરી રહ્યા છીએ કારણ કે અમે આને શોધી કાઢતા નથી ત્યાં સુધી અમે આખા એરેથી પસાર થયા છીએ અને પાર્ટિશન પૂર્ણ કર્યું છે. આ કહે છે TN tn ઓછા 1 વત્તા n છે અને જો આપણે આ વિસ્તૃત કરીએ તો આ બરાબર એ જ પુનરાવર્તન થાય છે કે જે આપણે સોર્ટિંગ સોર્ટ અને પસંદગી સોર્ટ માટે જોયું છે. તેથી, TN 1 વત્તા 2 થી n સુધી હશે અને આ સત્ર માત્ર n એ n વત્તા 1 થી 2 હશે જે ઓર્ડર n ચોરસ હશે. ક્લોસર્ટ વિશે વધુ વિરોધાભાસી વસ્તુ એ છે કે આ તો થશે જ, જો એરે યોગ્ય કમમાં અથવા ખોટા કમમાં સોર્ટ થાય છે. ધારો કે તમે ચડતા કમમાં સોર્ટ કરવાનો પ્રયાસ કરી રહ્યાં છો અને પહેલાથી જ ચડતા કમમાં એરે છે તો પ્રથમ ઘટક એ સૌથી નાનું તત્વ હશે. વિભાજિત આપણી પાસે એક બાજુએ n બાદ 1 આપણે અને બીજી બાજુ 0 મૂકીશું અને તે ચાલુ રહેશે. ક્લોસર્ટના સૌથી ખરાબ કેસ વાસ્તવમાં પહેલાથી સોર્ટ થયેલ એરે છે યાદ રાખો, આપણે જોયું છે કે દાખલ કરવા માટે સોર્ટ અને પહેલાથી સોર્ટ એરે સારી રીતે કામ કરે છે કારણ કે શામેલ પગલું ખૂબ ઝડપી અટકે છે. તેથી, ક્લોસર્ટ વાસ્તવમાં સૌથી વધુ ખરાબ સ્થિતિમાં છે જે દાખલ સોર્ટ કરતાં વધુ ખરાબ બનાવે છે અને ખાસ કરીને પહેલાથી સોર્ટ કરેલા એરેઝ પર.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 02:40)

જો કે, તે તારણ આપે છે કે આ એક અત્યંત મર્યાદિત પ્રકારનો ખરાબ કેસ છે અને કોઈ પણ વાસ્તવમાં દરેક શક્ય ક્રમયય પર ક્લોસ્ટોર્ટના વર્તનનો પ્રયાસ કરી શકે છે. તેથી, જો આપણે અંતિમ વિશિષ્ટ કિંમતો સાથે ઇનપુટ એરે લઈએ, તો આપણે ધારણ કરી શકીએ છીએ કે આ n મૂલ્યોના કોઈપણ ક્રમયય સમાન રીતે સંભવ છે અને આપણે ગણતરી કરી શકીએ કે આ દરેક અલગ અલગ ક્રમયયોમાં કેટલો સમયનો ઝડપી ઉપયોગ કરે છે અને એમ ધારી રહ્યા છીએ કે તમામ સમાન રીતે સંભવ છે, જો અમે એ ક્રમયયો પર સરેરાશ રાખીએ છીએ જે આપણે સરેરાશ મૂલ્યની ગણતરી કરી શકીએ છીએ. હવે, આ સાઉન્ડ સરળ છે પરંતુ ગાણિતિક રીતે તે અત્યાધુનિક છે અને સોર્ટિંગ દુર્લભ ઉદાહરણોમાંનું એક છે જ્યાં તમે તે ઇનપુટસની તમામ સંભવિત ઇનપુટ્સ અને સંભાવનાઓને અર્થપૂર્ણ રીતે ગણતરી કરી શકો છો, પરંતુ જો તમે આ ગણતરી કરો છો તો તે ચોક્કસ ગાણિતિક અર્થમાં ઝડપથી ચોક્કસપણે સરેરાશમાં ક્રમમાં n લોગ એન માં કામ કરે છે. તેથી, ખરાબ કેસ હોવા છતાં તે ઓર્ડર n ચોરસ ખરેખર ખૂબ જ દુર્લભ છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 03:41)

ધુમ્મસગ્ર તત્વની નિશ્ચિત પસંદગીને કારણે સૌથી ખરાબ કેસ ઊભો થાય છે. તેથી, અમે પ્રથમ એલિમેન્ટને અમારા એલ્ગોરિધમમાં પીવટ તરીકે પસંદ કર્યું છે અને તેથી ખરાબ કિસ્સામાં ઇનપુટ રચવા માટે, જો આપણે હંમેશા પ્રથમ સ્થાને સૌથી નાનું કે સૌથી મોટું એલિમેન્ટ મુકીશું તો અમને ખરાબ કેસ ઇનપુટ મળશે. આ આપણને કહે છે કે ચડતા અથવા ઉતરતા ક્રમમાં ગોઠવેલ ઇનપુટ અમારી પસંદના ઓબ્જેક્ટ માટે સૌથી ખરાબ કેસ છે. ધારો કે, તેના બદલે અમે મિડપોઇન્ટ પસંદ કરવા માગીએ છીએ, તો અમે એરેમાં મધ્યમ ઘટકને એક રેન્ડમ તરીકે લઈએ છીએ અને પછી ફરીથી અમે સૌથી ખરાબ મૂલ્ય મૂકીને પછાત કામ કરીને ખરાબ કેસ ઇનપુટ બનાવી શકીએ છીએ જેથી દરેક તબક્કે મધ્યમ મૂલ્ય સૌથી નાનું હોય. અથવા સૌથી મોટું મૂલ્ય તેથી, ધ્વનિની કોઈપણ નિશ્ચિત પસંદગી માટે, જો અમારી પાસે એક ધરી હોય, જે દરેક પુનરાવર્તનમાં એક નિશ્ચિત રીતે લેવામાં આવે છે, તો અમે હંમેશા સૌથી ખરાબ કેસનું પુનઃનિર્માણ કરી શકીએ છીએ. તેમ છતાં, જો આપણે અમારી વ્યૂહરચના બદલીએ અને કહીએ કે દરેક વખતે અમે કોલોર્ટોઈ કહીએ છીએ તો અમે રેન્ડમ એલિમેન્ટ્સની શ્રેણીની અંદર એક મૂલ્ય પસંદ કરીએ છીએ અને પીવટ તરીકે પસંદ કરીએ છીએ, પછી તે તારણ કાઢે છે કે આપણે આ ક્રમમાં n વર્ગે સૌથી ખરાબ કેસને હરાવી અને અપેક્ષિત મેળવી શકીએ છીએ ચાલી રહેલ સમય, એવરેજ રનિંગ સમય ઓન n લોગ n ની સંભવિતરૂપે.

(સ્લાઈડ સમયનો સંદર્ભ લો: 04:56)

આના પરિણામે, કારણ કે તે સૌથી ખરાબ કેસ ઓર્ડર n વર્ગે છે, પરંતુ સરેરાશ ઓર્ડર એન લોગ એન, ક્લોસ્ટ વાસ્તવમાં ખૂબ ઝડપી છે. આપણે જે જોયું તે મર્જ સોર્ટ સાથેના એક મુદ્દાને સંબોધિત કરે છે કારણ કે ફરીથી ગોઠવણીને સોર્ટ કરીને અમે વધારાની જગ્યા બનાવી નથી. આપણે જોયું નથી કે તમે ક્યાં તો બીજી કોઈ પુસ્તિકા વાંચી શકો છો કે નહીં તે આપણે જોઈ શકીએ છીએ કે આપણે ફરીથી આવર્તક ભાગને દૂર કરી શકીએ છીએ જે આપણે વાસ્તવમાં ઝડપી-શોર્ટોઈ ચલાવી શકીએ છીએ, જે આપણે ઉકેલવા માગીએ છીએ. તેથી, તેના પરિણામ સ્વરૂપે ક્લોસ્ટોર્ટ પ્રણાલીમાં સૌથી કાર્યક્ષમ સોર્ટિંગ એલ્ગોરિધમ્સમાંથી એક બની ગયું છે અને જ્યારે અમારી પાસે એક સ્પેડશીટ જેવી ઉપયોગીતા છે કે જ્યાં અમારી પાસે એક બટન છે, જે આ સ્ટાંભની સોર્ટ કહે છે, વધુ વખત તેઓ તે નથી આંતરિક એલ્ગોરિધમ જે અમલમાં આવે છે તે વાસ્તવમાં ક્લોસોર્ટ છે, આપણે જોયું કે પાયથોન એક ફંક્શન I ડોટ સોર્ટ ધરાવે છે જે અમને સાઇન સૂચિને સોર્ટ કરવા માટે પરવાનગી આપે છે. તમે કદાચ પૂછી શકો છો, દાખલા તરીકે, એલ્ગોરિધમ સોર્ટ કેવી રીતે કરવું એ પાયથોન નો ઉપયોગ કરે છે; ઘણી વખત તે ઝડપીસ્પોર્ટ હશે. તેમ છતાં, કેટલાક કિસ્સાઓમાં આમ છતાંમને એલ્ગોરિધમ સૂચિમાંના મૂલ્યો વિશે નક્કી કરશે અને મૂલ્યોના પ્રકાર અનુસાર અલગ પ્રકારની સોર્ટિંગ એલ્ગોરિધમનો અમલ કરશે, પરંતુ

ડિફોલ્ટ સામાન્ય રીતે ક્વિકોકોર્ટ છે. તેથી, આગળ વધતા પહેલાં, આપણે અમારા દાવાનો પ્રયાસ અને માન્યતા કરીએ છીએ કે ક્લોસ્ટોર્ટનું સૌથી ખરાબ કેસ વર્તન વાસ્તવમાં પહેલાથી સોંટ કરેલી સૂચિમાં સૌથી ખરાબ કેસ ઇનપુટના વર્ણન સાથે બંધાયેલું છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 06:22)

અહીં, આપણે આપણી પાયથોન ને ઝડપી પ્રકારની અમલીકરણ કરતા પહેલા કરીએ છીએ જેમાં આપણે પહેલા લખેલ કોડને પુનરાવર્તિત કર્યો છે. હવે, આપણે બીજું કાર્ય લખીશું જે નીચે પ્રમાણે કરશે. તે વારંવાર બે અનુક્રમણિકાઓ દ્વારા યાદીના ઘટકોને શફલ કરશે અને ફક્ત તેમને સ્વેપ કરશે. આ અમને રેંજ આઉટપુટની સંભાળ રાખવાની મંજૂરી આપે છે અને તેને યોગ્ય રીતે રેન્ડમ શફલ બનાવી દે છે; અહીં તેના માટેનો કોડ છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 06:52)

તે ખૂબ જ સરળ છે, તમે રેન્ડમ નામના પાયથોન લાઇબ્રેરીનો ઉપયોગ કરો છો જે તમને કેટલાક ફંક્શન્સને રેન્ડમ નંબરો અને જે વસ્તુઓ છે કે જે આ લાઇબ્રેરીમાં છે તે આમાંથી એક છે. રેન્ડોમ રેન્જ 0 માં પૂર્ણાંક 1 થી લંબાઈ 1 ની લંબાઈ પેદા કરે છે. તેથી, આપણે તેને એક યાદી આપીએ છીએ અને અમે વારંવાર બે અનુક્રમણિકા j અને k પસંદ કરીએ છીએ, જે શ્રેણી 0 થી લંબાઈની લંબાઈ 1 સુધીની છે અને આપણે લેજ અને એલ.કે. ઘણી વખત આપણે આ કરીએ છીએ? સારું, આપણે આ કિસ્સામાં મોટી સંખ્યામાં કહીએ છીએ કે આપણી પાસે સૂચિમાં 10000 તત્વો છે, આપણે 5000 વખત આ કરીશું આપણે તેને 2 ગણી લંબાઈ કરીશું. ચાલો જોઈએ કે આ કેવી રીતે કાર્ય કરે છે, આપણે સામાન્ય રીતે પિથન દ્વારા તરીકે લોડ કરીએ છીએ અને ત્યારબાદ આપણે ટૂંક સમયમાં આયાત કરીએ છીએ અને ત્યારબાદ આપણે રેન્ડમાઇઝ કરવું આયાત કરીએ છીએ. તેથી, તમે આ કરી શકો છો, તમે બહુવિધ ફાઇલોમાં પાયથોન કાર્યો લખી શકો છો અને બીજા પછી એક આયાત કરી શકો છો અને તેઓ બધા લોડ થશે. હમણાં પહેલાં આપણે 1 ને કહીશું ઉદાહરણ તરીકે, આ સૂચિ હોઈ શકે છે. તેથી, ચાલો SYS નો સમાવેશ કરીએ અને તે રિકર્ડન સીમા પ્રક્રિયાને સમાપ્ત કરીએ કારણ કે આપણે જાણીએ છીએ કે તે આપણને મારવા માટે છે. તેથી, મોટી રિકર્ડનની મર્યાદા સેટ કરો અને હવે અમે એક નિશ્ચિતપણે મોટી સૂચિ ગોઠવીએ છીએ જે અમે છેલ્લે 7500 થી નીચે જમણી બાજુએ કર્યું છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 08:19)

અને આપણે જોયું કે, જો આપણે આ સૂચિને ઝડપથી સોંટ કરવાનો પ્રયાસ કરીએ તો તે લાંબો સમય લે છે કારણ કે તે 7500 છે અને તે એક ખરાબ કેસ છે. આપણે હવે શું કરવાનો પ્રયાસ કરીએ છે અને તે એકસાથે સોંટ થાય પછી પણ થશે, કારણ કે તે સોંટ થયા પછી પણ હજુ પણ સૌથી ખરાબ કેસ ઇનપુટ છે હવે તે ચડતા ક્રમમાં છે. તેથી, ઉત્તરતા ક્રમમાં અને ચડતા ક્રમમાં બંને લાંબા સમય લે છે. હવે, આપણે ધારીએ છીએ કે હું એલ રેન્ડાઈઝ કરું છું. તેથી, જો તમે 1 પર નજર કરો તો તમે જોઈ શકો છો કે સંખ્યાઓ ક્રમમાં નથી. તો, તમે 7500 અને 2000 ની વચ્ચે 6000 અને તેથી વધુ જોઈ શકો છો. અમારું દાવા એ છે કે આ ઝડપથી જશે અને ખરેખર તમે જોશો કે જો તમે તેના પર ક્લોસોર્ટ ચલાવો છો તો તે લગભગ તરત જ પાછો ફરે છે અને તે એટલા માટે નથી કારણ કે ક્લોસ્ટોર્ટ વધુ ઝડપી બની ગયું છે, તે ઇનપુટના હુકમને કારણે છે, કારણ કે ફરીથી અમારી પાસે ઝડપી સ્કોલ છે છટણી યાદી ફરીથી તે ધીમું રહ્યું છે આ ફક્ત ખૂબ જ અસરકારક રીતે દર્શાવે છે કે જો આપણે આ યાદીને રદ કરો અને અમે તરત જ બહાર આવે ત્યારે ઝડપી રન ચલાવો, પરંતુ જો આપણે તેને રેન્ડમાઇઝ કરવું નહીં અને જો આપણે ખરેખર સોંટ કરેલ સૂચિને સોંટ કરવા માંગીએ છીએ તો તે લાંબા સમય લે છે. તેથી, આપણે વાસ્તવમાં તપાસ કરી શકીએ છીએ, દાખલા તરીકે, જો આપણે આ સૂચિમાં પાછા જઈએ અને આપણે તેને 10000 કદાચ 15000 જેટલી મોટી કંઈક કહીએ છીએ અને પછી આપણે તેને રદ કરો અને પછી આપણે તેને સોંટ

કરીએ છીએ કે તે સહેજ ઝડપી આવે છે. તેથી, તે અમારા દાવાને માન્ય કરે છે કે સરેરાશ પર ક્લોસ્ટોર્ટ ઝડપી છે, તે ત્યારે જ છે જ્યારે તમે તેને આ અત્યંત ખરાબ ઇનપુટ્સ આપો છો જે એકવાર પહેલાથી જ છટણી કરવામાં આવે છે જે તે ગરીબ રીતે વર્તે છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 09:54)

હવે, એક વધુ માપદંડ છે કે જ્યારે કોઈ માહિતીને સોંટ કરે ત્યારે જાણવાની રહે છે. તેથી, ઘણી વાર આ સોર્ટિંગ બહુવિધ વિશેષતાઓ પરના તબક્કામાં થાય છે, ઉદાહરણ તરીકે, તમારી પાસે ક્વિઝ અથવા પરીક્ષણ પછી રોલ સૂચિમાંના મૂળાક્ષર ક્રમમાં સૂચિબદ્ધ વિદ્યાર્થીઓની સૂચિ હોય, તેઓ બધા ગુણ મેળવે છે. હવે, તમે માર્ક્સની ક્રમમાં તેમને સૂચિબદ્ધ કરવા માગો છો, પરંતુ જ્યાં સંબંધો છે કે જ્યાં બે કે તેથી વધુ વિદ્યાર્થીઓ સમાન ગુણ ધરાવે છે જે તમે તેમને મૂળાક્ષર ક્રમમાં ગોઠવવાનું ચાલુ રાખવા માગો છો. તેથી, બીજા શબ્દોમાં કહીએ તો તમારી પાસે આલ્ફાબેટીક ક્રમમાં મૂળ હુકમ છે અને પછી તમે ગુણ લેતા હોવ અને ગુણ દ્વારા સોર્ટ કરીને અન્ય વિશેષતા મેળવી શકો છો જે સોર્ટિંગ પહેલાથી જ મૂળાક્ષર ક્રમમાં અસ્તિત્વમાં નથી. તે શું કહેવું છે તે એ છે કે જો અમારી મૂળ સૂચિમાં બે સૂચિની બે વસ્તુઓ હોય તો તે સમાન ક્રમમાં જાળવી રાખવી જોઈએ કારણ કે સોર્ટિંગ પછી તે હતી. તેથી, મને બે ઘટકો ન લેવા જોઈએ જે સોર્ટ કરતી વખતે સમાન અને સોર્ટ કરે છે, અને જ્યારે તમે સ્પ્રેડશીટની જેમ કંઈક વાપરી રહ્યા હો ત્યારે તે નિર્ણાયક હશે કારણ કે જો તમે એક કોલમ દ્વારા સોર્ટ કરો છો, તો તમે જે સોર્ટિંગ કર્યું છે બીજા કોલમ દ્વારા

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 11:04)

દુર્ભાગ્યવશ, આપણે જે રીતે વર્ણન કર્યું છે તે રસ્તો ઝડપી નથી કારણ કે જ્યારે પણ આપણે આ પાર્ટીશન તબક્કામાં કોઈ પાર્ટીશન વિસ્તારીએ છીએ અથવા પીવટને કેન્દ્ર તરફ ખસેડીએ છીએ તો આપણે શું કરી રહ્યા છીએ તે ઘટકોના હુકમને ખલેલ પહોંચે છે ક્રમમાંગોઠવાયેલનથી યાદીમાં પહેલેથી જ હતા તેથી, અમે અગાઉ એવી દલીલ કરી હતી કે આ ઓર્ડરને ખલેલ પહોંચાડવાથી કોઈ વાંધો નથી કારણ કે અમે તેને સોર્ટ કરવાના છીએ તે કોઈપણ રીતે, પરંતુ સોર્ટિંગ સ્થિર હોવા જરૂરી છે તે બાબત છે જો ત્યાં એક કારણ હતું કે શા માટે આ ઘટકો વર્તમાન હરોળ માટે ખાસ ક્રમમાં નથી, પરંતુ વિવિધ લક્ષણ માટે અને અમે તેમને ફરતે ખસેડીએ છીએ ત્યારે અમે મૂળ છટણી હુકમનો નાશ કરી રહ્યા છીએ. બીજી તરફ, સોર્ટ કરો, આપણે જોઈ શકીએ છીએ તે વાસ્તવમાં સ્થિર છે જો તમે ખાતરી કરો કે અમે હંમેશા એક બાજુથી પસંદ કરીએ છીએ, જો કિંમતો બરાબર હોય તો. તેથી, જ્યારે અમે ડાબી અને જમણી મર્જ કરીએ છીએ જ્યારે અમારી પાસે સમાન કેસ છે, તો આપણે તત્વને ડાબેથી અંતિમ સૂચિમાં અથવા જમણે મૂકવું પડશે. જો આપણે સતત ડાબી બાજુ પસંદ કરીએ તો તે હંમેશા જમણી બાજુના ડાબી બાજુ પરના તત્વોને ડાબેથી રાખશે અને તેથી, તે સ્થિર સોર્ટ રહેશે તેવી જ રીતે, નિવેશ સોર્ટ પણ સ્થિર રહેશે જો તમે ખાતરી કરો કે આપણે ફક્ત પાછળની વસ્તુઓ જ ખસેડીએ છીએ, જો તે પાછળની તરફ જઈને સખત રીતે નાના હોય અને આપણે કંઈક શોધીએ જે વર્તમાન મૂલ્ય જેટલું છે, તો અમે તેમાં સામેલ થવાનું બંધ કરીએ છીએ. તેથી, નિવેશ સોર્ટ સોર્ટ કરો જેવું સોર્ટ કરો સોર્ટ કરો સોર્ટ કરો સોર્ટ કરો સોર્ટ કરો સોર્ટ કરો જેમ કે અમે વર્ણવેલ છે તે સ્થિર નથી, તેમ છતાં વધુ સાવચેત અમલીકરણ કરવું અને તેને સ્થિર કરવું શક્ય છે.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રો. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
અઠવાડિયું - 04 વ્યાખ્યાન - 05 ટુપલ્સ અને ડિક્સીઅન્સ

(સ્લાઇડ ટાઇ નો ઉલ્લેખ કરો: 00:01)

અમે આ પ્રકારની એકસાથે સોંપણી, જ્યાં આપણે ડાબી બાજુ પર ત્રણ નામ લઈએ છીએ અને જમણી બાજુએ ત્રણ મૂલ્યો તેમને સોંપીએ છીએ, અને અમે આ રાઉન્ડ કૌંસમાં બંધ કરીએ છીએ. તેથી, રાઉન્ડ કૌંસ સાથેના મૂલ્યોની આ પ્રકારની શ્રેણીને ટુપલ્સ કહેવામાં આવે છે. સામાન્ય રીતે અમે જોડીઓ, ટ્રિપલ, ક્વેડ્રપ્યુલેલ્સ વિશે વાત કરીએ છીએ, પરંતુ સામાન્ય રીતે જ્યારે તે K ના મૂલ્યમાં જાય છે ત્યારે આપણે તેને k ટુપલ્સ કહીએ છીએ. પાયથોન પર, ટુપલ્સ પણ માન્ય કિંમતો છે. તમે એક નામ લઈ શકો છો; અને તેને મૂલ્યોનો એક ટુપલ્સ સોંપો. દાખલા તરીકે, આપણે x - કોઓર્ડિનેટ્સ 3.5 અને 4.8 સાથે બે પરિમાણીય બિંદુ લઈ શકીએ અને કહીએ તે બિંદુની કિંમત 3.5 અલ્પવિરામ 4.8 છે, અને આ એક સૂચિ નથી, પરંતુ એક ટુપલ્સ છે. અને એક મિનિટમાં આપણે જોઈશું કે તુવલ શું છે. એ જ રીતે, આપણે કહી શકીએ કે તારીખ ત્રણ ભાગો, દિવસ, એક મહિના અને એક વર્ષનો બનેલો છે; અને આપણે ત્રણ મુલ્ય અથવા ટ્રિપલ માં જોડાઈ શકીએ છીએ. તેથી, ટુપલ એક યાદીની જેમ વર્તે છે, તેથી તે એક પ્રકારનો ક્રમ છે. તેથી, શબ્દમાળાઓ અને સૂચિની જેમ, ટ્યૂપલમાં તમે શ્રેણીના એક ઘટકને બહાર કાઢી શકો છો. તેથી, આપણે કહી શકીએ છીએ કે પોઇન્ટની 0 થી મુલ્ય x કોઓર્ડિનેટ છે. આ x ના મૂલ્ય x ને x ના x સંકલન માટે સોંપે છે, અથવા આપણે કહી શકીએ કે આપણે માત્ર 7 અને 2013 માગીએ છીએ, તો આપણે તારીખ લઈએ છીએ અને સ્લાઇડસને એકથી લઈને અંત લાવીએ પછી આપણે મળશે 7 અલ્પવિરામ 2013. તેથી, તે ખૂબ જ અલગ પ્રકારની શ્રેણીની જેમ વર્તે છે જે બરાબર સ્ટ્રિંગ્સ અને યાદીઓ જે અમે અત્યાર સુધી જોયેલા છે, પરંતુ એક ટુપલ્સ અને સૂચિ વચ્ચેનો તફાવત એ છે કે એક ટુપલ્સ અસમર્થનીય છે

(સ્લાઇડ સમયનો સંદર્ભ લો: 01:44)

તો, આ કિસ્સામાં ટુપલે વધુ વણસ્યા કરે છે, ઉદાહરણ તરીકે, તારીખ 8 ની તારીખથી આપણે બદલી શકતા નથી, તે સ્થાને તારીખને બદલીને મૂલ્ય 8 દ્વારા બદલવું જોઈએ. આ યાદીમાં શક્ય છે, પરંતુ એક ટુપલ્સ માં નથી તેથી, ટુપલ્સ અસંબદ્ધ સિક્વન્સ છે, અને આ બાબત શા માટે છે તે એક મિનિટમાં તમે જોશો

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 02:10)

ચાલો યાદીઓ પર પાછા જઈએ. સૂચિ મૂલ્યોનો ક્રમ છે, અને સંલગ્ન રીતે આ ક્રમાંક સાથે સંકળાયેલ 0 થી શરૂ કરીને યાદીની લંબાઈ 1 ની લંબાઈ સાથે જોડાયેલી છે. તેથી, સૂચિ જોવાનું એક વૈકલ્પિક રીત એ છે કે તે દરેક સ્થાનને નક્કા કરે છે મૂલ્ય; આ કિસ્સામાં, કિંમતો પૂર્ણાંક છે અમે કહી શકીએ છીએ કે આ સૂચિ 1 એ ડોમેન 0, 1, 2, 3, 4 થી ગાણિતિક અર્થમાં નક્કા અથવા કાર્ય છે પૂર્ણાંકોની શ્રેણી; અને ખાસ કરીને, તે એલ 13 ને 13, એલ 4 નું 72 અને તેથી જ્યાં આપણે આને ફંક્શન મુલ્ય તરીકે જોઈ રહ્યા છીએ. તેથી, આ વિશે વિચારવાનો પ્રોગ્રામ લેંગ્વેજ રીત એ છે કે 0, 1, 2, 3, 4 ને કીઓ કહેવામાં આવે છે. તેથી, આ તે મૂલ્યો છે જેની સાથે અમારી કેટલીક વસ્તુઓ સંકળાયેલ છે. તેથી, અમે 1 સાથે સંકળાયેલ

આઈટમ માટે શોધ કરીશું અને આપણે પાછા 46 મેળવીશું. આપણી પાસે કીઓ છે અને સૂચિમાં અનુરૂપ એન્ટ્રીઝને મૂલ્ય કહેવામાં આવે છે. તેથી, સૂચિ કીઝને મૂલ્યો માટે સાંકળવાની એક રીત છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 03:19)

આપણે કીઓને 0 થી n ની -1 ની માત્રાની શ્રેણી સિવાય બીજું કોઈ સેટ સિવાય બીજું કશું આપીને આ ખ્યાલને સામાન્ય કરી શકીએ છીએ. તેથી, ઉદાહરણ તરીકે કી એ સ્ટ્રિંગ હોઈ શકે છે. તેથી, આપણે એવી એક યાદી જોઈવીએ છીએ જેમાં કોઈ ખેલાડીના નામે મૂલ્યોને આપણે અનુક્રમિત કરીએ છીએ. તેથી, દાખલા તરીકે, તમે કહી શકો છો કે દરેક ખેલાડીના નામ માટે મૂલ્ય સંકળાયેલું મૂલ્ય શું છે. તેથી, ધવનનો સ્કોર 84 છે, પૂજારાનો સ્કોર 16 છે, કોહલીનો સ્કોર 200 છે, અમે આ બધાને વધુ સામાન્ય યાદીમાં સંગ્રહિત કરીએ છીએ જ્યાં સૂચિ મૂલ્યો સ્થાન દ્વારા અનુક્રમિત નથી, પરંતુ આ કિસ્સામાં કેટલાક વધુ અમૂર્ત કી દ્વારા ખેલાડીનું નામ. પાયથોન એક શબ્દકોશ કહે છે, કેટલીક અન્ય પ્રોગ્રામિંગ ભાષામાં તેને એસોસિએટીવ એરે પણ કહેવાય છે. તેથી, તમે સાહિત્યમાં આ જોઈ શકો છો. તેથી, અહીં મૂલ્યોનો સંગ્રહ છે જે કી દ્વારા એક્સેસ કરવામાં આવે છે જે માત્ર એક પદ નથી, પરંતુ કેટલાક મનસ્વી ઇન્ડેક્સ અને પાયથોન નું નિયમ એ છે કે કોઈપણ અવિચારી મૂલ્ય કી હોઈ શકે છે

(સ્લાઈડ સમયનો ઉલ્લેખ કરો: 04:26)

આનો અર્થ એ છે કે તમે શબ્દમાળાઓનો ઉપયોગ કરી શકો છો જે અસમર્થનીય છે. અને અહીં દાખલા તરીકે તમે ટુપલ્સનો ઉપયોગ કરી શકો છો, પરંતુ તમે સૂચિનો ઉપયોગ કરી શકશો નહીં. અને એક શબ્દકોશનો બીજો લક્ષણ એ છે કે યાદીની જેમ, તે પરિવર્તનીય છે; આપણે કી સાથે મૂલ્ય લઈ શકીએ છીએ અને તેને બદલી શકીએ છીએ. તેથી, અમે પુજારાના સ્કોરને બદલી શકીએ છીએ, જો તમે 72 થી અસાઇનમેન્ટ દ્વારા જોઈ શકો છો અને આ ફક્ત હાલના શબ્દકોશ લેશે અને પુજારા સાથે સંકળાયેલ મૂલ્યને 16 થી 72 સુધી બદલીશું. તેથી, શબ્દકોષો સ્થાનાંતરિત થઈ શકે છે અને તેથી તે અસ્પષ્ટ છે યાદીઓની જેમ

(સ્લાઈડ સમયનો ઉલ્લેખ કરો: 04:59)

આપણે પાયથોન ને કહેવું જોઈએ કે કેટલાક નામ શબ્દકોશ છે અને તે એક સૂચિ નથી. તેથી, આપણે સર્પાકાર કૌસ દ્વારા એક ખાલી શબ્દકોશ દર્શાવવું. તેથી યાદ રાખો, આપણે સૂચિ માટે ચોરસ કૌસનો ઉપયોગ કરીએ છીએ. તેથી, જો તમે તે શબ્દકોશની શરૂઆત કરવા માંગતા હોવ કે જે આપણે પહેલા જોયું હતું તો આપણે પ્રથમ કહીએ કે ટેસ્ટ 1 એ અહીંના કૌસને આપીને ખાલી શબ્દ છે અને ત્યારબાદ આપણે ધવન અને પૂજારા જેવા તમામ ખેલાડીઓને મૂલ્ય આપવાનું શરૂ કરી શકીએ છીએ. તેથી પર તેથી, નોંધ લો કે આ તમામ ત્રણ સિક્વન્સ અને પ્રકારની વસ્તુઓ છે જે અલગ છે, તેથી અલબત્ત શબ્દમાળાઓ માટે, અમે ડબલ કોડેડ અથવા સિંગલ કોડેડનો ઉપયોગ કરીએ છીએ; સૂચિ માટે આપણે ચોરસ કૌસનો ઉપયોગ કરીએ છીએ; ટુપલ્સ માટે, અમે રાઉન્ડ કૌસનો ઉપયોગ કરીએ છીએ; અને શબ્દકોશ માટે, અમે કૌસનો ઉપયોગ કરીએ છીએ. તેથી, પાયથોન ને સંકેત આપવાની એક નિશ્ચિત રીત છે કે આપણે કયા પ્રકારનું સંગ્રહ નામ સાથે જોડાઈ રહ્યા છીએ, જેથી અમે તેને યોગ્ય કાર્યવાહી સાથે ચલાવી શકીએ જેનો તે પ્રકારના સંગ્રહ માટે નિર્ધારિત છે. તેથી, એક વાર ફરી એક શબ્દકોશ માટે, કી કોઈપણ અસંબંધ મૂલ્ય હોઈ શકે છે; તેનો અર્થ એ કે, તમારી યાદી પૂર્ણાંક હોઈ શકે, તે એક ફ્લોટ હોઈ શકે છે, તે એક બૂલ હોઈ શકે છે, તે સ્ટ્રિંગ હોઈ શકે છે, તે એક ટુપલ્સ હોઈ શકે છે, તે સૂચિ અથવા શબ્દકોશ નથી. તેથી, આપણી પાસે સૂચિ દ્વારા અથવા શબ્દકોશ દ્વારા અનુક્રમિત મૂલ્ય હોઈ શકતું નથી

(સ્લાઈડ સમયનો સંદર્ભ લો: 06:21)

તેથી, અમારી પાસે ઘણી સંખ્યામાં હોઈ શકે છે જેમ કે અમે નેસ્ટ કરેલ સૂચિ ધરાવે છે જ્યાં સૂચિ ધરાવતી સૂચિ હોઈ શકે છે અને પછી અમારી પાસે બે સૂચકાંકો 0th સૂચી ધરાવે છે અને પછી તેમની 0 ની સૂચિમાં પ્રથમ સ્થાન છે, અમે કીઓ બે સ્તર હોઈ શકે છે જો તમે બહુવિધ ટેસ્ટ મેચોમાં સ્કોર્સનો ટ્રેક રાખવા માંગતા હો, તો બે શબ્દકોશો કર્યા વિના આપણે એક શબ્દકોશ કરી શકો છો જ્યાં પ્રથમ કી ટેસ્ટ મેચ ટેસ્ટ 1 અથવા ટેસ્ટ 2 છે, અને બીજી કી ખેલાડી છે ઉદાહરણ તરીકે, સમાન પ્રથમ કી સાથે, ઉદાહરણ તરીકે, સમાન બીજી પ્રથમ કી સાથે, ટેસ્ટ 1 અને ટેસ્ટ 2; તમે ધવન માટે બે અલગ અલગ સ્કોર્સનો ટ્રેક રાખી શકો છો. તેથી, ટેસ્ટ 1 માંનો સ્કોર અને ટેસ્ટમાં સ્કોર 2. અને ટેસ્ટ ટેસ્ટમાં એક કરતા વધારે ખેલાડી હોઈ શકે છે, જેમ કે અહીં છે; અમે કોહલી અને ધવન બંને આ એક છે. જો તમે પાચથોન ને શબ્દકોશમાં દર્શાવવાનો પ્રયત્ન કરો છો, તો તે આ કૌંસમાં તમને આ પ્રકારની સર્પાકાર કૌંસ નોટેશનમાં બતાવશે, જ્યાં દરેક પ્રવેશ કોલની દ્વારા અલગ પાડવામાં આવતી કિંમતો દ્વારા અનુસરવામાં કી હશે અને પછી તે સૂચિની જેમ હશે. અલ્પવિરામથી અલગ અને જો અમારી પાસે બહુવિધ કીઓ હોય તો આવશ્યકપણે આ શબ્દકોશમાં એક સંપૂર્ણ એન્ટ્રી છે, અને કી ટેસ્ટ 1 માટે, મારી પાસે આ મૂલ્યો છે; કી ટેસ્ટ 2 માટે, મારી પાસે આ મૂલ્યો છે અને આંતરિક રીતે તેઓ ફરીથી શબ્દકોશ છે, તેથી તેમની પાસે પોતાની કી મૂલ્ય છે

(સ્લાઇડ સમયનો સંદર્ભ લો: 07:39)

ચાલો જોઈએ કે તે કેવી રીતે કાર્ય કરે છે અમે ખાલી શબ્દકોશ સાથે શરૂ કરીએ સ્કોર અને હવે આપણે કીઓ બનાવવાની ઇચ્છા રાખીએ છીએ, તેથી ધારણા કરો કે આપણે સ્કોર ટેસ્ટ કહીશું, ધવન 76 બરાબર છે આ અમને એક ભૂલ આપવા જઈ રહ્યું છે, કારણ કે અમે તેને કહ્યું નથી કે સ્કોર ટેસ્ટ 1 એ એક શબ્દકોશ છે તેથી, એ નથી જાણતું કે આપણે ધવન શબ્દ સાથે વધુ ઇન્ડેક્સ કરી શકીએ છીએ. તેથી, અમારે પહેલા કહેવું પડશે કે માત્ર સ્કોર એ કોઈ શબ્દકોશ નથી, તેથી તે ટેસ્ટ 1 છે અને સંભવ છે કારણ કે આપણે તેનો ઉપયોગ કરીશું, તેથી તે ટેસ્ટ છે. હવે આપણે પાછા જઈ શકીએ અને પ્રથમ ટેસ્ટમાં ધવનનો સ્કોર પ્રથમ ટેસ્ટમાં 76 અને તમે બીજા ટેસ્ટને 27 થી સેટ કરી શકો છો અને કદાચ અમે પ્રથમ ટેસ્ટમાં કોહલીનો સ્કોર 200 માં મૂકી શકીએ. હવે, જો તમે મને બતાવવા માટે પૂછો કે સ્કોર્સ શું જુએ છે, તો આપણે જોઈ શકીએ છીએ કે તેને બે કી ટેસ્ટ સાથે બાહ્ય શબ્દકોશ છે 1, પરીક્ષણ 2 જેમાં દરેક નેસ્ટેડ શબ્દકોશ છે પુનરાવર્તિત શબ્દકોશોમાં, અમારી બે કીઓ ધવન અને કોહલી છે, જે મૂલ્ય તરીકે 76 અને 200 રન કરે છે. ટેસ્ટ 2 માં, એક શબ્દકોશ તરીકે ધવનને કી તરીકે પ્રવેશ મળે છે અને 27 એ સ્કોર છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 08:52)

જો તમે કોઈ શબ્દકોશ પર પ્રક્રિયા કરવા માંગતા હો તો અમને તમામ મૂલ્યો દ્વારા ચલાવવાની જરૂર પડશે; અને કિંમત દ્વારા ચલાવવા માટેનો એક રસ્તો બધા મૂલ્યો કીઓ બહાર કાઢવા અને વળાંક દ્વારા દરેક મૂલ્ય બહાર કાઢવાનો છે. તેથી, એક કાર્ય ડી ડોટ કીઓ છે જે ડિક્શનરીની કીઓની ક્રમ આપે છે t. અને આપણે જે લાક્ષણિક વસ્તુ કરીએ છીએ તે દરેક કીમાં છે, t ડોટ કીઝમાં ડી ચોરસ કૌંસ સાથે કંઈક છે. તેથી, બધી કીઝ પસંદ કરો આ યાદીમાં દરેક સ્થાન માટે કહીને એવું છે કે તે પદ પર મૂલ્યનું કંઈક કરો. આ યાદીમાં દરેક કી માટે કંઈક છે જે તેની સાથે સંકળાયેલું મૂલ્ય ધરાવે છે. હવે એક વાત ધ્યાનમાં રાખવી જોઈએ કે જે હું એક મિનિટમાં બતાવીશ તે એ છે કે ડટ ડોટ કોઈપણ ધારણાત્મક ક્રમમાં નથી. તેથી, ઝડપથી કી સાથે મૂલ્ય પરત કરવા માટે આંતરિક રીતે ઓપ્ટિમાઇઝ કરવામાં આવે છે તે કીઓને જે ક્રમમાં દાખલ કરવામાં આવે છે તેમાં સાચવી શકશે નહીં. તેથી, તમે કેવી રીતે ડી ડોટ કીઝ અમને પ્રસ્તુત કરાશે તે વિશે કોઈ અનુમાન નથી કરી શકતા. આ કરવા માટે એક રીત એ સોર્ટ કરેલ ઇન્ક્શનનો ઉપયોગ કરવો. આપણે સોર્ટ ડી ડોટ કીઝ, પ્રક્રિયા ડીકેમાં k માટે કહી શકીએ છીએ, અને આ સોર્ટ ઇન્ક્શન મુજબ સોર્ટ કરેલ ક્રમમાં અમને કી આપશે. તેથી, સોર્ટ એલ એક કાર્ય છે જે અમે અત્યાર સુધી જોયો નથી; સોર્ટ કરેલું l ની એક સોર્ટ કરેલી નકલ આપે છે, તે સંશોધિત કરતું નથી. અમે અત્યાર સુધી જે જોયું છે તે ડોટ સોર્ટ છે, જે કાર્ય છે જે યાદી લે છે અને તેને સ્થાનાંતરિત કરે છે. તેથી, sorted l ઇનપુટ સૂચિ લે

છે, તે કોઈ યથાવત છોડી દે છે, પરંતુ તે એક સોર્ટ વર્ઝન આપે છે. બીજી વસ્તુ ધ્યાનમાં રાખવી એ છે કે તે માને છે કે ડોટ કી એક યાદી છે તે લાલચ છે, તે એક સૂચિ નથી; તે શ્રેણી અને અન્ય વસ્તુઓ જેવું છે. તે ફક્ત મૂલ્યોનો ક્રમ છે જે તમે ની અંદર ઉપયોગ કરી શકો છો, તેથી આપણે યાદીની સૂચિનો ઉપયોગ ડીએટ કીઝની બહારની યાદી બનાવીએ.

(સ્લાઇડ સમય જણાવો: 10:46)

તો, ચાલો દાવો કરીએ છીએ કે કીઓ કોઈ ચોક્કસ ક્રમમાં રાખવામાં નથી આવી તેથી, ચાલો ખાલી શબ્દકોશ સાથે શરૂ કરીએ. અને હવે ચાલો દરેક અક્ષર અને એન્ડ્રી માટે બનાવીએ જે તે અક્ષર જેવું જ છે. તેથી, આપણે l, b, c, d, e, f, g, h, i, di, di માં l માટે કહી શકીએ છીએ. તેથી, આ શું કહે છે, તેથી જ્યારે તમે શબ્દમાળામાં l માટે કહીએ છીએ ત્યારે તે શબ્દમાળામાં દરેક અક્ષરને જાય છે, તેથી d ની કી સાથે એ કહેવું છે કે મૂલ્ય એ છે, d એ કી b છે તે મૂલ્ય b છે અને તેથી જમણે. તેથી, હવે, જો હું તમને કહીશ કે દા શું છે, તો તમે, શું છે, તે હું છું. હવે નોંધ લો કે કીઓ એ, b, c, d, e, f, g, h, i ક્રમમાં દાખલ કરવામાં આવે છે, પરંતુ જો હું d ડોટ કીઝ માટે પૂછું છું તો તે તેને કેટલાક અત્યંત રેન્ડમ ક્રમમાં બનાવે છે. તેથી, ઈ પ્રથમ છે અને એક રસ્તો નીચે છે અને તેથી વધુ. ત્યાં કોઈ ચોક્કસ ક્રમમાં નથી કે તમે આમાંથી મેળવી શકો છો. તેથી, આ ફક્ત એટલું જ ભાર મૂકે છે કે જે ક્રમમાં ડિલિવરીમાં કીઓ દાખલ કરવામાં આવી છે તે ક્રમમાં હોઈ શકશે નહીં જેમાં તેઓ કીઓ ફંક્શન દ્વારા પ્રસ્તુત થાય છે. તેથી, તમારે હંમેશા તેની ખાતરી કરવી જોઈએ કે જો તમે ચોક્કસ ક્રમમાં કીઓની પ્રક્રિયા કરવા માંગો છો, તો ખાતરી કરો કે તમે તે ક્રમને જાળવી રાખશો જ્યારે તમે કીઓને બહાર કાઢશો કે જે તમે ધારતા નથી કે કીઓ કોઈપણ ક્રમમાં બહાર આવશે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 12:06)

અન્ય કોઈ રીતે શબ્દકોશમાં મૂલ્યો ચલાવવા માટે છે, ડોટ મૂલ્યનો ઉપયોગ કરવો. તેથી, d ડોટ કી આવર્તન કી ક્રમમાં છે, d ડોટ વેલિઝ તમને અમુક ક્રમમાં કિંમતો આપે છે. તેથી, આ ઉદાહરણ તરીકે x માં l માટે કહે છે. તેથી, તમે તે મૂલ્યો મેળવો છો જે તમને તે સ્થાનો નહીં મળે. અહીં તમે મૂલ્યો મેળવો છો જે તમને કીઓ મળી નથી. તેથી, જો તમે શબ્દકોશમાંથી બધી કિંમતો ઉમેરવા માંગતા હો, તો તમે કુલ 0 થી શરૂ કરીને શરૂ કરી શકો છો, અને દરેક મૂલ્ય માટે, તમે તેને હા ઉપર જ ઉમેરી શકો છો. તેથી, તમે દરેક એકને ટેસ્ટ l ડોટ મૂલ્યમાં પસંદ કરી શકો છો અને તેને કુલ પર ઉમેરી શકો છો.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 12:50)

તો, તમે ઓપરેટરનો ઉપયોગ કરીને શબ્દકોશમાં કી તરીકે ચકાસવા માટે કરી શકો છો, જેમ કે સૂચિ માટે જ્યારે તમે x ને l નો ઉલ્લેખ કરો છો તો તે તમને સાચું કહે છે જો x એ l ની કિંમત છે x l થી છે, તે તમને અન્યથા ખોટા કહે છે. આ જ કીઓ વિશે સાચું છે તેથી, જો હું વ્યક્તિગત બેટ્સમેનો માટે રન ઉમેરવો હોય તો પણ મને ખબર નથી કે દરેક ટેસ્ટ મેચમાં બેટિંગ કરી હોય તો. તેથી, હું દરેક કીઓ માટે કહું છું, આ કિસ્સામાં, ધવન અને કોહલી, જે મેં પહેલાથી જ સેટ ન કરેલું છે તે શબ્દકોશનો પ્રારંભ કર્યો છે, હું તે સેટ કરી શકત. તેથી, ધવન સાથે કુલ 0, કુલ કોહલી 0 ની સાથોસાથ કુલ છે. અમારા પુનરાવર્તિત શબ્દકોશમાં પ્રત્યેક મેચ માટે, જો ધવનને તે મેચમાં બેટ્સમેન તરીકે દાખલ કરવામાં આવે છે, તેથી જો ધવન કે મેચ માટે કુલ સ્કોર તરીકે કી તરીકે દેખાય છે પછી અને માત્ર પછી તમે સ્કોર ઉમેરો, કારણ કે જો તે દેખાતું નથી તો તે મેચને એક્સેસ કરવા માટે ગેરકાયદેસર છે. તેથી, આ સુનિશ્ચિત કરવાનો એક રસ્તો છે કે જ્યારે તમે કોઈ શબ્દકોશમાંથી મૂલ્યનો ઉપયોગ કરો છો, ત્યારે કી વાસ્તવમાં અસ્તિત્વમાં છે, તમે ફંક્શનમાં ઉપયોગ કરી શકો છો.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 14:00)

અહીં યાદ રાખવાની રીત છે કે કોઈ શબ્દ સૂચિથી અલગ છે. જો હું ખાલી શબ્દકોષથી શરૂ કરું તો હું એક ચાવી સોંપી, જે અત્યાર સુધી જોવામાં આવી નથી, શબ્દકોશમાં કોઈ સમસ્યા નથી તે શબ્દકોશમાં આ કી શામેલ કરવા બરાબર છે, જો તે ડી પહેલેથી જ અસ્તિત્વમાં હશે તો સુધારાશે. તેથી, ક્યાં તો તમે અપડેટ કરો છો અથવા તમે શામેલ કરો છો. આ સૂચિથી વિપરીત છે, જ્યાં તમારી પાસે ખાલી સૂચિ હોય અને પછી કોઈ સ્થાન પર દાખલ કરવાનો પ્રયાસ કરો જે અસ્તિત્વમાં નથી, તો તમને ઇન્ડેક્સ ભૂલ મળે છે

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 14:42)

કોઈ શબ્દકોશમાં, તે ચાવીરૂપ રીતે નવી કીઓ સમાવવા વિસ્તારી શકે છે અથવા ચાવીનું અસ્તિત્વ છે કે નહીં તે કી પર આધાર રાખે છે. સારાંશ માટે, એક શબ્દકોષ તમારી સૂચિમાંની કી કરતાં મૂલ્યની વધુ સાનુકૂળ જોડાણ છે; એકમાત્ર અવરોધ એ છે કે પાયથોન ને લાદવામાં આવે છે કે બધી કી અનિવાર્ય મૂલ્યો હોવા જોઈએ. તમારી પાસે કીઓ હોઈ શકતી નથી, જે પરિવર્તનીય મૂલ્યો છે તેથી, અમે શબ્દકોશો નો ઉપયોગ કરી શકતા નથી અથવા પોતાની જાતને કીઓ તરીકે સૂચિબદ્ધ કરી શકતા નથી, પરંતુ તમે આમાંના બહુવિધ સ્તરો સાથે નેસ્ટેડ શબ્દકોશો કરી શકો છો. બીજું વસ્તુ એ છે કે આપણે શબ્દકોશમાં તમામ કીઝને ચક્રમાં રાખવા માટે ૮ ડોટ કીનો ઉપયોગ કરી શકીએ છીએ, અને એ જ ડી ડોટ મુલ્ય, પરંતુ ક્રમમાં જે આ કીઓ ડી ડોટ કીમાંથી ઉદ્ભવે છે તે અનુમાનિત નથી. તેથી, અમે તેને કંઈક બીજું કરવા માટે તેને સોર્ટ કરવાની જરૂર છે જો અમે તેમને અનુમાનિત ક્રમમાં પ્રક્રિયા કરવા માગીએ છીએ તેથી, તે તારણ કાઢે છે કે તમે જોશો કે ખરેખર શબ્દકોશ છે જે પાયથોનને ટેક્સ્ટ ફાઇલો અથવા કોષ્ટકોમાંથી માહિતીને હેરફેર કરવા માટે ખરેખર ઉપયોગી ભાષા બનાવે છે, જો તમારી પાસે અલ્પવિરામથી જુદાં જુદાં મૂલ્ય કોષ્ટકો કહેવાય છે, તો તે સ્પ્રેડશીટમાંથી લેવામાં આવે છે કારણ કે તે પછી સ્તંભ હેડિંગનો ઉપયોગ કરી શકે છે અને મૂલ્યો એકઠા કરી શકો છો અને તેથી. તેથી, તમારે તમારા પ્રોગ્રામિંગ કુશળતામાં શબ્દકોશ સમજવું અને આત્મસાત કરવું જોઈએ, કારણ કે આ તે છે જે સ્ક્રિપ્ટને ચાલાકી કરવા માટે પાયથોન ને અત્યંત શક્તિશાળી ભાષા બનાવે છે.

પ્રોગ્રામિંગ ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રોફેસર. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ
ઇન્ડિયન ઇન્સ્ટીટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 04 લેક્ચર - 06

ફંક્શનની વ્યાખ્યાઓ આપણે જોયું છે કે આપણે વ્યાખ્યાયિત કરતી વખતે દલીલ સમૂહ માટે મૂલ્યોને બદલીને વિધેયોને મૂલ્યો પસાર કરીએ છીએ. કાર્ય

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 00:02)

અને, આ અસરકારક રીતે ગર્ભિત સોંપણી હોવા જેટલું જ છે. તેથી, જ્યારે આપણે પાવર x^n કહીએ છીએ, અને આપણે તેને 3 અને 5 સાથેના મૂલ્યો કહીએ છીએ, તો આપણી પાસે આ અસાઇનમેન્ટ x બરાબર 3 અને 5 ની બરાબર 5 છે. તે ખરેખર ત્યાં નથી, પરંતુ તે આ પ્રમાણે છે, આ કોડ દ્વારા ચલાવવામાં આવે છે આ સોંપણી પહેલા અને અલબત્ત, તેને કાર્ય તરીકે બોલાવવાનો ફાયદો એ છે કે, આપણે કાર્ય વ્યાખ્યામાં x અને n નો ઉલ્લેખ કરવાની જરૂર નથી; તે કોલ સાથે આવે છે તેથી, x અને n ના જુદા જુદા મૂલ્યો માટે, આપણે સમાન કોડ એક્ઝીક્યુટ કરીશું.

(સ્લાઇડ સમયનો સંદર્ભ લો: 00:02)

પ્રથમ વસ્તુ કે જે પાયથોન ને સહેલાઈથી કરવા માટે પરવાનગી આપે છે, તે ઓર્ડર દ્વારા ન જાય; તે તે નથી, પ્રથમ x છે, અને બીજો એ n છે; અમે, જો તમને ઓર્ડર યાદ ન હોય, પરંતુ અમે કિંમતો, તેમને સોંપેલ નામો ખબર નથી, અમે ખરેખર તેમને દલીલ ના નામનો ઉપયોગ કરીને કહી શકો છો તેથી, આપણે આ વસ્તુને ઉલટાવી શકીએ છીએ, અને કહી શકીએ, પાવર પર કોલ કરો. અને હું જાણું છું કે, x એ નીચલું મૂલ્ય છે જે મને ખબર છે કે તે એન નો પાવર એન છે, પરંતુ મને યાદ નથી કે શું એક્સ પ્રથમ આવે છે, અથવા n પ્રથમ આવે છે. હું કહી શકું છું, ચાલો આપણે સલામત રીતે રમીએ અને એન 5 ની બરાબર, બરાબર 4 ની બરાબર કહીએ અને આ યોગ્ય રીતે દલીલના નામ પ્રમાણે મૂલ્યને સાંકળશે અને સ્થાન અનુસાર નહીં.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 01:24)

પાયથોન ની અન્ય એક સરસ સુવિધા એ છે કે, તે કેટલીક દલીલોને છોડી દેવાની પરવાનગી આપે છે અને સર્વથા મૂળભૂત મૂલ્યો ધરાવે છે. યાદ રાખો કે, અમે આ પ્રકારનું રૂપાંતરણ કાર્ય એ પૂર્ણાંકનું નિર્ધારિત કર્યું છે, જે શબ્દમાળા લેશે અને તેને પૂર્ણાંક તરીકે પ્રસ્તુત કરવાનો પ્રયાસ કરશે, જો તે પૂર્ણાંકની માન્ય રજૂઆત છે. તેથી, અમે એમ કહીએ છીએ કે, જો આપણે તેને "76" શબ્દ આપીએ તો પૂર્ણાંક તેને 76 નંબર પર રૂપાંતરિત કરે છે. જો બીજી તરફ, આપણે તેને "A5" શબ્દ આપીએ તો પૂર્ણાંક તેને "A5" વાસ્તવમાં એક ભૂલ પેદા કરશે હમણાં, તે તારણ આપે છે કે, પૂર્ણાંક ખરેખર એક દલીલનું કાર્ય નથી, પરંતુ બે દલીલો; અને બીજું દલીલ આધાર છે. તેથી, આપણે તેને શબ્દમાળા આપીએ છીએ અને તેને આધાર B માં સંખ્યામાં રૂપાંતરિત કરીએ છીએ, અને જો આપણે બાય આપતા નથી, તો, મૂળભૂત રીતે b નું મૂલ્ય 10 હોય છે. તેથી, પહેલાનાં પૂર્ણાંક રૂપાંતરમાં શું થઈ રહ્યું છે, તે આ પ્રમાણે છે છતાં અમે કહીએ છીએ, આધાર 10 સાથે "76", પરંતુ ત્યારથી, અમે 10 ને પૂરું પાડતા નથી, પાયથોન ને તે કિંમત લેવા માટેની પદ્ધતિ છે જે પૂરું પાડવામાં આવતી નથી, અને મૂળભૂત મૂલ્ય 10 નું સ્થાન. હવે, જો આપણે તે મૂલ્ય છે,

તે પછી, ઉદાહરણ તરીકે, આપણે "A5" નો અર્થ પણ બનાવી શકીએ છીએ. જો તમારી પાસે 16 આધાર હોય, તો તમે શાળામાં 16 મા ધોરણમાં અભ્યાસ કર્યો હોય તો તમને ખબર હોત કે, તમારી પાસે શૂન્યથી 9 અંક છે, પરંતુ બેઝ 16 સંખ્યા 15 સુધી છે. તેથી, 9 ની ઉપરની સંખ્યા સામાન્ય રીતે A નો ઉપયોગ કરીને લખવામાં આવે છે, બી, સી, ડી, ઇ, એફ. તેથી, A અનુલક્ષે છે, આપણે 10 ની 10 માં 10 સંખ્યામાં શું વિચારીશું. તેથી, જો તમે આધાર 16 માં "A5" લખો તો, આ સોળમી સ્થિતિ છે અને આ તે પોઝિશન છે તેથી, આપણી પાસે 16 ગુણ્યા 10 છે, કારણ કે એ 10, વત્તા 5 છે. સંખ્યાત્મક દ્રષ્ટિએ, આ 165 ને યોગ્ય રીતે પાછું આપશે. આ પાયથોન માં કેવી રીતે કામ કરે છે આ આંતરિક રીતે હશે, જો તમે સમાન કાર્ય લખી હોત, તો તમે તેને લખી શકો છો. તેથી, તમે દલીલો આપો છો, અને દલીલ માટે કે જેના માટે તમે વૈકલ્પિક મૂળભૂત દલીલ માંગો છો, તમે ફક્શન વ્યાખ્યામાં મૂલ્ય પ્રદાન કરો છો. તેથી, આ વ્યાખ્યા શું કહે છે તે છે, ઇન્ટ લે છે 2 આર્ગ્યુમેન્ટ્સ અને b અને b એ ધારવામાં આવે છે 10, અને તેથી, વૈકલ્પિક છે; જો વ્યક્તિ બીજા દલીલને અવગણશે, તો તે આપમેળે મૂલ્ય 10 લેશે. નહીં તો, તે ફક્શન કોલ દ્વારા પ્રદાન કરેલ મૂલ્ય લેશે. ડિફોલ્ટ મૂલ્ય ફક્શન વ્યાખ્યામાં પ્રદાન કરવામાં આવે છે અને જો તે પેરામીટર અવગણવામાં આવે છે, તો તેના બદલે મૂળભૂત મૂલ્યનો ઉપયોગ થાય છે. પરંતુ, યાદ રાખવું એક બાબત એ છે કે, આ મૂળભૂત મૂલ્ય એવી વસ્તુ છે જે ફક્શન વ્યાખ્યાયિત થવા પર ઉપલબ્ધ હોવાનું માનવામાં આવે છે. તે કંઈક ન હોઈ શકે જેનું ગણતરી થાય છે, જ્યારે કાર્ય કહેવાય છે. તેથી, અમે ઝડપી સોર્ટ અને મર્જ કરો સોર્ટ અને બાઈનરી શોધ જેવા વિભિન્ન કાર્યો જોયાં હતાં, જ્યાં અમને એરેની શરૂઆતની સ્થિતિ અને અંતની સ્થિતિ સાથે પાસ કરવાની ફરજ પડી હતી. હવે, આ મધ્યવર્તી કોલ્સ માટે દંડ છે, પરંતુ, જ્યારે આપણે ખરેખર એક સૂચિને સોર્ટ કરવા માંગતા હોઈએ ત્યારે, પ્રથમ વખત આપણે તેને શૂન્ય સાથે કોલ કરવાનું યાદ રાખવું જોઈએ, અને સૂચિની લંબાઈ. તેથી, તે કહેવું આકર્ષાય છે કે, આપણે કાર્યને કંઈક વ્યાખ્યાયિત કરીએ છીએ જે પ્રારંભિક એરેને પ્રથમ દલીલ તરીકે લે છે, અને પછી, મૂળભૂત રીતે ડાબી સીમાને શૂન્ય તરીકે લે છે, જે સાડું છે, અને તે જમણી સરહદ છે એ લંબાઈ. પરંતુ, સમસ્યા એ છે કે, આ જથ્થો, A ની લંબાઈ એ પોતે પર આધારિત છે. તેથી, જ્યારે કાર્ય વ્યાખ્યાયિત કરવામાં આવે છે, ત્યાં હશે, અથવા A માટે મૂલ્ય હોઈ શકશે નહીં અને A માટે તમે જે મૂલ્ય પસંદ કર્યું છે, તે ન હોય તો, તે લંબાઈ મૂળભૂત તરીકે લેવામાં આવશે. અમે ક્લિકોઈને કોલ કરીએ ત્યારે તે ગતિશીલ રીતે ગણતરી કરવામાં આવશે નહીં. તેથી, આ કામ કરતું નથી, અધિકાર. તેથી, જ્યારે તમારી પાસે ડિફોલ્ટ મૂલ્ય હોય, ત્યારે ડિફોલ્ટ મૂલ્ય એક સ્ટેટિક મૂલ્ય હોવું જોઈએ, જે પરિભાષાને પ્રથમ વખત વાંચવામાં આવે ત્યારે નિર્ધારિત કરી શકાય છે, જ્યારે તે એક્ઝેક્યુટ થતી નથી. અહીં એક સરળ પ્રોટોટાઇપ છે. ધારો કે આપણી પાસે 4 દલીલો A, b, c, ડી સાથે કાર્ય છે અને અમારી પાસે છે, c પાસે મૂળભૂત મૂલ્ય 14 છે, અને ડી પાસે મૂળભૂત મૂલ્ય 22 છે. પછી, જો તમારી પાસે માત્ર 2 દલીલો છે, તો પછી, આ એ અને બી સાથે સંકળાયેલું હોવું જોઈએ, અને તેથી, તેને એફ 13, 12, અને ગુમ થયેલા દલીલ માટે અને ડી માટે તમને ડિફોલ્ટ 14 અને 22 મળશે. બીજી તરફ, તમે 3 દલીલો આપી શકો છો, જેમાં કેસ, એક 13 બને છે, b 12 પહેલાની જેમ બને છે, અને C 16 બની જાય છે, પરંતુ d એ અસ્પષ્ટ છે; તેથી, તે ડિફોલ્ટ મૂલ્ય પસંદ કરે છે આને 13, 12, 16 અને ડિફોલ્ટ મૂલ્ય 22 ની એફ તરીકે અર્થઘટન કરવામાં આવે છે. તેથી, ધ્યાનમાં રાખવાની બાબત એ છે કે, મૂળભૂત મૂલ્યો સ્થાન દ્વારા આપવામાં આવે છે. આખા ફક્શનમાં કોઈ રીત નથી એમ કહી શકાય કે, 16 ડી માટે આપવું જોઈએ, અને મને સી માટે ડિફોલ્ટ મૂલ્ય જોઈએ છે; તમે ફક્ત અંતથી પોઝિશન દ્વારા મૂલ્યોને છોડી શકો છો તેથી, જો મારી પાસે 2 મૂળભૂત મૂલ્યો છે, અને જો હું તેમને ફક્ત બીજાનો ઉલ્લેખ કરવા માગું છું, તો તે શક્ય નથી; તેને પુનઃક્રમાંકિત કરવા માટે કાર્યને ફરી નિર્ધારિત કરવું પડશે. તેથી, તમારે ખાતરી કરવી જ જોઈએ કે, જ્યારે તમે આ ડિફોલ્ટ મૂલ્યોનો ઉપયોગ કરો છો, ત્યારે તે અંતે આવે છે અને તેઓ સ્થાન દ્વારા ઓળખાય છે અને તેને ભળવું નહીં, અને આ વસ્તુઓનો રેન્ડમ રીતે સંયોજિત કરીને જાતે મૂંઝવતા નથી. તેથી, દલીલોનો ક્રમ મહત્વપૂર્ણ છે

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 06:43)

એક ફંક્શન ડિફર્મેશન એક ફંક્શન બોડીને એક નામથી સાંકળે છે. તે કહે છે, નામ એફને વિધેય તરીકે અર્થઘટન કરવામાં આવશે જે કેટલાક દલીલો લે છે અને કંઈક કરે છે. ઘણી રીતે, પાયથોન નામના મૂલ્યની કોઈપણ અન્ય સોંપણી જેવા અર્થઘટન કરે છે. દાખલા તરીકે, આ મૂલ્યને અલગ અલગ રીતે, બહુવિધ રીતે, શરતી રીતે વ્યાખ્યાયિત કરી શકાય છે. તેથી, જેમ તમે આગળ વધો છો, એક કાર્ય ફરીથી વ્યાખ્યાયિત કરી શકાય છે, અથવા ગણતરી કેવી રીતે થાય છે તેના આધારે તેને અલગ અલગ રીતે વ્યાખ્યાયિત કરી શકાય છે. અહીં એક શરતી વ્યાખ્યાનું ઉદાહરણ છે. તમારી પાસે એક શરત છે; જો તે સાચું હોય, તો તમે એક રીતે એફ વ્યાખ્યાયિત કરો છો; અન્યથા, તમે એફ અન્ય રીતે વ્યાખ્યાયિત. તેથી, આ વ્યાખ્યા કઈ રીતે હાથ ધરવામાં આવી હતી તેના પર આધાર રાખીને, પાછળથી એફ ની કિંમત પર અલગ અલગ હશે. હવે, આ કહેવું નથી કે, આ કરવા માટે એક ઇચ્છનીય વસ્તુ છે, કારણ કે તમે શું કરી રહ્યા છે તે અંગે મૂંઝવણમાં હોઈ શકે છે. પરંતુ, એવી પરિસ્થિતિઓ છે કે જ્યાં તમે એક રીતે, અથવા બીજી રીતે એફ લખી શકો છો, ગણતરીને કેવી રીતે આગળ વધી રહી છે તેના આધારે; અને પાયથોન તમે આ કરવા માટે પરવાનગી આપે છે. સંભવતઃ, પાયથોન ને પ્રાધાન્ય આપવા માટે, આ ખૂબ ઉપયોગી નથી; પરંતુ, આ જાણવું ઉપયોગી છે કે આવી આવશ્યકતા અસ્તિત્વમાં છે અને ખાસ કરીને, તમે આગળ વધો અને તમે આગળ વધો ત્યારે એફ ફરીથી નિર્ધારિત કરી શકો છો. બીજી વસ્તુ જે તમે પાયથોન માં કરી શકો છો, જે તમને થોડી વિચિત્ર લાગે શકે છે, તે તમે હાલના ફંક્શનને લઈ શકો છો, અને તેને નવા નામ પર મેપ કરી શકો છો. તેથી, આપણે એક કાર્ય એફ વ્યાખ્યાયિત કરી શકીએ છીએ, જે આપણે કહીએ છીએ, નામ એફ સાથેના સહયોગી, આ કાર્યનું શરીર; પછીના તબક્કે, આપણે કહી શકીએ કે જો ઇક્વલ ટુ એફ અને આનો અર્થ શું છે તે, આપણે એ, બી, સીનો જો નો પણ ઉપયોગ કરી શકીએ છીએ અને તેનો અર્થ A, B, C ના F તરીકે થાય છે. તેથી, જો તમે ફંક્શનમાં જોનો ઉપયોગ કરો છો, તો તે બરાબર તે જ વિધેયનો ઉપયોગ કરશે, તે એક બીજાને એક સૂચિ સોંપવા જેવું છે, અથવા એક શબ્દકોશ બીજામાં અને તેથી વધુ. હમણાં, શા માટે તમે આ કરવા માંગો છો? તેથી, એક ઉપયોગી રીત છે કે જેમાં તમે આ કરી શકો છો, આ સુવિધાનો ઉપયોગ અન્ય કાર્યમાં ફંક્શન પસાર કરવાનો છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 08:40)

ધારો કે, આપણે તેનાં દલીલ માટે n એ આપેલ કાર્યને એફ કરવા માંગીએ છીએ, તો આપણે સામાન્ય રીતે આ કાર્યને લખી શકીએ છીએ, જે 3 દલીલો લે છે. પ્રથમ કાર્ય છે, બીજું દલીલ છે, અને ત્રીજું વખતની સંખ્યા છે, પુનરાવર્તનો. તેથી, અમે તમને પ્રદાન કરેલ મૂલ્યથી શરૂ કરીએ છીએ, અને તમને કહેવામાં આવે છે તેટલી વખત, તમે કાર્યને પુનરાવર્તન કરો છો. તેથી, ચાલો આપણે એક નક્કર ઉદાહરણ જોઈએ. ધારો કે, અમે x નું ફંક્શન સ્કેલ વ્યાખ્યાયિત કર્યું છે, જે ફક્ત x વખત x આપે છે; અને હવે આપણે કહી શકીએ, વેલ્યુ 5 થી બે વાર સ્કેલ લાગુ કરો. તેથી, આનો અર્થ શું છે, 5 નો ચોરસ લાગુ કરો, અને તે પછી, તેનો વર્ગ; તેથી, ચોરસ બે વાર કરો. તેથી, તમને 5 સ્કેલ 25 મળે છે; 25 સ્કેલ 625. તેથી, અહીં શું થઈ રહ્યું છે તે છે, ચોરસ એ એફ ને સોંપવામાં આવી રહ્યું છે, 5 એ x ને સોંપવામાં આવી રહ્યું છે, અને 2 નો n ને સોંપવામાં આવી રહ્યો છે. આ બરાબર છે જેમ આપણે કહ્યું હતું કે, પહેલાં, જેવું, કહીએ તો એફ બરાબર ચોરસ છે. તેથી, આ અર્થમાં, કાર્ય નામ લેવા અને તેને અન્ય નામે સોંપવા માટે સક્ષમ છે, તે ખૂબ જ ઉપયોગી છે, કારણ કે, તે આપણને એક જગ્યાએથી બીજા સ્થાને કાર્યોને પસાર કરવાની પરવાનગી આપે છે અને તે વિધેયને અન્ય વિધેયની અંદર, તે જાણ્યા વિના આગળ શું તે કાર્ય છે

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 10:08)

આનો એક પ્રાયોગિક ઉપયોગ સોર્ટ જેવા કાર્યોને કસ્ટમાઇઝ કરવા માટે છે કેટલીકવાર, અમે વિવિધ માપદંડ પર આધારિત મૂલ્યો સોર્ટ કરવાની જરૂર છે. તેથી, આપણી પાસે અમૂર્ત સરખામણી કાર્ય હોઈ શકે છે, જે પ્રથમ 1 ત્રિપરિજ્ઞ, જો શૂન્ય હોય તો 2 આર્ગ્યુમેન્ટ્સ બરાબર હોય, અને વત્તા 1 જો પ્રથમ દલીલ બીજા કરતાં મોટી હોય તો 1 ઓછા આપે છે. તેથી, શબ્દમાળાઓની તુલના કરતી વખતે,

આપણા મનમાં શબ્દોની તારણોના 2 અલગ અલગ રીત હોઈ શકે છે, અને આપણે આ 2 અલગ અલગ રીતે સોર્ટ કરતી વખતે તફાવત તપાસવું જોઈએ અમે એક ક્રમમાં હોઈ શકે છે જેમાં આપણે શબ્દકોશમાં શબ્દોની તુલના કરીએ છીએ. તેથી, એબ જેવા શબ્દમાળા એબ પહેલાં આવે છે, કારણ કે, બીજી સ્થિતિ બી કરતાં નાની છે. તેથી, આ બાદબાકી 1 થશે, કારણ કે, પ્રથમ દલીલ બીજા દલીલ કરતાં નાની છે. જો, બીજી બાજુ, આપણે લંબાઈથી શબ્દમાળાઓની તુલના કરવા માંગીએ છીએ, તો પછી, એ જ દલીલ આપણને વત્તા 1 આપશે, કારણ કે, એબની લંબાઈ 3 છે અને તે અબ કરતાં લાંબી છે. તો, આપણે સોર્ટ ફંક્શન લખી શકીએ છીએ, જે એક યાદી લે છે, અને બીજી દલીલ લે છે, જે છે, સૂચિમાંની એન્ટ્રીઓની સરખામણી કેવી રીતે કરવી. સોર્ટ ફંક્શનને જાણ કરવાની જરૂર નથી કે સૂચિમાંના તત્વો શું છે; જ્યારે પણ તેને મનસ્વી મૂલ્યોની સૂચિ આપવામાં આવે છે, તે પણ તેની સાથે કેવી રીતે સરખામણી કરવી તે પણ કહેવામાં આવે છે. તેથી, આ કરવાની જરૂર છે, આ ફંક્શનને 2 મૂલ્યો પર લાગુ કરો, અને તપાસો કે તેનો જવાબ ઓછા 1, શૂન્ય અથવા વત્તા 1 છે અને તેનો અર્થ એ છે કે તે કરતાં ઓછો, સમાન અથવા વધારે છે. પછી, જો તમે ઇચ્છો તો, તમે તેને અગાઉની સુવિધા સાથે ભેગા કરી શકો છો, જે છે, તમે તેને મૂળભૂત વિધેય આપી શકો છો. જો તમે સોર્ટ ફંક્શનને નિર્દિષ્ટ કરતા નથી, તો એક ગર્ભિત કાર્ય હોઈ શકે છે જે સોર્ટ ફંક્શનનો ઉપયોગ કરે છે; નહિતર તે સરખામણી ફંક્શનનો ઉપયોગ કરશે જે તમે પ્રદાન કરો છો.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 11:50)

સારાંશ માટે, કાર્ય વ્યાખ્યાઓ માત્ર નામોનાં મૂલ્યોના અન્ય સોંપણીઓની જેમ વર્તે છે. તમે ફંક્શનમાં નવી વ્યાખ્યા ફરીથી આપી શકો છો. તમે તેને સંબંધિત રીતે વ્યાખ્યાયિત કરી શકો છો. નિર્ણાયક રીતે, તમે એક ફંક્શનનો ઉપયોગ કરી શકો છો અને નિર્દેશ કરી શકો છો, નામ બિંદુને બીજા કાર્યમાં, અને જ્યારે અન્ય કાર્યોમાં વિધેયો અને સોર્ટિંગ જેવી પરિસ્થિતિઓને પસાર કરીએ ત્યારે આ પૂર્ણ રૂપે ઉપયોગમાં લેવાય છે, તમે તમારી તુલનાત્મક કાર્યને પસાર કરીને તમારા સોર્ટિંગ વધુ સાનુકૂળ બનાવી શકો છો કિંમતોને યોગ્ય જે આપણે સોર્ટ કરીશું.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રોફેસર. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
અઠવાડિયું - 04 લેક્ચર - 07

સૂચિની લાગણી ઘણી વાર, આપણે એક સંપૂર્ણ યાદીમાં કંઈક કરવા માંગીએ છીએ.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 00:02)

દાખલા તરીકે, અમે એક્સની અમુક મેળવેલ મૂલ્ય એફ દ્વારા સૂચિમાં દરેક વસ્તુને બદલવા માંગીએ છીએ. તેથી, આપણે નીચે પ્રમાણે લૂપ લખીશું, તેમાં દરેક x માટે, X ની x ની જગ્યાએ x ને બદલો. હવે, આપણે આ કરવા માટે ફંક્શન લખી શકીએ છીએ, જે વિવિધ યાદીઓ અને 1 ની જુદી જુદી કિંમતો માટે કરે છે. અમે કહી શકીએ, લાગુ સૂચિને વ્યાખ્યાયિત કરો, જે કાર્ય એફ અને સૂચિ 1 લે છે, અને પ્રત્યેક x અને 1 માટે, તમે ફક્ત x ના f દ્વારા આ x ને બદલો છો; અને ત્યારથી 1, સૂચિ ફેરફારવાળા વસ્તુ છે, આ ફોન ફંક્શનમાં પણ સૂચિને અપડેટ કરશે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 00:45)

પાયથોનમાં બિલ્ટ-ઇન કાર્ય નકશો છે, જે ચોક્કસપણે આ કરે છે. તેથી, નકશા FL લાગુ પડે છે, બદલામાં 1 ની દરેક ઘટક. હવે, જો તમે એમ વિચારતા હોવ કે, જો તમે કોઈ સૂચિ લો, તો, x 1, x 2, અને તમે નકશાને લાગુ કરો છો, અને તમને એક્સ 1 ની એફ મળે છે, તે નકશાનું આઉટપુટ બીજી સૂચિ હોવું જોઈએ, કમનસીબે, પાયથોન 3 માં, અને આ પાયથોન 3 અને પાયથોન 2 વચ્ચેની બીજો તફાવત છે, નકશાનું આઉટપુટ સૂચિ નથી. તેથી, તમારે સૂચિ ફંક્શનનો ઉપયોગ કરવો પડશે જેમ આપણે પહેલાં કર્યું હતું. તેથી, તમારે સૂચિ મેળવવા માટે નકશાની સૂચિ કહેવાની જરૂર છે, અને જો તમે મેપમાં સૂચિમાં મે માટે ફ્લોપ કરો છો, તો તમે નકશાના આઉટપુટનો સીધી રીતે લૂપ માટે ઉપયોગ કરી શકો છો, અથવા તમે મે માટે નકશામાં કહી શકો છો, આ કામ કરશે તેથી, તમારે સૂચિ સંકેતનો ઉપયોગ કરવાની જરૂર નથી, જો તમે ફક્ત ઇન્ડેક્સ મેનૂ માગો છો, પરંતુ જો તમે તેને સૂચિ તરીકે ઉપયોગ કરવા માંગતા હો, તો તમારે તેને રૂપાંતરિત કરવા માટે સૂચિ ફંક્શનનો ઉપયોગ કરવો જોઈએ. અને, આ ખૂબ જ થાય છે, રેંજ અને ડી ડોટ કીઝ જેવા ફંક્શન્સ સાથે અને તેથી. આ બધી વસ્તુઓ છે જે આપણને મૂલ્યોની સિક્વન્સ આપે છે. આ સિક્વન્સ સંપૂર્ણપણે યાદીઓ નથી; તેઓ વિધેયો માટે ઉપયોગ કરી શકાય છે, પરંતુ જો તમે તેમને યાદીઓ તરીકે ઉપયોગમાં લેવા માગો છો, અને તેમને યાદીઓ તરીકે ચાલાકી આપો છો, તો તમારે તેમને સૂચિમાંથી સૂચિ ફોર્મમાં રૂપાંતરિત કરવા માટે યાદીનો ઉપયોગ કરવો પડશે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 02:10)

બીજી કોઈ વસ્તુ જે આપણે સામાન્ય રીતે કરવા માંગીએ છીએ તે એક યાદી લેવાની અને મૂલ્યોને કાઢવાનો છે જે ચોક્કસ મિલકતને સંતોષે છે. તેથી, આપણી પાસે સંખ્યાની સૂચિ કહેવાતી પૂર્ણાંકોની સૂચિ હોઈ શકે છે, અને આમાંથી, આપણે પ્રાઈમ્સની સૂચિ બહાર કાઢવાની ઇચ્છા રાખી શકીએ છીએ. અમે એમ કહીને શરૂ કરીએ છીએ કે, આપણે જે પ્રાઈમ્સ જોઈએ છે તે ખાલી છે, અને અમે સંખ્યા યાદી મારફતે ચલાવીએ છીએ, અને તે સૂચિમાં દરેક નંબર માટે, આપણે પરીક્ષણને લાગુ કરીએ છીએ, તે એક મુખ્ય છે; જો તે એક મુખ્ય છે, તો પછી અમે અમારી આઉટપુટ યાદીમાં સૂચિ જોડીએ છીએ. તેથી,

આપણે x_1, x_2 ની સૂચિ સાથે શરૂઆત કરીએ છીએ. અને પછી, અમે પરીક્ષણને લાગુ કરીએ છીએ અને તેમાંના કેટલાક પસાર થઈ જશે, અને તેમાંના કેટલાક સફળ થશે, તેમાંના કેટલાક નિષ્ફળ જશે, અને અંતમાં, વસ્તુઓ જ્યાં પણ પસાર થાય ત્યાં, તે વસ્તુઓ આઉટપુટમાં બહાર આવશે. તેથી, સામાન્ય રીતે, આપણે એક પસંદ કાર્ય લખી શકીએ છીએ જે મિલકત અને સૂચિ લે છે, અને તે યાદીમાં દરેક તત્વ દ્વારા જઈને પેટા સૂચી બનાવે છે, તપાસ કરતી હોય કે મિલકત ધરાવે છે કે નહીં, અને તે ગુણધર્મ કે જે તે ધરાવે છે તે સબ સૂચિમાં પસંદગી અને અમારા અગાઉના નકશા કાર્યમાં તફાવત એ છે કે, મિલકત એક મનસ્વી કાર્ય નથી; તે એલ એ બધાને ચાલાકી આપતું નથી, તે જે કરે છે તે છે, તે તપાસ કરે છે કે મિલકત સાચી છે કે નહીં. મિલકત કાર્ય હશે જે સૂચિમાં એક તત્વ લેશે, અને અમને સાચું કે ખોટું કહેશે; જો તે સાચું હોય, તો તે આઉટપુટમાં નકલ થઈ જાય છે; જો તે ખોટી છે, તો તે કાઢી નાખવામાં આવે છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 03:46)

આ માટે બિલ્ટ-ઇન ફંક્શન પણ છે તેને ફિલ્ટર કહેવામાં આવે છે. તેથી, ફિલ્ટર ફંક્શન પી લે છે, જે પ્રત્યેક ઘટક માટે સાચું કે ખોટા આપે છે, અને તે ચોક્કસપણે બહાર નીકળે છે કે જે ઉપદ્રવી 1 છે, જેના માટે 1 માં દરેક વસ્તુ, જે ઉપદ્રવી satisfies p માં આવે છે. ચાલો એક નક્કર ઉદાહરણ જોઈએ.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 04:00)

ધારો કે, અમારી પાસે 0 થી 99 ની સંખ્યાઓની સૂચિ છે. આપણે પહેલા યાદીમાં ફક્ત સંખ્યાઓ પણ ખેંચી લેવા માગીએ છીએ. તે ફિલ્ટર કામગીરી છે; અને પછી, આ દરેક નંબરો માટે, આપણે તેમને ચોરસ કરવા માંગીએ છીએ. તેથી, અહીં, અમે ફિલ્ટર્સનો ઉપયોગ કરીને, પણ, યોગ્ય સંખ્યાઓ લઈએ છીએ, અને પછી, અમે ચોરસને મેપ કરીએ છીએ. પછી, અમને એક સૂચિ મળે છે. અને પછી, અલબત્ત, આ સૂચિ મળી છે, તો પછી આપણે તેને ઉમેરી શકીએ છીએ. આ રકમ આ કાર્યનો ભાગ નથી. જો આપણે સૌ પ્રથમ નંબરોના ચોરસને કાઢવા માગીએ, અને તે ફિલ્ટરનાં સંયોજનનો ઉપયોગ કરીને કરી શકાય છે, અને પછી, નકશા. ફિલ્ટર કરો, પ્રથમ અમને પણ સંખ્યાઓ આપે છે અને પછી નકશા આપણને ચોરસ આપે છે અને ચોરસ અહીં વ્યાખ્યાયિત કરે છે અને અહીં પણ વ્યાખ્યાયિત કરવામાં આવે છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 04:46)

તે નોટેશનનો ઉપયોગ કર્યા વિના, મેપ અને ફિલ્ટરને સંયોજિત કરવાની ખૂબ સુધડ રીત છે. ચાલો એક સરળ ગાણિતિક ઉદાહરણ દ્વારા, તે તરફ જઈએ. તેથી, તમે કદાચ સ્કૂલમાં, જમણા હાથથી, જમણા ખૂણાવાળા ત્રિકોણનો અભ્યાસ કરી શકો છો, કે જે પાયથાગોરસના પ્રમેય દ્વારા તમે જાણો છો કે જો x, y અને z અનુક્રમે બંને બાજુઓની લંબાઈ અને હાયપોટેન્યુઝ છે, તો પછી, એક્સ વર્ગ વત્તા વાય ચોરસ z ચોરસ હશે. તેથી, પાયથાગોરસ ટ્રિપલ પૂર્ણાંકનો સમૂહ છે, 3, 4 અને 5, ઉદાહરણ તરીકે, જેમ કે, x ચોરસ વત્તા વાય ચોરસ z ચોરસ છે; 3 ચોરસ 9 છે; 4 ચોરસ 16 છે; 5 ચોરસ 25 છે. ચાલો કહીએ, આપણે x, y અને z ની તમામ પૂર્ણાંક મૂલ્યો જાણવા માગીએ છીએ, જેની કિંમતો n ની નીચે છે, જેમ કે, x, y અને z એ પાયથાગોરસ ટ્રિપલ ઇસ્યુનું સ્વરૂપ છે. પરંપરાગત ગાણિતિક સંકેતલિપીમાં, તમે આ પ્રકારની અભિવ્યક્તિ જોશો. તે કહે છે, મને તમામ ત્રિપાદઓ x, y અને z આપો, જેમ કે આ બાર આવા માટે વપરાય છે; જેમ કે, x, y અને z , બધા 1 અને n વચ્ચે આવેલા છે. અને, વધુમાં, x ચોરસ વત્તા વાય ચોરસ z ચોરસ બરાબર છે. આ, કેટલાક અર્થમાં, જ્યાંથી આપણે મૂલ્યો મેળવીએ છીએ; આ વર્તમાન સેટ છે. અમે 1 થી n સુધીના એક્સ, 1 થી n સુધીના વાય, 1 થી n સુધીના ઝેડ અને અમે તમામ શક્ય સંયોજનોને એકસાથે મૂકીએ છીએ, પછી આપણે આપેલ પ્રોપર્ટીને સંતોષવા માટે તે સંયોજનો કાઢીએ, x સ્કવેર વ્યુ વાય ચોરસ બરાબર છે માટે z ચોરસ, અને તે છે કે અમે બહાર કાઢવા બહાર. સેટ થિયરીમાં, તેને સેટ ગમ કહેવામાં આવે છે. જૂની સેટમાં કેટલીક શરતી વસ્તુઓ લાગુ કરીને આ એક નવું સેટ બનાવવાની રીત છે. આ પણ એક પ્રકારનું ટ્રિપીંગ ઓપરેટર લાગુ કરતું છે; તે 3 અલગ સમૂહો, x ને 1

થી n, y 1 થી n, z એ 1 થી n લે છે, તેમને ટ્રીપલ્સમાં જોડે છે ફિલ્ટરિંગ પ્રક્રિયા છે કે જેના દ્વારા તમે ફક્ત તે ટ્રીપલ્સને ખેંચો છો, જ્યાં x ચોરસ વત્તા વાય ચોરસ z ચોરસ છે; અને પછી, ત્યાં એક હેરફેરનું પગલું છે, જ્યાં તમે તેમને એક ટ્રિપલ, એક્સ કોમમા વાય કોમા ઝેડમાં ભેગા કરો છો. પરંતુ, સામાન્ય રીતે, મુખ્ય બિંદુ એ છે કે તમે હાલના સમૂહોમાંથી એક નવો સેટ બનાવી રહ્યા છો. તેથી, શું પાયથોન કરે છે અને અન્ય ઘણી ભાષાઓ પણ, જેમાંથી પાયથોન ને પ્રેરિત કરવામાં આવે છે, તે અમને આ સૂચનને યાદીઓમાં વિસ્તૃત કરવા દે છે. આ વાસ્તવમાં કાર્યાત્મક પ્રોગ્રામિંગ નામના પ્રોગ્રામિંગની શૈલીમાંથી આવે છે, જે, જ્યાંથી આ પ્રકારના સંકેત મળે છે, અને પાયથોન તે ઉછીનું લીધું છે અને તે ખૂબ સારી રીતે કામ કરે છે. અહીં તે છે કે તમે અમારી અગાઉની વસ્તુ, જે અમે કહ્યું હતું કે, નીચે 100 ના અંકોની સંખ્યા પણ લખીશું. અગાઉ, અમે નકશા ફિલ્ટર વસ્તુ આપી હતી. તેથી, અમે કહ્યું હતું કે, અમે એક રેન્જ લેશે, અને અમે તે ક્રમશઃ ફિલ્ટર કરીશું, અને પછી, અમે ચોરસનો નકશો કરીશું. પાયથોન માં, સેટ નોટેશનમાંથી $\&\#39;$ માટે $\&\#39;$ પહેલાં, નીચે અંતર્ગત લંબ રેખા છે. તે માત્ર એક્સનું ચોરસ લે છે, હું 100 ની રેન્જમાં, જેમ કે, એક્સનો ઇંચ છે - આપણી પાસે અહીં 3 ભાગો છે. તેથી, આપણી પાસે જનરેટર છે, જે આપણને કહે છે કે ક્યાંથી મૂલ્યો પ્રાપ્ત કરવા. યાદ રાખો કે, સૂચિની ગમ અથવા સેટ ગમ, યાદીઓના હાલના સેટમાંથી મૂલ્યો ખેંચે છે, તેથી અમે સૌ પ્રથમ યાદી બનાવીએ છીએ. આ કિસ્સામાં, સૂચિ શ્રેણી 100, પરંતુ અમે અમારી અન્ય યાદીઓનો ઉપયોગ કરી શકીએ છીએ; અમે કોઈપણ માટે $\&\#39;$ હું $\&\#39;$ માટે ઉપયોગ કરી શકીએ છીએ. પછી, અમે તેને ફિલ્ટર લાગુ કરીશું, જે આ સૂચિમાંનાં મૂલ્યો છે, જે તમે જાળવી શકશો. અને પછી, તે દરેક મૂલ્ય માટે આપણે તેને કંઈક કરી શકીએ છીએ. આ કિસ્સામાં, આપણે સ્ક્વેર્ડ કર્યું છે અને તે આપણું આઉટપુટ હશે. આ રીતે આપણે નકશા અને ગાળકોનો ઉપયોગ કરીને નકશા અને ફિલ્ટરનો ઉપયોગ કરીને યાદી બનાવીએ છીએ, તમે ફક્ત $\&\#39;$ માટે $\&\#39;$ જનરેટર માટે, $\&\#39;$ if $\&\#39;$ ફિલ્ટર માટે, અને નકશો માત્ર ફિલ્ટરમાં જનરેટરના આઉટપુટને ઇન્કશન લાગુ પાડીને ગર્ભિત છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 08:52)

ચાલો પાછા પાયથાગોરસના ત્રણ ઉદાહરણ પર જઈએ. અમે x, y, z થી 100 નીચે બધા પાયથાગોરસનો ત્રિપુલો જોઈએ છે. આ પ્રમાણે, આપણે કહ્યું છે કે તે રેન્જમાં x, y અને z ના બધા મૂલ્યો મારફતે ચક્ર આવવાની જરૂર છે. તે પહેલાં જે કર્યું તે કરતાં થોડું વધારે જટિલ છે, જ્યાં ફક્ત એક જ જનરેટર છે, 0 થી 100 શ્રેણીમાં બધા મૂલ્યો. તે બહુવિધ સ્વરૂપો સાથે તેને લખવા માટે પૂરતી સરળ છે. તેથી, આપણે કહીએ છીએ કે, હું એક્સ કોમિયા અને અલ્પવિરામ ઝેડ કરવા માંગો છો, રેન્જ 100 માં x માટે, 100 માટે y માટે, શ્રેણી 100 માં z માટે, પૂરી પાડવામાં આવેલ, x સ્ક્વેર્ડ વત્તા y સ્ક્વેર્ડ બરાબર z સ્ક્વેર્ડ છે. તે $\&\#39;$ if $\&\#39;$ સાથે લખાયેલું છે હવે, ફક્ત સ્લાઈડ પર ફિટ કરવા માટે, મેં તેને બહુવિધ રેખામાં વિભાજિત કરી છે, પરંતુ વાસ્તવમાં, તે પાયથોન કોડની એક લીટી હશે. આમાં કયા ક્રમમાં જનરેટ કરવામાં આવશે? ઠીક છે, તે બરાબર નેસ્ટેડ લૂપની જેમ વર્તે છે. કલ્પના કરો, અમે જેમાં લૂપ લખ્યું હતું, જેમાં રેન્જ 100 માં x , રેન્જ 100 માં વાય માટે, 100 માં ઝેડ માટે, અને તેથી વધુ. તેથી, અહીં શું થાય છે તે, પ્રથમ - 0 નું મૂલ્ય x માટે નક્કી કરવામાં આવશે, અને પછી 0 નું મૂલ્ય વાય માટે નક્કી કરવામાં આવશે, પછી z માટે 0. પ્રથમ જોડીમાં, ત્રણ કે ત્રણ બહાર આવે તે 0, 0, 0 છે. પછી, ઝેડની કિંમત બદલાઈ જશે, અંદરના લૂપ બદલાશે. આગામી એક હશે 0, 0, 1. આ x છે, આ વાય છે, આ z છે. તેથી, આ રીતે, આપણે 0, 0, 99 સુધી તે ચાલુ રાખીએ. તેથી, જ્યારે તે 99 નો સ્કોર કરે છે, તો લૂપ માટે તે બહાર નીકળી જશે અને આપણે 1 પર જઈશું. તેથી, મને 0, 1, 0 મળશે, અને 0, 1, 90, 0, 1, 99 અને તેથી પર. અંદરની અંદર માટે, તેથી z ચક્ર પ્રથમ, પછી y , અને પછી x ચક્રમાં ધીમું રહેશે. તેથી, યાદ રાખો કે તે.

(Slide Time નો સંદર્ભ લો: 10:35)

ચાલો જોઈએ કે કેવી રીતે પાચથોન માં કામ કરે છે. ચાલો પ્રથમ ચોરસ વ્યાખ્યાયિત કરીને શરૂ કરીએ; એક્સ રીટર્ન x ટાઇમ્સ x નું ચોરસ; તે પછી, આપણે તપાસવા માટે ઇબેવેન x વ્યાખ્યાયિત કરી શકીએ છીએ, કે, બાકીની x 2 વડે ભાગ્યા છે તે 0 છે. તેથી, આપણી પાસે ચોરસ 8, 64 છે; 67 ખરેખર ખોટા હોવા જોઈએ; ઇસીવ 68 સાચી હોવી જોઈએ અને તેથી વધુ. હવે, અમારી સૂચિ ગમ છે. ચાલો ચોરસ x ના સમૂહને, રેન્જ 100 માં x માટે જોઈએ, જેમ કે x પણ છે. તો, આપણે તે જોઈ શકીએ છીએ કે, 0 ત્યાં છે. તેથી 0 ચોરસ, 2 ચોરસ, 4 ચોરસ, 6 ચોરસ અને તેથી વધુ. આ અમારી યાદી ગમ છે. હવે, ચાલો પાચથાગોરિયન ટ્રિપલ એક કરીએ. અમે જણાવ્યું હતું કે, આપણે એક્સ, વાય, ઝેડ એક્સ માટે રેન્જ 100 માં, રેન્જ 100 માં, 100 શ્રેણીમાં z માટે જોઈએ છે. આ અમારી 3 જનરેટર છે, x વખત x , વત્તા વાય ટાઇમ y , જે સમાન છે તેની શરત છે. z વખત z . હવે, તમે જે ઘણાં બધાં આવ્યાં છે તે જુઓ. ખાસ કરીને, તમારે પ્રારંભિક તબક્કામાં ક્યાંય જોવા જોઈએ, જે વસ્તુઓ આપણે પરિચિત છીએ, જેમ કે 3, 4, 5, વગેરે. પણ, તમે કેટલીક અવિવેકી આકૃતિ પણ જુઓ છો, જે 4, 0, 4. તો, આપણે કદાચ આ વધુ સારું કરવું જોઈએ, પણ તમે તે વિશે ચિંતા કરશો નહીં, પણ હું જે ભાર મૂકે છે તે એ છે કે, તમે વસ્તુઓની જેમ જુઓ, તમે 0, 77, 77 જુઓ, જે મૂર્ખ છે; ઉદાહરણ તરીકે, તમે 3, 4, 5 જુઓ છો. તેથી, આપણે 3, 4, 5, ક્યાંક - તેથી 3, 4, 5 જોયા. પણ, તમે પછીથી 4, 3, 5. હવે, એક દલીલ કરે છે કે, 3, 4, 5, અને 4, 3, 5, તે જ ત્રિપાદ છે. તો, આપણે આ ડુપ્લિકેટ કેવી રીતે દૂર કરીએ?

(સ્લાઇડ સમયનો સંદર્ભ લો: 12:46)

તેથી, આપણી પાસે એક પરિસ્થિતિ હોઈ શકે છે, જેમ કે `for loop` માં છે, જ્યાં પછીની લૂપ અગાઉની લૂપ પર આધાર રાખે છે; જો બાહ્ય લૂપ કહે છે, હું કેટલાક માટે, હું કંઈકથી કંઈક કરવા જાય છે, પછીના લૂપ કહી શકે છે કે, j હું થી શરૂ થાય છે, અને આગળ જાય છે. હમણાં પૂરતું, આપણે હવે અમારા પાચથાગોરસ ટ્રિપ્લસને ફરીથી લખી શકીએ છીએ કે, એક્સ રેન્જ 100 માં છે, પરંતુ વાય 0 થી શરૂ નથી કરતું; તે x આગળથી શરૂ થાય છે તેથી, y એ x કરતાં નાની નથી, અને z એ y કરતાં નાના નથી. તેથી, ઝે પણ એક્સ કરતા નાનું પણ નથી, કારણ કે y પોતે x કરતા નાની નથી, અને આ સંસ્કરણ ખરેખર ડુપ્લિકેટ્સને દૂર કરશે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 13:21)

અહીં પાચથાગોરસની અમારી પહેલી વ્યાખ્યા છે, જ્યાં અમારી પાસે એક્સ, વાય અને ઝેડ હતા. તેથી, હું શું કરું છું, હું પાછો જાઉં છું અને હું કહું છું કે, y એ 100 ની શ્રેણીમાં નથી, પરંતુ y એ x થી 100 રેન્જમાં છે, અને z શ્રેણી y થી 100 માં છે. અને હવે, તમે ખૂબ નાના સૂચિમાં અને ખાસ કરીને તમે જોશો કે, દરેક શ્રેણીમાં જે પેદા થાય છે, x વાય કરતાં ઓછું અથવા બરાબર z બરાબર છે; તમે 3, 4, 5 જેવી વસ્તુઓની એક કોપિ મેળવી શકો છો. તેથી, તમે 3, 4, 5 જુઓ છો, પરંતુ તમે 4, 3, 5 જોઈ શકતા નથી; 3, 4, 5 અહીં છે આગળું એક 5, 12, 13 છે; 4, 3, 5 નાબુદ થાય છે કી વસ્તુ છે, જનરેટર બાહ્ય જનરેટર પર આધારિત હોઈ શકે છે - આંતરિક જનરેટર બાહ્ય જનરેટર પર આધારિત હોઈ શકે છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 14:06)

આ સૂચિ ગૌણ સંકેત ખાસ કરીને યાદીઓ પ્રારંભ કરવા માટે ઉપયોગી છે, ઉદાહરણ તરીકે, મેટ્રિક્સ પ્રારંભ કરવા માટે, જ્યારે અમે ગણતરીઓ જેવા મેટ્રિક્સ કરી રહ્યા છીએ. ધારો કે, હું 4 થી 3 મેટ્રિક્સને તમામ ઝેરોઝમાં પ્રારંભ કરવા માંગુ છું. તેથી, 4 બાય 3 મેટ્રિક્સમાં 4 પંક્તિઓ અને 3 સ્તંભો છે, અને હું સંમેલનનો ઉપયોગ કરું છું કે, હું તેને પંક્તિ-મુજબ સંગ્રહિત કરું છું. તેથી, મને પ્રથમ પંક્તિ સંગ્રહિત કરવી પડશે તેથી, તે પ્રથમ પંક્તિ માટે 3 એન્ટ્રી હશે; પછી, બીજી પંક્તિ માટેની 3 એન્ટ્રીઓ અને તેથી વધુ. અહીં એક આરંભ છે, જે કહે છે, એલ બાહ્ય વસ્તુઓ માટે કંઈક કહે છે, આ દરેક પંક્તિ માટે છે તે દરેક પંક્તિ માટે કંઈક છે 4 પંક્તિઓ માટે 0, 1, 2, 3, હું કંઈક કરું છું; અને તે શું છે? હું તે કદના શુન્યોની સૂચિ

બનાવી છે. દરેક પંક્તિ j , 0 થી 3 માં, સ્તંભો 0, 1, 2, જે શૂન્ય છે. આ વાસ્તવમાં સાચી અનુક્રમને પેદા કરશે જે આપણે જોયું, તો જનરેટરને પ્રારંભ કરવાની જરૂર છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 15:18)

મેટ્રિક્સને પ્રારંભ કરવા માટે તે યાદી ગમ સંકેત છે. તેથી, તે કહે છે, દરેક જ શ્રેણી 4 માં, અધિકાર, અમે એક સૂચિ બનાવીએ છીએ અને તે સૂચિ મારી શ્રેણી 3 માં શૂન્ય છે, અને જો તમે આ કરો છો, અને તે જુઓ તો, યોગ્ય રીતે તે 3 શિરો છે, અને 3 શૂરો, અને 3 શૂન્ય, 4 વખત. આ ચાર પંક્તિઓ છે

(સ્લાઇડ સમયનો સંદર્ભ લો: 15:37)

એમ ધારો કે, તેના બદલે, અમે આ આરંભને 2 પગલાંમાં વિભાજિત કરી છે; આપણે પહેલા 3 ઝોરોની યાદી બનાવીએ છીએ જેને ઝોરોલિસ્ટ કહેવાય છે, જે શ્રેણી 3 માં શૂન્ય માટે કહે છે. આ 3 શૂન્યની યાદી બનાવે છે; અને પછી, અમે આ સૂચિ 4 વખત, ચાર પંક્તિઓમાં નકલ કરીએ છીએ. અમે કહીએ છીએ કે વાસ્તવિક મેટ્રિક્સ એલ પાસે ઝોરોલિસ્ટની 4 નકલો છે. હવે, અમે એક એન્ટ્રી પર જઈએ છીએ; કહેવું, અમે 1 પંક્તિ 1 માં એન્ટ્રી બદલીએ છીએ. ટોચ પરથી, તે ખરેખર બીજી પંક્તિ છે તે બીજી પંક્તિ, બીજો કોલમ છે, જો તમે સામાન્ય દ્રષ્ટિએ વિચારવું હોય તો તેથી, અમે સૂચિ 1 લો, જે બીજી સૂચિ છે. હવે, તમે જે અપેક્ષા કરો છો તે આનું આઉટપુટ છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 16:22)

ત્યાં આપણી પાસે શૂન્ય યાદીઓ છે, અને પછી, અમે કહીએ છીએ, 1 ચાર શ્રેણીમાં ઝોરોલિસ્ટની 4 નકલો છે, જેમા શ્રેણી 4 માં છે. તેથી, ઉપરી સપાટી પર, 1 બરાબર એ જ દેખાય છે. હવે, આપણે કહીએ છીએ એલ -1 1 બરાબર 7 છે, અને જો તમે એલ પર જોશો, તો અમે શોધીશું કે અમારી પાસે એક 7 નથી, પરંતુ 7 ની 4 નકલો. આ દેખીતી એવી એવી વસ્તુ છે જે અમે અપેક્ષા રાખી ન હતી. (0: 7, 0], [0, 7, 0], [0, 7, 0], [0, 7, 7, 0] [0, 7, 0], [0, 7, 0], [0, 7, 0] 0] અને આ શા માટે થઈ રહ્યું છે; સારું, તે એક જ ઝોરોલિસ્ટ બનાવીને, અને ત્યારબાદ 4 નકલો બનાવીને, અમે આ જ યાદી સાથે અસરકારક રીતે 4 નામો બનાવી દીધા છે. તેથી, આપણે તેને એલ 0 અથવા એલ 1 કે એલ 2 અથવા એલ 3 દ્વારા એક્સેસ કરીએ છીએ, તેમાંથી તમામ 4 એ એક જ ઝોરોલિસ્ટ તરફ સંકેત કરે છે. તેથી, તે અપડેટ્સમાંથી કોઈ એક ખરેખર તમામ 4 યાદીઓને અપડેટ કરશે. જો તમે 2 પરિમાણીય મેટ્રિક્સ બનાવવું અને તેને પ્રારંભ કરવા માંગો છો, તો નિશ્ચિત કરો કે તમે તેને નેસ્ટ કરેલ રેંજનો ઉપયોગ કરીને એક શોટમાં પ્રારંભ કરો, અને આની જેમ 2 નકલોમાં નહીં, કારણ કે આ 2 નકલો અજાણતામાં 2 પંક્તિઓને સમાન નકલોમાં જોડશે વસ્તુ, અને એક પંક્તિના અપડેટ્સ પણ બીજી હરોળને અપડેટ કરશે

(સ્લાઇડ સમયનો સંદર્ભ લો: 17:45)

સારાંશ માટે, મેપ અને ફિલ્ટર યાદીઓમાં હેરફેર કરવા માટે ખૂબ ઉપયોગી કાર્યો છે, અને પાયથોન પ્રોગ્રામિંગના આધારે, અન્ય ઘણી પ્રોગ્રામિંગ લેન્ગ્વેજની જેમ, સૂચિની સૂચિતાર્થ સંકેત આપે છે, નકશા અને ફિલ્ટરને સંયોજિત કરવા માટે. અને, સૂચિની સમજણ માટે આપણે જે ઉપયોગો જોયાં છે તેમાંથી દ્વિદિશ અથવા બહુપરીમાણીય યાદીઓને કેટલાક મુળભુત મૂલ્યોમાં યોગ્ય રીતે આરંભ કરવો છે.

પ્રોગ્રામિંગ ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રો. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
મોડ્યુલ - 05 લેક્ચર - 01 અપેક્ષા રાખવી હેન્ડલિંગ

(સ્લાઇડ ટાઇ નો ઉલ્લેખ કરો: 00:03)

ચાલો જોઈએ કે જ્યારે વસ્તુઓ ખોટી થાય ત્યારે શું કરવું અમારા કાર્યક્રમો સાથે હવે ઘણી બધી વસ્તુઓ છે જે ખોટી જઈ શકે છે. દાખલા તરીકે, આપણી પાસે એક્સ (x) વડે ઝેડ (expression) વડે સમીકરણ હોઈ શકે છે, અને z પાસે મૂલ્ય શૂન્ય છે. તેથી, આ અભિવ્યક્તિ મૂલ્યની ગણતરી કરી શકાતી નથી, અથવા આપણે કોઈ શબ્દને સ્ટ્રિન્જમાંથી પૂર્ણાંકમાં રૂપાંતરિત કરવાનો પ્રયાસ કરી રહ્યા હોઈ શકે છે જ્યાં સ્ટ્રિન્જ પૂર્ણાંકની માન્ય રજૂઆત નથી. અમે એવા નામનો ઉપયોગ કરીને પણ અભિવ્યક્તિની ગણતરી કરવાનો પ્રયાસ કરી શકીએ છીએ, જેના મૂલ્યની વ્યાખ્યા કરવામાં આવી નથી, અથવા આપણે સૂચિમાં પોઝિશન ઇન્ડેક્સ કરવાનો પ્રયાસ કરી શકીએ છીએ જે અસ્તિત્વમાં નથી. જેમ જેમ આપણે આગળ વધીએ છીએ તેમ આપણે ડિસ્ક પરના ફાઇલોમાંથી કેવી રીતે વાંચવું અને લખવું તે જોઈશું. તો, આપણે કોઈ ફાઇલમાંથી વાંચવાનો પ્રયાસ કરી રહ્યા હોઈએ, પરંતુ કદાચ આવી કોઈ ફાઇલ નથી અથવા આપણે કોઈ ફાઇલમાં લખવાનો પ્રયત્ન કરી રહ્યા હોઈએ, પરંતુ ડિસ્ક ખરેખર સંપૂર્ણ છે. તેથી, ત્યાં ઘણી પરિસ્થિતિઓ છે જેમાં અમારા પ્રોગ્રામ ચાલી રહ્યો છે ત્યારે અમને ભૂલ આવી શકે છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 00:59)

આમાંથી કેટલીક ભૂલો અપેક્ષિત થઈ શકે છે, જ્યારે અન્ય અનપેક્ષિત છે. જો આપણે કોઈ ભૂલની ધારણા કરી શકીએ તો અમે તેને ભૂલ તરીકે નહિ, પણ એક અપવાદ તરીકે વિચારીશું. તેથી, શબ્દ અસાધારણ વિશે વિચારો. અમે એક સામાન્ય પરિસ્થિતિ અનુભવીએ છીએ, જે રીતે અમે અમારા પ્રોગ્રામને ચલાવવા માગીએ છીએ અને ક્યારેક ક્યારેક આપણે એક અસાધારણ પરિસ્થિતિ અનુભવી શકીએ છીએ, જ્યાં કંઈક ખોટું થાય છે અને આપણે શું કરવા માગીએ છીએ તે આ અસાધારણ પરિસ્થિતિ સાથે કેવી રીતે કામ કરવું અને કેવી રીતે યોજનાઓ પૂરી પાડે છે તેને અપવાદ હેન્ડલિંગ કહેવામાં આવે છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 01:36)

તેથી, અપવાદ હેન્ડલિંગ પૂછે છે, જ્યારે કંઈક ખોટું થાય છે અમે કેવી રીતે સુધારાત્મક ક્રિયા પ્રદાન કરીએ છીએ. હવે સુધારાત્મક પ્રકારનો પ્રકાર તે કયા પ્રકારની ભૂલ પર આધાર રાખે છે. દાખલા તરીકે જો આપણે કોઈ ફાઇલ વાંચવાનો પ્રયાસ કરીએ છીએ અને ફાઇલ અસ્તિત્વમાં નથી તો અમે યુઝરને ફાઇલ નામ લખવા માટે પૂછ્યું છે. તો, તમે મેસેજ પ્રદર્શિત કરી શકો છો અને યુઝરને ફાઇલ નામ ફરીથી ટાઇપ કરવા માટે કહી શકો છો, એવું કહી રહ્યું છે કે ફાઇલ અસ્તિત્વમાં નથી. પ્રશ્નની બીજી બાજુ જો સૂચિ બાહ્ય લિંક્સથી અનુક્રમિત કરવામાં આવે તો આપણા પ્રોગ્રામમાં કદાચ એક ભૂલ છે, અને આપણે આ મૂલ્યને ઇન્ડેક્સના મૂલ્યને છાપવા માંગીએ છીએ કે અમારા પ્રોગ્રામમાં શું ખોટું થઈ રહ્યું છે તે તપાસવા અને તેનું નિદાન કરવું. કેટલીકવાર ભૂલ નિયંત્રણમાં કદાચ અમારી ભૂલ પ્રક્ષેપણ પ્રોગ્રામ ડિબગિંગ થઈ શકે છે. આ માટે જે જરૂરી છે તે આ પ્રોગ્રામની અંદર આ ભૂલોને પકડવાની એક રીત છે કારણ કે તે પ્રોગ્રામની હત્યા વગર ચાલી રહી છે. તેથી, જેમ આપણે જોયું કે જ્યારે અમે ભૂલો જોયા છે, જ્યારે કોઈ ભૂલ થાય છે ત્યારે અમે દુભાષિયાનો ઉપયોગ કરી રહ્યા છીએ અને અમે આ રીતે તે ફાંસું નહીં કરીએ તો

પ્રોગ્રામ ખરેખર રદબાતલ અને બહાર નીકળશે. તેથી, અમે પ્રોગ્રામને રદબાતલ વિના ભૂલને પકડી અને તેની સાથે વ્યવહાર કરવાની રીત માંગીએ છીએ.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 02:53)

હવે, ઘણી જુદી જુદી પ્રકારની ભૂલો છે અને તેમાંના કેટલાક અમે જોયા છે, પરંતુ અમે આમાંની પેટાપ્રદેશને કદાચ ઉદાહરણ તરીકે જોયો નથી, જ્યારે આપણે પાયથોન ચલાવીએ છીએ અને આપણે જે કંઈક ખોટું લખીએ છીએ, તો પછી આપણે કંઈક સિન્ટેક્સ ભૂલ અને સંદેશો કે જે પાયથોન આપણને આપે છે તે સિન્ટેક્સ ભૂલ અમાન્ય સિન્ટેક્સ છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 03:19)

ઉદાહરણ તરીકે, ધારણા છે કે આપણે એક યાદી બનાવવાનો પ્રયાસ કરીએ છીએ અને ભૂલથી આપણે અલ્પવિરામને બદલે અર્ધવિરામનો ઉપયોગ કરીએ છીએ. તે પછી તરત જ તે અર્ધવિરામ પર પેથન નિર્દેશ કરે છે અને કહે છે કે આ વાક્યરચના ભૂલ છે તે અમાન્ય પિથન સિન્ટેક્સ છે. અલબત્ત, જો અમારી પાસે અમાન્ય સિન્ટેક્સ છે; તેનો અર્થ એ કે, પ્રોગ્રામ હજી ચલાવવાનું નથી રહ્યું અને આપણે ત્યાં વધારે કરી શકતા નથી. તેથી, જે વાસ્તવમાં રુચિ ધરાવે છે તે તે છે જે માન્ય પ્રોગ્રામમાં થાય છે. આ પ્રોગ્રામ સિન્ટેક્સની રીતે સાચી છે, તે કંઈક છે કે જે પાયથોન ઇન્ટરપ્રીટર એક્ઝિક્યુટ કરી શકે છે, પરંતુ જ્યારે કોડ એક્ઝિક્યુટ થઈ રહ્યો છે ત્યારે કોઈ ભૂલ થાય છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 04:01)

આ છે જેને રન ટાઇમ એરરો કહેવામાં આવે છે, આ ભૂલ છે જે પ્રોગ્રામ ચાલી રહી હોય ત્યારે થાય છે અને અહીં ફરીથી અમે આ ભૂલો જોયાં છે અને તે કેટલીક ડાયગ્નોસ્ટિક માહિતી સાથે આવે છે. દાખલા તરીકે, જો આપણે એવા નામનો ઉપયોગ કરીએ છીએ જેના મૂલ્યનું અવ્યાખ્યાયિત છે, તો અમને પાયથોન તરફથી સંદેશ મળે છે કે નામ વ્યાખ્યાયિત નથી અને અમને લીટીની શરૂઆતમાં એક કોડ પણ મળે છે જે કહીને આ નામ ભૂલ છે. આ પાયથોન ને કહેવાની રીત છે કે તે કંઈ પ્રકારની ભૂલ છે, જો અમારી પાસે અંકગણિત અભિવ્યક્તિ છે, જ્યાં આપણે મૂલ્ય 0 દ્વારા ભાગાકાર કરીએ છીએ, તો પછી આપણે કંઈક શૂન્ય વિભાગની ભૂલ અને છેલ્લે, જો તમે અનુક્રમણિકા તેની શ્રેણીની બહાર સૂચિબદ્ધ કરીએ તો આપણે કંઈક ઇન્ડેક્સ ભૂલ કહેવાય છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 04:54)

ચાલો આ બધી ભૂલ જુઓ ફક્ત ખાતરી કરો કે અમે સમજીએ છીએ ધારો કે, આપણે y બરાબર 5 ગુણ્યા x છે અને આપણે x માટે કંઈપણ વ્યાખ્યાયિત નથી, તો તે આપણને ઇન્ડેક્સની ભૂલ નામની ભૂલ આપે છે અને તે સ્પષ્ટ રીતે કહે છે કે, નામ x વ્યાખ્યાયિત નથી. બીજી બાજુ, જો આપણે કહીએ કે 5 ભાગ્યા 0 બરાબર છે તો આપણને 0 ડિવિઝનની ભૂલ મળી શકે છે અને સાથે સાથે સંદેશ ડિવિઝન 0 વડે અને છેવટે, જો તમારી પાસે સૂચિ 1, 2 કહે છે અને પછી આપણે પોઝિશન ત્રણ જોઈએ પછી તે કહેશે કે આ સૂચિમાં કોઈ સ્થાન નથી. તેથી, આ ઇન્ડેક્સની ભૂલ છે તેથી, આ ત્રણેય ઉદાહરણો છે જેમાં પાયથોન દૂભાષક અમને કહે છે અને નોંધ્યું છે કે શરૂઆતના ઇન્ડેક્સ ભૂલ નામની ભૂલ શૂન્ય શ્રેણીની ભૂલમાં તે પછી ભૂલનું નામ છે અને તે પછી નિદાન વિશ્લેષણ છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 05:45)

ચાલો આપણે પહેલા કેટલાક પરિભાષા પર પતાવટ કરીએ. તેથી, સામાન્ય રીતે ભૂલને સંકેત આપવાની કાર્યવાહી અપવાદ વધારવામાં આવે છે. તેથી, જ્યારે પાઠ્યશોન દુભાષિયો એક ભૂલ શોધે છે ત્યારે અમને આ ભૂલ વિશેની માહિતી આપે છે અને જેમ આપણે જોયું તેમ તે બે ભાગોમાં આવે છે, તો ભૂલનું પ્રકાર છે કે તે કઈ પ્રકારની ભૂલ છે તેથી, તે નામ ભૂલ અથવા ઇન્ડેક્સ ભૂલ અથવા શૂન્ય વિભાગ ભૂલ છે અને બીજું, કેટલીક ડાયગ્નોસ્ટિક માહિતી અમને જણાવે છે કે આ ભૂલ ક્યાં થાય છે. તેથી, તે ફક્ત એટલું જ કહેવું પૂરતું નથી કે કેટલાક મૂલ્ય નિર્ધારિત કરવામાં આવતી નથી, તે અમને ખાસ જણાવે છે કે એક્સ નામ વ્યાખ્યાયિત નથી. આ અમને કેટલાક સંકેત આપે છે કે જ્યાં ભૂલ હવે હોઈ શકે છે, જ્યારે ક્ષતિને પાઠ્યશોન દ્વારા સંકેત આપવામાં આવે છે કે અમે શું કરવા ઇચ્છીએ છીએ, તો અમારા પ્રોગ્રામની અંદર જ તે સાચું છે તેથી, અમે ભૂલ પ્રકાર પર આધારિત સુધારાત્મક પગલાંની પૂર્વાનુમાન અને લેવા માંગીએ છીએ. તેથી, અમે દરેક એરર પ્રકાર માટે સમાન પ્રકારની ક્રિયા લેવાની ઇચ્છા રાખી શકીએ નહીં. તેથી જ તે જાણવા માટે મહત્વનું છે કે શું તે નામ ભૂલ છે અથવા ઇન્ડેક્સની ભૂલ અથવા બીજું કંઈક છે. અને ભૂલ શું છે તેના પર આધાર રાખીને, અમે તે પ્રકારની ભૂલ માટે યોગ્ય પગલાં લઈ શકીએ છીએ અને છેલ્લે, જો અમને કોઈ ભૂલ અથવા અપવાદ મળે કે જેનો અમે સ્પષ્ટ રીતે નિયંત્રિત ન કર્યો હોય તો અજમાયશી દુભાષિયો પાસે કોઈ વિકલ્પ નથી, પરંતુ પ્રોગ્રામનો બંધ કરવાની છે. તેથી, જો અપવાદને નિયંત્રિત ન કરતા હોય તો અપવાદ બંધ નહીં થાય તો અમલ બંધ થઈ જાય છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 07:15)

આ એક નવા પ્રકારનાં બ્લોકનો ઉપયોગ કરીને કરવામાં આવે છે જે આપણે પ્રયાસ કરતા પહેલા જોયો નથી. તેથી, અમારી પાસે શું છે તે બ્લોક પ્રયાસ કરો. તેથી, જ્યારે આપણી પાસે કોડ છે, અહીં અમે અપેક્ષા રાખીએ છીએ કે કેટલીક ભૂલ હોઈ શકે છે અમે તેને એક પ્રયાસમાં મૂકીએ છીએ. આ કોડનો અમારો સામાન્ય બ્લોક છે જ્યાં અમે ધારણા કરી શકીએ કે કંઈક ખોટું થઈ શકે છે અને હવે અમે બધી વસ્તુઓ માટે આકસ્મિકતા પૂરી પાડીએ છીએ જે ભૂલના પ્રકાર પર આધાર રાખીને ખોટી જઈ શકે છે અને આ સિવાય સ્ટેટમેન્ટ સિવાય આ પ્રદાન કરવામાં આવે છે. તે કહે છે આ પ્રયાસ કરો અને જો કંઈક ખોટું થાય, તો પછી બીજા પછી એક સિવાય યોગ્ય પર જાઓ પ્રથમ એક કહે છે કે ઇન્ડેક્સની ભૂલ થાય ત્યારે શું થાય છે. તેથી, આ તે કોડ છે જે થાય છે જો બીજી બાજુ ઇન્ડેક્સની ભૂલ થાય છે તો કદાચ મને નામની ભૂલ મળી શકે છે અથવા બન્ને માટે હું એક કી ભૂલ કરી શકું છું, તેથી આ બ્લોક સિવાયની બીજી બાજુ છે. તેથી, તમારી પાસે બ્લોક્સ સિવાય ઘણા બધા હોઈ શકે છે કારણ કે તમારી પાસે ભૂલોના પ્રકારો છે જે તમે ભૂલો માટે અપેક્ષા રાખતા હોવ તે દરેક પ્રકારના ભૂલને નિયંત્રિત કરવા માટે ફરજિયાત નથી, માત્ર તે જ તમે જે અપેક્ષા રાખશો અને અલબત્ત, હવે તમે સામાન્ય રીતે કંઈક કરવા માંગો છો બધી ભૂલો જે તમે અપેક્ષા રાખતા નથી તેથી, તમે બ્લોક સિવાય શુદ્ધ હોઈ શકો છો. તેથી બ્લોક સિવાયનો નવો પ્રકારનો પ્રકાર કે જેમાં તમે ભૂલનો પ્રકાર નિર્દિષ્ટ કર્યો નથી અને ડિફોલ્ટ રૂપે ડિફોલ્ટ સિવાય અન્ય તમામ અપવાદોને પકડી શકે છે. મહત્વની વસ્તુઓ રાખો કે આ ક્રમ માં થાય છે જો મારી પાસે ઉદાહરણ માટે ત્રણ ભૂલો છે, જો મારી પાસે ઇન્ડેક્સની ભૂલ અને નામની ભૂલ અને શૂન્ય વિભાગની ભૂલ છે, તો શું થશે - એ છે કે, તે સૌ પ્રથમ અહીં આવશે અને શોધી કાઢશે કે ઇન્ડેક્સની ભૂલ છે આ કોડ ચલાવશે. નામ ભૂલ કોડ બીજી બાજુ એક્ઝેક્યુટ કરશે નહીં, જો મારી પાસે માત્ર એક નામની ભૂલ હતી અને જો મારી પાસે શૂન્ય ડિવિઝનની ભૂલ હતી, ઉદાહરણ તરીકે, ત્યારબાદ નામ ભૂલ છે, પ્રથમ તો તે અહીં આવશે અને તે ત્યાં મળશે. કોઈ ઇન્ડેક્સની ભૂલ નથી તો પછી અહીં આવશે અને કદેશો ત્યાં નામની ભૂલ છે અને આ કોડ એક્ઝેક્યુટ કરશે. શૂન્ય ડિવિઝનની ભૂલને સ્પષ્ટપણે નિયંત્રિત કરવામાં આવશે નહીં, પ્રોગ્રામ રદબાતલ થશે નહીં, પરંતુ શૂન્ય ડિવિઝન ભૂલ માટે કોઈ કોડ એક્ઝેક્યુટ કરવામાં આવશે નહીં; તે એ નથી કે તે આમાંના દરેકને અજમાવે છે જે ભૂલથી મેળ ખાય છે અને બાકીનાને અવગણે તે સિવાય તે જે રીતે પ્રયાસ કરશે તે તેથી, છેવટે, જો મારી પાસે આ ચોક્કસ ઉદાહરણમાં માત્ર એક શૂન્ય ડિવિઝન ભૂલ હતી, કારણ કે તે ઇન્ડેક્સની ભૂલ નથી અને તે કોઈ નામની ભૂલ નથી, તો તે આમાંથી વળાંકમાં જવાનો પ્રયાસ કરશે જે અહીં આવશે તે શોધો કે આ નથી ભૂલનો એક પ્રકાર તે કોઈ પ્રકારની ભૂલ નથી અને તે ડિફોલ્ટ સિવાય ડિફોલ્ટ પર જશે અને અન્ય તમામ અપવાદોને પકડાશે.

છેલ્લે, પિથન અમને બીજું ખૂબ ઉપયોગી વૈકલ્પિક ખંડ આપે છે. તેથી, આ બીજું એક જ આત્મામાં છે કારણ કે 'ફોર' અથવા 'એ' સાથે સંકળાયેલું બીજું યાદ રાખવું કે જે તે માટે તૂટી ન જાય તે માટે સામાન્ય રીતે તે બંધ થઈ જાય છે અને જો કોઈ વિરામ હોય તો તે છોડવામાં આવે છે એ જ રીતે, જો પ્રયાસ સામાન્ય રીતે ચલાવવામાં આવે તો ત્યાં કોઈ ભૂલ નથી જે પછી મળી આવે તો બીજું કોઈ અન્યને ચલાવશે નહીં તો બીજું જ ખોટું છે. તેથી, અમારી પાસે એક બીજું બ્લોક છે જે ચલાવવામાં આવશે જો કોઈ ભૂલ વિના સામાન્ય રીતે ટર્મિનેટ થાય તો આ એ એકંદર માળખું છે કે અમે અપવાદોને કેવી રીતે હેન્ડલ કરીએ છીએ જે કોડને અમે એક પ્રયાસ બ્લોકની અંદર મુકીએ છીએ પછી અમારી પાસે એક બ્લોક સિવાયના અનુક્રમ છે જે અપવાદોના વિવિધ પ્રકારોને પકડી રાખે છે, જેમાં આપણે એક શ્રેણીમાં અપવાદ દ્વારા એક પ્રકારનો અપવાદ મેળવી શકીએ છીએ. અપવાદોના તૂટી, અપવાદ સિવાય અન્ય કોઈ અપવાદોને પકડી ન શકાય તે સિવાય કોઈ મૂળભૂત નામ સિવાય કોઈ ડિફોલ્ટ હોઈ શકે છે અને આખરે, અમારી પાસે બીજું એક છે જેનો પ્રયાસ સામાન્ય રીતે સમાપ્ત થઈ જાય છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 10:58)

હવે, જ્યારે અમે સામાન્ય રીતે ભૂલોનો સામનો કરવા માટે અપવાદ હેન્ડલિંગનો ઉપયોગ કરીએ છીએ જે અમે અપેક્ષા રાખતા નથી. પ્રોગ્રામિંગની શૈલી બદલવા માટે આપણે તેનો ઉપયોગ કરી શકીએ છીએ. તેથી, ચાલો આપણે એક વિશિષ્ટ ઉદાહરણ જોઈએ. અમે તાજેતરમાં જોયું કે આપણે પાયથોન માં શબ્દકોશો વાપરી શકીએ છીએ. તેથી, કીઓ સાથે શબ્દકોશો સંલગ્ન મૂલ્યો અહીં આપણે બે કવિઓ ધવન અને કોહલી અને દરેક કી સાથે છે જે નામ છે, અમારી પાસે સ્કોર્સની યાદી છે. તેથી, આ સ્કોર એ શબ્દકોશ છે કે જેની કીઓ શબ્દમાળાઓ છે અને જેની કિંમતો નંબરોની સૂચિ છે. હવે માની લો કે આપણે આમાં સ્કોર ઉમેરવા માંગીએ છીએ. કોઈ ખાસ બેટસમેન સાથેનો સ્કોર સંકળાયેલ છે તેથી, બેટસમેન માટે અમારી પાસે સ્કોર છે અને અમે આ શબ્દકોશને અપડેટ કરવા માંગીએ છીએ. હવે બે પરિસ્થિતિઓ છે, એક એ છે કે આપણી પાસે પહેલેથી જ શબ્દકોશમાં b છે જે કોઈ પણ કિસ્સામાં આપણે હાલની લિસ્ટમાં બીજા સ્થાને ઉમેરવા માંગીએ છીએ, બીજી સ્થિતિ એ છે કે આ એક નવું બેટસમેન છે, ત્યાં કોઈ બેટસમેન નથી. આ કિસ્સામાં આપણે અધિકારીઓની સૂચિ ધરાવતી યાદીના બરાબર બી ની સ્કોર્સ બનાવીને એક કી બનાવવાની જરૂર છે. આપણી પાસે ઓપરેશનના બે વૈકલ્પિક મોડ છે, તે કોઈ અવિશ્વાસુ કી સાથે જોડવાનો પ્રયાસ કરવામાં ભૂલ છે, પરંતુ જો કોઈ અસ્તિત્વમાંની યાદી છે અમે તે ફરીથી સોંપણી દ્વારા ગુમાવી નથી માંગતા. તેથી, અમે તેને ઉમેરવા માંગીએ છીએ. તેથી, અમે આ બે કેસોને અલગ કરવા માંગીએ છીએ.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 12:21)

આને આપણે પહેલેથી જોયું છે તે ઉપયોગ કરવા માટે એક પ્રમાણભૂત રીત છે, તે ચકાસવા માટે સ્ટેટમેન્ટમાં આદેશનો ઉપયોગ કરવો કે નહીં તે ચકાસવા માટે કે શું કિંમત b એ સ્કોર્સમાં કી તરીકે થાય છે. તેથી, આપણે કહીએ છીએ કે b એ સ્કોર્સ છે, ડોટ કીઓ છે જો આપણી પાસે વર્તમાન કી તરીકે બી હોય તો આપણે ગુણ ઉમેરવો જોઈએ નહિ તો આપણે નવી એન્ટ્રી બનાવીશું. તેથી, આ સારું છે, હવે આપણે નીચે પ્રમાણે અપવાદ હેન્ડલિંગનો ઉપયોગ કરીને આ કરી શકીએ છીએ; અમે તેને ઉમેરવાનો પ્રયાસ કરીએ છીએ અમે ડિફોલ્ટ દ્વારા ધારે છે કે બક્સિસ બટલર બી પહેલેથી જ આ શબ્દકોષમાં કી તરીકે અસ્તિત્વ ધરાવે છે અને અમે સ્કોર્સ બી ડોટ એપેન્ડ ઓનો પ્રયાસ કરીએ છીએ. જો બી ન હોય તો શું થશે? વેલ પાયથોન એ કહીને ભૂલ સંકેત આપશે કે આ એક અમાન્ય કી છે અને તે કી ભૂલ કહેવાય છે તેથી, અમે પછી સ્ટેટમેન્ટ સિવાય આ પર પાછા જઈ શકીએ છીએ, ઓહ કહીએ છીએ કે કી ભૂલ છે જ્યારે હું એમને સ્કોર્સના સ્કોર્સમાં ઉમેરવાનો પ્રયત્ન કરું છું, પછી b નો એન્ટ્રી સ્કોર્સ બનાવવા તે બરાબર છે. તેથી, આ માત્ર એક અલગ શૈલી છે, તે એમ નથી કહેતા કે એક બીજા કરતાં વધુ સારી છે, પરંતુ તે ફક્ત એટલું ભાર મૂકે છે કે એક વાર આપણી નિયંત્રણ હેઠળ અપવાદ નિયંત્રણ હોય તો આપણે શું કરીએ

છીએ તેમાંથી અલગ વસ્તુઓ કરી શકીએ છીએ અને કેટલીકવાર આ વધુ સ્પષ્ટ થઈ શકે છે તે શૈલીની બાબત છે જે તમને વધુ ડાબે અથવા જમણે હોઈ શકે છે, પરંતુ બન્ને પાયથોન કોડના માન્ય ટુકડા છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 13:48)

ચાલો જોઈએ કે જ્યારે ભૂલ થાય ત્યારે ખરેખર શું થાય છે. તેથી, ધારીએ કે આપણે કંઈક એક્ઝિક્યુટ કરવાનું શરૂ કરીએ છીએ અને અમારી પાસે ફંક્શન એફ માટે ફંક્શન કોલ છે, પેરામીટર વાય અને ઝે સાથે આ ચાલશે અને કાર્ય માટેની વ્યાખ્યા અને કદાચ વ્યાખ્યામાં કદાચ તેથી, આ કોલને આ કોડ ચલાવવામાં પરિણમે છે અને આ વ્યાખ્યામાં તે અન્ય ગ્રુપ કહેવાય છે જેને *g* આ બદલાશે અમને નવા વ્યાખ્યામાં પરિવર્તન લાવશે માફ કરશો, તે આ જ લાઇન પર હોવી જોઈએ અને આમાં ફરી એક ફંક્શન હાર હોઈ શકે છે અને છેલ્લે, જ્યારે આપણે સમસ્યા પર જઈએ છીએ ત્યારે કદાચ આ પર જઈએ છીએ. હમણાંમાં અંદર કદાચ ઇન્ડેક્સની ભૂલ છે અને જ્યાં અમે આ સૂચિને ઉદાહરણ તરીકે ઉપયોગમાં લીધી છે, *h* માં અમે તેને એક પ્રયાસ બ્લોક પર મૂક્યો નથી અને તેથી, એરર હેન્ડલ કરવામાં આવી નથી. તેથી, અમે શું કહ્યું છે કે જ્યારે કાર્યક્રમ ભૂલને નિયંત્રિત કરવામાં ન આવે ત્યારે ભૂલ થાય છે, પરંતુ કાર્યક્રમ સીધી રીતે બંધ ન થાય; આ ફંક્શન રદબાતલ થઈ જશે અને તે ભૂલને તે ગમે તેટલું પાછું ફેરવશે. તો, અહીં શું થશે તે એ છે કે આ ઇન્ડેક્સની ભૂલ બિંદુ પર પાછા આવશે, જ્યાં *g* એ *h* માં બોલાવવામાં આવશે. હવે, એવું છે કે *g* એ ઇન્ડેક્સ ભૂલ બોલાવવાથી *h* દ્વારા ઇન્ડેક્સ ભૂલ પેદા થઈ છે. તેથી, ઇન્ડેક્સની ભૂલ વાસ્તવમાં હવે જોમાં છે કારણ કે *H* એ ભૂલ સાથે કંઈપણ કર્યું ન હતું, અમે ભૂલ સાથે તેને પાછું પસાર કર્યું છે. હવે, અમારી પાસે બે વિકલ્પો છે, જેમાં *G* એ એક પ્રયાસ બ્લોક છે, પરંતુ જો *g* એ એક પ્રયાસ બ્લોક ન હોય તો આ ભૂલ રદ કરવા માટેનું કારણ બને છે. તેથી, આગળ શું થશે એ છે કે જો જો તેને હેન્ડલ કરતું નથી તો આ પાછું જ્યાં જો માં બોલાવવામાં આવ્યું હતું અને તે જ રીતે જો હવે એફ તેને હેન્ડલ કરતું નથી, તો તે પાછું જ્યાં મુખ્ય કાર્યક્રમમાં કહેવાયું હતું તે પાછું જશે. તેથી, અમે ભૂલને પાછો ફાળતા ફંક્શન કોલ્સની શ્રેણીમાં પાછા જવાનું ચાલુ રાખીએ છીએ. જો આપણે તેને કાર્યમાં નહીં રાખીએ છીએ જ્યાં અમે હમણાં છીએ, તો એરર પાછો જાય છે, આ ફંક્શન બંધ થઈ જાય છે તે પાછો જાય છે અને છેવટે, જ્યારે તે મુખ્ય શ્રેડને પ્રથમ પિથન કોડના પ્રથમ કાર્ય પર પહોંચે છે, ત્યારે આપણે ત્યાં કાર્યરત છીએ. જો આપણે તેને નિયંત્રિત ન કરીએ તો નિશ્ચિતપણે એકંદરે પાયથોન પ્રોગ્રામ બંધ થઈ જશે. તેથી, એવું નથી કે પ્રથમવાર આપણે કોઈ ભૂલ શોધી કાઢીએ જે તેને નિયંત્રિત કરવામાં ન આવે તો તે ફક્ત રદબાતલ થઈ જશે, જ્યાંથી તેને કોલની ક્રમાંકમાંથી અને સમગ્ર શ્રેણીમાં કહેવામાં આવતું હતું તે ક્રમમાં અમે તેને કોઈપણ સમયે પકડી શકીએ છીએ. તેથી, આપણે તે ક્ષણને ભૂલથી પકડી ન રાખવી જોઈએ કે જ્યાં તેનું સંચાલન થાય છે, તે આપણને તે બોલાવે છે કે જે અમને બોલાવે છે.

(સ્લાઈડ સમયનો સંદર્ભ લો: 16:23)

અપવાદ હેન્ડલિંગનો સારાંશ અમને રનટાઈમ ભૂલોને પ્રભાવિત કરવા માટે પરવાનગી આપે છે. તેથી, પાયથોન ને ફ્લેગ કરતી વખતે પાયથોન આપણને પ્રકારનો ભૂલ અને કેટલાક ડાયગ્નોસ્ટિક માહિતી આપે છે. પ્રયાસ અને બ્લોક સિવાય, અમે ભૂલ પ્રકાર તપાસો અને પ્રકાર પર આધારિત યોગ્ય પગલાં લઈ શકે છે. અમે એ પણ જોયું કે તે શબ્દકોશમાં મૂલ્ય શામેલ કરે છે કે આપણે પ્રોગ્રામિંગની નવી શૈલીઓ વિકસાવવા માટે અપવાદ હેન્ડલિંગનો ફાયદો ઉઠાવી શકીએ છીએ અને છેવટે, આપણે શું જોશું, આપણે આગળ વધીએ છીએ અને અમે ઇનપુટ આઉટપુટ અને ફાઇલ્સ સાથે વ્યવહાર કરવાનું શરૂ કરીએ છીએ. તેના બદલે વધુ સામાન્ય છે કારણ કે જેમ પહેલાં આપણે જોયું તે ઉદાહરણમાંની કોઈ એક ફાઇલ મળી ન હતી કે ડિસ્ક ભરાઈ જાય છે. ઇનપુટ અને આઉટપુટમાં સ્વાભાવિક રીતે પ્રોગ્રામની બહાર બહારની વસ્તુઓ સાથે ઘણી ક્રિયાપ્રતિક્રિયા થાય છે અને તેથી તે ઘણી ભૂલો માટે સંવેદનશીલ છે અને તેથી, અમારી યુક્તિની બેગની અંદર આ પદ્ધતિ સક્ષમ છે.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગારિધમ્સ ઇન પાયથોન
પ્રો. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
અઠવાડિયું - 05 લેક્ચર - 02

સ્ટાન્ડર્ડ ઇનપુટ અને આઉટપુટ હવે સુધી, તમને તમારા કાર્યમાં લખવાનું કહેવામાં આવ્યું છે તે તમામ પ્રોગ્રામ્સ વાસ્તવમાં ફક્ત કાર્યો જ છે

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 00:02)

આ કાર્ય છે, જેને પાયથોન કોડના અન્ય ટુકડામાંથી કહેવામાં આવે છે અને તેમને મૂલ્ય પરત કરે છે. હવે, જ્યારે તમે એકલા પાયથોન પ્રોગ્રામ ધરાવો છો, તો તે ઇનપુટ્સ મેળવવા અને આઉટપુટ બનાવવા માટે વપરાશકર્તા સાથે ક્રિયાપ્રતિક્રિયા કરવી આવશ્યક છે. ચાલો જોઈએ, પાયથોન તેના પર્યાવરણ સાથે કેવી રીતે ક્રિયાપ્રતિક્રિયા કરે છે. પર્યાવરણ સાથે ક્રિયાપ્રતિક્રિયા કરવાની સૌથી મૂળભૂત રીત છે કીબોર્ડમાંથી ઇનપુટ લેવાનું અને સ્ક્રીન પર આઉટપુટ પ્રદર્શિત કરવું.

(સ્લાઇડ સમયનો સંદર્ભ લો: 00:54)

પરંપરાગત રીતે, આ મોડ્યુલને પ્રમાણભૂત ઇનપુટ અને સ્ટાન્ડર્ડ આઉટપુટ કહેવામાં આવે છે. તેથી, પ્રમાણભૂત ઇનપુટનો અર્થ થાય છે, કીબોર્ડ અથવા કોઈપણ પ્રમાણભૂત ઇનપુટ ઉપકરણથી ઇનપુટ લો અને તે અને પ્રમાણભૂત આઉટપુટનો અર્થ એ થાય કે સ્ક્રીન પર સીધા જ આઉટપુટ પ્રદર્શિત કરવું. કીબોર્ડમાંથી વાંચવા માટે પાયથોનમાં મૂળભૂત આદેશ ઇનપુટ છે. જો આપણે વિધેય ઇનપુટને કોઈ આર્ગ્યુમેન્ટ્સ સાથે બોલાવતા નથી અને તેને એક નામ તરીકે સોંપી છે, તો, વપરાશકર્તા નામના ડેટાને ઇનપુટ કમાન્ડ પર ચુઝર દ્વારા લખવામાં આવેલ કિંમત મળશે. યાદ રાખો કે, તે ઇનપુટની એક રેખા વાંચે છે. વપરાશકર્તા સંકેત આપે છે કે ઇનપુટ સમાપ્ત થાય છે તે રીતે, કીબોર્ડ પર રીટર્ન બટન અને રીટર્ન સુધી અક્ષરોની સંપૂર્ણ શ્રેણીને હિટ કરીને, પરંતુ રીટર્ન સહિત, વપરાશકર્તા ડેટાને સ્ટ્રિંગ તરીકે સંચારિત કરવામાં આવે છે. હવે, અલબત્ત, જો કાર્યક્રમ ફક્ત ઇનપુટ માટે તમારા માટે જ રાહ જોતો હોય, તો તે ખૂબ ગૂંચવણમાં મૂકે છે. તેથી, તમે પ્રોમ્પ્ટ પૂરું પાડવા માંગો છો, જે વપરાશકર્તાને સંદેશ છે, વપરાશકર્તાને કહેવા, અપેક્ષિત શું છે તેથી, તમે ઇનપુટ માટે દલીલ તરીકે શબ્દમાળા ઉમેરીને આવી વસ્તુ આપી શકો છો. જો તમે આની જેમ ઇનપુટ કરવા માટે દલીલ કરો છો, તો તે એક સ્ટ્રિંગ છે જે પ્રદર્શિત થાય છે જ્યારે વપરાશકર્તા ઇનપુટ માહિતી માટે માનવામાં આવે છે. હવે, આ સ્ટ્રિંગ તે દર્શાવવામાં આવે છે. તેથી, તમે તેને થોડો વધુ વપરાશકર્તા મૈત્રીપૂર્ણ બનાવવા યોગ્ય અનુકૂળનો કરી શકો છો. અમે એક મિનિટમાં એક ઉદાહરણ જોશો, પરંતુ તમે જગ્યા છોડી શકો છો અથવા તમે એક નવી રેખા દાખલ કરી શકો છો મૂળભૂત રીતે, તમે ઇનપુટ કમાન્ડનો ઉપયોગ ચુઝર તરફથી એક ઇનપુટ વાંચવા માટે કરી શકો છો અને તમે વપરાશકર્તાને તેનાથી શું અપેક્ષા છે તે કહેવા માટે સંદેશો દર્શાવી શકો છો. તેથી, અહીં શું થાય છે જો હું માત્ર ચુઝરડેટા ઇનપુટ () માટે સમાન હોઉં, તો પાયથોન પ્રોગ્રામ માત્ર રાહ જોશે અને હવે, જે વપરાશકર્તાને ખબર નથી કે શું અપેક્ષિત છે, અમને ખબર નથી કે તે કોઈ પ્રક્રિયા કરી રહી છે કે કેમ ઇનપુટની રાહ જોવી

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 02:30)

હવે, તે તારણ કાઢે છે કે, જો આપણે કંઈક ટાઇપ કરીએ અને એન્ટર દબાવીએ, તો તે પાછો આવશે. હવે, જો હું યુઝરડેટા નામની સામગ્રી માટે પૂછતો હોઉં તો તે મને મારી ટાઇપ કરેલ વસ્તુઓની સ્ટ્રિંગ પર અસર કરશે. તેથી, કોઈ સંદેશ વગર ઇનપુટ પ્રોમ્પ્ટ પૂરો પાડવાથી વપરાશકર્તા માટે મૂંઝવણ થઈ શકે છે. તો, આપણે જે કહ્યું તે હતું, આપણે આના જેવું ઇનપુટ પૂરું પાડવાનું કંઈક કહી શકીએ. હવે, તે અમને સંદેશ આપે છે, પરંતુ ઉદાહરણ તરીકે આપણે જે નંબર લખીએ છીએ તે સંદેશમાં અટવાઈ જાય છે. તે ખૂબ જ વાંચી શકાય તેવું નથી. તો, વપરાશકર્તાના ડેટા ખરેખર એક નંબર નથી, હવે, તે એક સ્ટ્રિંગ છે કારણ કે આપણે એક મિનિટમાં જોશું. પરંતુ હકીકત એ છે કે, અમને કોઈ જગ્યા કે બીજું કોઈ મળ્યું નથી. તે થોડી નીચ દેખાય છે તેથી, આપણે શું કહ્યું કે તમે વાસ્તવમાં, ઉદાહરણ તરીકે, કોલોન અને જગ્યા મૂકી શકો છો જેથી સંદેશ આની જેમ આવે. હવે, આ થોડું સરસ પ્રોમ્પ્ટ છે અને જો તમે ઇચ્છો તો નવી લાઈન `printfut` પણ કરી શકો છો, જે આ વિશિષ્ટ અક્ષર દ્વારા સંકેત આપે છે, બેકસ્લેશ `n` હવે, સંદેશ આવે છે અને પછી, તમે નવી લીટી પર લખો છો અને તમામ કેસોમાં, પરિણામ એ જ છે; વપરાશકર્તાના ડેટા, તમે ઇનપુટ વાંચી રહ્યા છો તે નામ, વપરાશકર્તા દ્વારા ટાઇપ કરેલ શબ્દમાળા પર સેટ થઈ જાય છે. જો હું તેને ફરીથી કરું છું અને જો હું 993 જેવા કંઈક બીજું લખું છું, ઉદાહરણ તરીકે, પછી વપરાશકર્તાના ડેટા શબ્દમાળા `"993"` બની જાય છે. તેથી, તમે સંદેશ સાથે ઇનપુટનો ઉપયોગ કરી શકો છો અને મેસેજ વાંચવાયોગ્ય તરીકે તમે કરી શકો છો.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 04:16)

જેમ આપણે જોયું, જ્યારે આપણે દુભાષિયો સાથે રમી રહ્યા હતા, ફંક્શન ઇનપુટ દ્વારા વાંચવામાં આવતી ઇનપુટ હંમેશાં એક સ્ટ્રિંગ છે. જો તમે કહો છો, તો સંખ્યામાં એક સંખ્યા અને વપરાશકર્તા પ્રકારો દાખલ કરો, આ વાસ્તવમાં નંબર નથી. જો તમે તેનો નંબર તરીકે ઉપયોગ કરવા માંગો છો, તો તમારે આ પ્રકારના રૂપાંતરણનો ઉપયોગ કરવો પડશે. યાદ રાખો, આપણી પાસે આ વિધેયો પૂર્ણાંક, `str` અને તેથી વધુ છે. તેથી, આપણે વપરાશકર્તાએ ટાઇપ કરેલું કન્ફર્મ ટાઇપ કરવા પૂર્ણાંક માટે ઇન્ટ ફંક્શનનો ઉપયોગ કરવો પડશે. હવે, અલબત્ત, યાદ રાખો કે, જો વપરાશકર્તા કેટલાક કચરોને લખે છે, તો તમને ભૂલ મળે છે, જમણે. જો વપરાશકર્તા કોઈ પ્રકારનું નહીં લખતું હોય, તો કંઈક યોગ્ય હોય, તો તમને ભૂલ મળશે. તેથી, આપણે શું કરી શકીએ તે છે, અમે આ ભૂલ સાથે વ્યવહાર કરવા માટે અપવાદ હેન્ડલિંગનો ઉપયોગ કરી શકીએ છીએ. તેથી, આપણે શું કહી શકીએ તે છે, વપરાશકર્તાના ડેટા ને અજમાવી જુઓ આ તે કોડ છે જે પહેલાં આપણી પાસે છે: તે 2 લીટીઓ. તેથી, આપણે શું કહેવા માગીએ છીએ, આપણે આ લીટીઓનો પ્રયાસ કરીશું, પરંતુ જો વપરાશકર્તા કોઈ પ્રકારનો પ્રકાર કે જે કોઈ સંખ્યા નથી, તો પછી આપણે તેને ફરીથી લખવા માટે કહીશું અને તે ચાલુ થશે, તે પ્રકારની ભૂલ પાચથોન માં મૂલ્ય ભૂલ કહેવામાં આવે છે

(સ્લાઇડ સમય જણાવો: 05:20)

તો, તમે પાચથોન ના ઇન્ટરપ્રીટર પર જઈને ચકાસી શકો છો. પાચથોન ના દુભાષિયામાં, જો આપણે કેટલીક અવિવેકી વસ્તુઓમાં લાગુ પાડવાનો પ્રયાસ કરીએ છીએ, તો અમને મૂલ્યની ભૂલ મળે છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 05:32)

તો, આપણે શું કહીએ છીએ તે છે કે, જો આપણને મૂલ્યની ભૂલ મળે, તો અમે યોગ્ય પગલાં લેવા માંગીએ છીએ. તો, આને અવરોધિત કરવાનો પ્રયાસ કરો અને જો આપણને મૂલ્યની ભૂલ દેખાય, તો આપણે શું કરીએ, આપણે મેસેજ છાપીએ છીએ. હવે, આપણે આ પછી જ છાપવા જઈ રહ્યા છીએ, પરંતુ અમે સંદેશને વપરાશકર્તાને છાપીએ છીએ, આ સંખ્યા નથી, ફરીથી પ્રયાસ કરો અને આ હવે, આખી વસ્તુ જ્યારે લૂપમાં સંલગ્ન છે અને આ જ્યારે લૂપ હોય ત્યારે શરત સાચું. બીજા શબ્દોમાં કહીએ તો, શરત ક્યારેય ખોટા બનશે નહીં; આ જ્યારે લૂપ સંખ્યા માટે પુછપરછ ચાલુ રાખશે. તો, આપણે આમાંથી

કેવી રીતે મેળવી શકીએ? ઠીક છે, જો હું અહીં આવીશ અને મને મુલ્યની ભૂલ મળે, તો તે પાછું જશે અને જ્યારે ફરીથી ફરીથી એક્ઝિક્યુટ થશે. પણ જો ત્યાં કોઈ ભૂલ ન હોય, તો યાદ રાખો, અહીં કોઈ ભૂલ ન હોય તો, તે બીજાની તરફ આવશે. બીજું કોઈ ભૂલ નથી અને બીજું શું કરે છે તો આ અન્યને ચલાવવામાં આવે છે, તે અમને આ પાપી ચક્રમાંથી બહાર કાઢે છે. બીજા શબ્દોમાં કહીએ તો, અમે અનંત લૂપમાં છીએ, જ્યાં આપણે યુઝર પાસેથી એક વધુ માહિતી મેળવવાનો પ્રયાસ કરતા રહીએ ત્યાં સુધી, જ્યાં સુધી આપણે કંઈક શોધીએ છીએ અને જ્યારે આપણે તે મેળવીએ છીએ, ત્યારે આપણે લૂપમાંથી બહાર નીકળી જઈએ છીએ. ઇનપુટ અને આઉટપુટના સંદર્ભમાં, અપવાદનો ઉપયોગ કરવા માટે આ એક અન્ય પ્રકારની રૂઢિગત રીત છે. જેમ જેમ આપણે છેલ્લા પ્રવચનમાં જણાવ્યું હતું કે, ઇનપુટ અને આઉટપુટ અંતર્ગત ભૂલની શક્યતા છે, કારણ કે, તમે અનિશ્ચિત, ક્રિયાપ્રતિક્રિયાત્મક વાતાવરણ સાથે વ્યવહાર કરી રહ્યા છો, જે વસ્તુઓ કરી શકે છે, જે તમે અપેક્ષા કરી શકતા નથી અથવા નિયંત્રણ કરી શકતા નથી. તેથી, ખાતરી કરવા માટે કે જે દિશામાં તમે અપેક્ષા કરો છો તે દિશામાં જાય તે માટે તમારે યોગ્ય પગલાં લેવા જોઈએ. ક્રિયાપ્રતિક્રિયાનો બીજો ભાગ સંદેશ દર્શાવતો હોય છે, જેને આપણે સ્કીન પર માનક આઉટપુટ અથવા પ્રિન્ટિંગ કહીએ છીએ.

(સ્લાઈડ ટાઇમ નો ઉલ્લેખ કરો: 06:59)

અને, આ પ્રિન્ટ સ્ટેટમેન્ટનો ઉપયોગ કરીને પ્રાપ્ત થાય છે, જે આપણે ક્યારેક અનૌપચારિક રીતે જોયા છે, તે વ્યાખ્યાયિત કર્યા વિના ઔપચારિક રીતે. પ્રિન્ટ સ્ટેટમેન્ટનું મૂળ સ્વરૂપ, અલ્પવિરામ દ્વારા અલગ પાડવામાં આવેલા મૂલ્યોનો ક્રમ આપવાનો છે. તેથી, પ્રિન્ટ એક્સ, વાચ x ની મુલ્ય પ્રદર્શિત કરશે, પછી, એક જગ્યા, તે પછી, y ની મુલ્ય પ્રિન્ટ એ, બી, સી; 3 મુલ્ય પ્રદર્શિત કરશે; એ, બી અને સીના મૂલ્યો, ખાલી જગ્યા દ્વારા અલગ. હવે, બીજી વસ્તુ જે આપણે કરી શકીએ તે છે, સીધા શબ્દમાળા અથવા સંદેશને છાપો. જેમ આપણે અગાઉની ઉદાહરણમાં જોયું તેમ, આપણે કહી શકીએ કે, “કોઈ સંખ્યા નથી. ફરીથી પ્રયત્ન કરો” આ સ્કીન પર આ સ્ટ્રિંગ પ્રદર્શિત કરશે. હવે, તમે આ બે બાબતોને રસપ્રદ રીતે સંયોજિત કરી શકો છો. તેથી, પ્રિન્ટ સામાન્ય રીતે, અલ્પવિરામથી અલગ કરીને, મનસ્વી લિંક્સની વસ્તુઓનો ક્રમ લે છે. આ વસ્તુઓ સંદેશ અથવા નામો હોઈ શકે છે તેથી, આપણે વસ્તુઓને કહી શકીએ છીએ, આપણે ધારવું જોઈએ કે x ની કિંમત છાપવી છેડી, પરંતુ અમે આઉટપુટને સૂચવવા માંગીએ છીએ, જે x છે, જે y છે. કહીને બદલે, ફક્ત x અલ્પવિરામ વાચ, જે સ્કીન પર 2 મૂલ્યો ઉત્પન્ન કરે છે, જેનો અર્થ એ નથી કે જે x છે અને જે y છે, તેનાથી આ વધુ વિસ્તૃત પ્રિન્ટ સ્ટેટમેન્ટ હોઈ શકે છે, જે 4 વસ્તુઓ છાપે છે. તે જે પ્રિન્ટ કરે છે તે એક સંદેશ છે જે કહે છે, કિંમતો x કોલોન છે. આ x કોલોન પ્રિન્ટ કરશે; તે જગ્યા છોડશે; પછી, તે x ની વર્તમાન કિંમતની કિંમત છાપશે; પછી, તે જગ્યા પછી વાચ કોલન છાપશે અને પછી, તે y ની વર્તમાન કિંમતને છાપશે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 08:39)

તેથી, આપણે વધુ વાંચવા યોગ્ય છે તે અર્થપૂર્ણ આઉટપુટ ઉત્પન્ન કરવા માટે, નામ સાથેના મૂલ્યો ધરાવતા સંદેશોને એકબીજા સાથે જોડે છે. ડિકોલ્ટ રૂપે, જ્યારે પણ તેને અમલમાં મુકાશે ત્યારે નવી લીટી પ્રિન્ટ ઉમેરે છે. બીજા શબ્દોમાં કહીએ તો, દરેક પ્રિન્ટ સ્ટેટમેન્ટ, અમે અત્યાર સુધી જે રીતે કર્યું છે તે છાપીએ છીએ, એક નવી લીટી પર દેખાય છે કારણ કે અગાઉના પ્રિન્ટ સ્ટેટમેન્ટના આઉટપુટને નવી લીટીમાં ખસેડે છે. હવે, જો આપણે આને અંકુશમાં લેવા માંગીએ છીએ, તો અમે એક વૈકલ્પિક દલીલનો ઉપયોગ કરી શકીએ છીએ જેને અંત કહેવાય છે. તેથી, આપણે કહી શકીએ કે, આ પ્રિન્ટ સ્ટેટમેન્ટના અંતમાં આપણે શું કરવું જોઈએ? મૂળભૂત રીતે, અહીંનું મૂલ્ય આ નવી લીટી પાત્ર છે. તેથી, આપણે આને બીજી કોઈ જગ્યાએ બદલી શકીએ છીએ. અહીં એક ઉદાહરણ છે. ધારો કે, આપણે આ 3 નિવેદનો લખીએ છીએ. પ્રથમ નિવેદનમાં કહે છે, “પ્રિન્ટ”; ચાલુ રાખો “; આ ફક્ત એક શબ્દમાળા છે; પરંતુ એક જગ્યા અંત સેટ. બીજી લીટી કહે છે, “પ્રિન્ટ”; એ જ લાઇન “; અને તે પછી, તે કહે છે, સંપૂર્ણ સ્ટોપ અને એક નવી લાઇન સમાપ્ત થાય છે. અને પછી, ત્રીજી નિવેદન કહે છે,

પ્રિન્ટ કરો; આગળની રેખા. તેથી, આપણે શું કરી રહ્યા છીએ, પ્રથમ 2 સ્ટેટમેન્ટમાં, આપણે ડિફોલ્ટને બદલી રહ્યા છીએ. ડિફોલ્ટ નવી લીટી છાપવાનું હોત, પરંતુ પ્રથમ સ્ટેટમેન્ટ નવી લીટી છાપતું નથી. જો આપણે આને છાપીએ તો આપણે શું જોઈએ છે, પ્રથમ બે નિવેદનો એક જ લીટી પર ચાલુ રહે છે, એક લીટી પર આવે છે, કારણ કે, અમે ડિફોલ્ટ પ્રિન્ટ નવી લીટી અક્ષમ કરી દીધી છે અને અમે સ્પષ્ટપણે અહીં નવી લાઇન મૂકી છે અને આને ફરજ પડી છે આગળના વાક્ય પર આવવા માટે આગામી એક. જો આપણે આ ભંગ કરીએ અને જુઓ કે અહીં શું થાય છે, તો આપણે જોયું કે, પ્રથમ નિવેદનમાં, અમારું આ અંત હતું, જે કહે છે કે ખાલી જગ્યા દાખલ કરો અને આ શા માટે આપણે પ્રથમ વાક્યમાં x અને y શબ્દ એ જ બીજી લાઇનમાં આવે છે. અન્યથા, x અને y એક જ શબ્દ તરીકે મળીને મિશ્રિત કરવામાં આવશે, અધિકાર. તેથી, અવકાશના બરાબર અંત આ પ્રિન્ટને જગ્યા દ્વારા આગામી પ્રિન્ટથી અસરકારક રીતે અલગ કરે છે. આગળનું પ્રિન્ટ સ્ટેટમેન્ટ એક સંપૂર્ણ સ્ટોપ અને એક નવી લીટી દાખલ કરે છે. નિશ્ચિતપણે, જોકે આ જ વાક્યનો શબ્દ સંપૂર્ણ સ્ટોપ વિના પૂરો થાય છે, અમે સંપૂર્ણ સ્ટોપ ઉત્પન્ન કરીએ છીએ અને પૂર્ણ સ્ટોપ ઉત્પન્ન કર્યા પછી, તે નવી લાઇન ઉત્પન્ન કરે છે અને છેવટે, આ નવી લીટી પછી, આગામી રેખા નવી લીટી અને અલબત્ત આવે છે, કારણ કે અહીં અમે કંઈપણ ન કહી નહોતી; જો આપણે તેના પછી છાપીએ તો, આપણે નવી લીટી પર છાપશે. અન્ય વસ્તુ જે અમે નિયંત્રિત કરવા માંગીએ છીએ તે એ છે કે કઈ વસ્તુઓ એક લીટી પર અલગ છે. અમે જણાવ્યું હતું કે, જો આપણે પ્રિન્ટ x અલ્પવિરામ વાચ કહીએ છીએ, તો પછી x અને y મૂળભૂત દ્વારા જગ્યા દ્વારા અલગ કરવામાં આવશે, જમણે. જો આપણે આ કરીએ તો પ્રિન્ટ એક્સ, વાચ, આપણે x બરાબર 7, y બરાબર 10 અને આપણે કહીશું x એ x છે અને y બરાબર છે અને પછી, આપણે સંપૂર્ણ સ્ટોપ સાથે સમાપ્ત કરવા માંગીએ છીએ. આ તમે શું કરવા માંગો છો; તમે સ્ટ્રીંગ લખવા માંગો છો, x છે, પછી x ની કિંમત અને y છે, પછી વાચ કિંમત અને પછી, એક સંપૂર્ણ સ્ટોપ હવે, કારણ કે દરેક જગ્યા ખાલી છે, આપણે શું શોધીએ છીએ તે, આપણે અહીં જગ્યા શોધીએ છીએ; શું તમે આ જુઓ છો? આ દંડ છે તેથી, આપણને અહીં જગ્યા મળી છે, કારણ કે, આ અલ્પવિરામથી છે; આપણને અહીં જગ્યા મળી છે, જે આ અલ્પવિરામથી છે; આપણને અહીં જગ્યા મળે છે, જે આ અલ્પવિરામથી છે. અને પછી, અમે 10 અને પૂર્ણ સ્ટોપ વચ્ચે અનિચ્છિત જગ્યા મેળવો. તો, આપણે કેવી રીતે આ જગ્યાથી છુટકારો મેળવી શકીએ, બીજી જગ્યા, અધિકાર? અમે જગ્યા પછી આવવા માટે એક સંપૂર્ણ સ્ટોપ નથી માંગતા. તેથી, જેમ અમારી પાસે વૈકલ્પિક દલીલ અંત છે, અમારી પાસે એક વૈકલ્પિક દલીલ એસઈપી છે, જે સ્પષ્ટ કરે છે કે કઈ સ્ટ્રીંગ અલગ કરવા માટે વપરાવી જોઈએ. તેથી, ઉદાહરણ તરીકે, જો આપણે પહેલાંની વસ્તુને લઈએ, તો આપણે કહી શકીએ, તેને કોઈ પણ વસ્તુ સાથે અલગ કરી શકતા નથી. હવે, અલબત્ત, તેને કોઈ પણ વસ્તુ સાથે અલગ ન કરો, તે બદલાવે છે, કારણ કે, તે પછી, આ એક્સ 7 છે અને તેને ફ્યુઝ કરવામાં આવશે અને આ ફ્યુઝ કરવામાં આવશે. તેથી, આપણે તેના બદલે શું કરીએ, અમે અહીં સ્પષ્ટપણે જગ્યા મૂકીએ છીએ. અગાઉ, અમારી પાસે કોઈ જગ્યા નથી, અંતે, અવતરણની આસપાસ. હવે, અમે જગ્યાઓ મૂકીએ છીએ જ્યાં આપણે તેમને જોઈએ છીએ અને અમે કહીએ છીએ કે કોઈપણ અન્ય જગ્યાઓ ન મૂકીએ. તેથી, આ શું કહેશે કે, એક્સ જગ્યા છે, હું આ જગ્યા આપીશ; જગ્યા ન મૂકી, x ની કિંમત મૂકી; જગ્યા ન મુકો, હવે, હું જગ્યા આપીશ. તેથી અને y છે, એક જગ્યા આપો અને પછી, અહીં જગ્યા ન મુકો. આ અલ્પવિરામ કોઈ પણ અવકાશમાં ફાળો આપતું નથી, કારણ કે મેં સુયોજિત કરેલું વિભાજક ખાલી હોવું જોઈએ અને ખાસ કરીને, આનો અર્થ શું છે કે, આ છેલ્લો અલ્પવિરામ, વાચ અને પૂર્ણ સ્ટોપ વચ્ચેનો અલ્પવિરામ, જગ્યા બનાવશે નહીં. અને વાસ્તવમાં, જો તમે આ એક્ઝેક્યુટ કરો છો, તો તમે આઉટપુટ મેળવશો, એક્સ 7 છે અને y 10 છે અને જે રીતે કામ કરે છે તે છે, તે આ પ્રથમ બ્લોક છે. આ બધું અહીં છે. પછી, આ બીજો બ્લોક છે, આ તે છે. પછી, આ ત્રીજા બ્લોક છે, જે આ સંપૂર્ણ બાબત છે, આપેલ જગ્યાઓ સાથે અને પછી, આ વાચનું મૂલ્ય છે અને છેવટે, આ સંપૂર્ણ સ્ટોપ છે. પ્રિન્ટ સ્ટેટમેન્ટનું આઉટપુટ નિયંત્રિત કરવાનો આ એક માર્ગ છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 13:28)

તેથી, વૈકલ્પિક દલીલો અંત અને sep સાથે, અમે નિયંત્રિત કરી શકીએ છીએ જ્યારે એક જ લીટી પર ક્રમિક છાપે ચાલુ રહે અને અમે અમુક મર્યાદિત હદ સુધી નિયંત્રિત કરી શકીએ, કેવી રીતે આ મૂલ્યો લીટી પર અલગ છે પરંતુ અમે ખરેખર ઘણું બધું કરવા માંગીએ છીએ. અમે વસ્તુઓ ક્રમ મૂકી શકો છો, જેથી, તેઓ બધા લાઇન, અધિકાર. ધારો કે, આપણે પ્રિન્ટ સ્ટેટમેન્ટનો ઉપયોગ કરીને કોષ્ટક છાપીએ છીએ, અમે એ સુનિશ્ચિત કરવા માંગીએ છીએ કે કોલમ અપ લાઇન. તેથી, આપણે કહી શકીએ કે, દરેક વસ્તુ જે આપણે છાપવા માંગીએ છીએ, જેમ કે આપણે સંખ્યાઓનો ક્રમ છાપી રહ્યા છીએ, વાક્ય દ્વારા રેખા, કારણ કે, સંખ્યાઓ અલગ અલગ પહોળાઈ હોઈ શકે છે; કેટલાક 3 અંકો હોઈ શકે છે, કેટલાક 5 અંકો હોઈ શકે છે; આપણે કહી શકીએ કે 7 અક્ષરો પહોળાઈ પર અધિકાર મૂકવા માટે, બધા જ છાપો. આ એક વસ્તુ છે જે આપણે કરવા માંગીએ છીએ, ટેક્સ્ટ સંરેખિત કરી શકીએ છીએ. હવે, આ ગોઠવણીની અંદર, આપણે વસ્તુઓને ડાબે અથવા જમણે સંરેખિત કરવા માંગીએ છીએ. જો અમારી પાસે મૂળભૂત છે, તો 10 અક્ષરો લખો; જો તે સંખ્યાઓ હોય, તો અમે તેમને જમણા સંરેખિત કરવા ઈચ્છીએ છીએ, જેથી એકમોની સંખ્યા એક સાથે બંધાયેલી હોય; જો તેઓ નામો છે, તો અમે તેમને ડાબી ગોઠવાયેલી જોઈ શકીએ છીએ, જેથી અમે તેમને ડાબેથી જમણી બાજુએ વાંચી શકીએ, વગર તે ખરબચડી દેખાશે. અને જો આપણે સરેરાશ અથવા કંઈક ગણતરી જેવી વસ્તુઓ કરી રહ્યા હોય, તો આપણે આખી વસ્તુને દર્શાવવા માંગતા ન હોઈએ; આપણે તેને બે દશાંશ પોઇન્ટ સુધી પહોંચું કરી શકીએ છીએ; કહેવું, તે ચલણ છે અથવા તે કંઈક છે. આ આઉટપુટને ફોર્મેટ કરવાની તમામ વધુ જટિલ રીત છે અને આપણે આગામી લેક્ચરમાં આ કેવી રીતે કરવું તે જોઈશું. પરંતુ હમણાં, તમે ન્યૂનતમ ફોર્મેટિંગ કરવા માટે ઓવરને અને sep નો ઉપયોગ કરી શકો છો.

(સ્લાઇડ સમયનો સંદર્ભ લો: 14:53)

સારાંશ માટે, તમે કીબોર્ડમાંથી વાંચવા માટે વૈકલ્પિક સંદેશ સાથે ઇનપુટ સ્ટેટમેન્ટનો ઉપયોગ કરી શકો છો. તમે પ્રિન્ટ સ્ટેટમેન્ટનો ઉપયોગ કરીને સ્ક્રીન પર છાપી શકો છો. હવે, અમે શરૂઆતમાં ઉલ્લેખ કર્યો છે કે, પાયથોન 2 અને 3 વચ્ચે કેટલાક તફાવતો છે અને અહીં તે વધુ સ્પષ્ટ તફાવત છે જે તમે જોશો, જો તમે પાયથોન 2 કોડ જુઓ, જે વિવિધ સ્રોતોમાંથી ઉપલબ્ધ છે. પાયથોન 2 માં, તમે છાપો જગ્યા કહી શકો છો અને પછી, આપીએ, જે પાયથોન ને પ્રિન્ટ કરવા માટે દલીલો તરીકે આપવામાં આવે છે 3. પાયથોન 3 એ તેના પર ભાર મૂકે છે જેને કૌંસમાં કાર્ય કરે છે; પાયથોન 2 માં કૌંસ વૈકલ્પિક છે. તેથી, તમે ઘણીવાર પિથન 2 કોડમાં, કોઈ પણ કૌંસ વિના આપેલ પ્રિન્ટ x, y, જેવો દેખાશે એવી કોઈ વસ્તુ દેખાશે. આ પાયથોન 2 માં કાનૂની છે; આ પાયથોન માં કાયદેસર નથી 3. ફક્ત આ વિશે સાવચેત રહો. અને આપણે જે જોયું તે એ છે કે, મર્યાદિત સંખ્યામાં નિયંત્રણ સાથે, આપણે ઇચ્છતા હોવાથી પ્રિન્ટની વર્તણૂક કરી શકીએ છીએ. તો, આપણે સ્પષ્ટ કરી શકીએ કે અંતે શું મુકવું. ખાસ કરીને, અમે કહી શકીએ છીએ કે નવો રેખા ન મૂકવા. તેથી, સમાન રેખામાં છાપવાનું ચાલુ રાખો અને આપણે મૂળભૂત જગ્યા પાત્ર સિવાયની અન્ય કોઈ વસ્તુ દ્વારા અલગ કરી શકીએ છીએ.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રો. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટીટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
અઠવાડિયું - 05 લેક્ચર - 03 ફાઇલોને હેન્ડલિંગ

(સ્લાઇડ ટાઇ નો ઉલ્લેખ કરો: 00:02)

છેલ્લા પ્રવચનમાં આપણે જોયું કે કેવી રીતે ઉપયોગ કરવો ઇનપુટ અને પ્રિન્ટ સ્ટેટમેન્ટ જે કીબોર્ડ છે તે સ્ટાન્ડર્ડ ઇનપુટમાંથી ઇનપુટ એકત્રિત કરે છે, અને પ્રિન્ટનો ઉપયોગ કરીને સ્ક્રીન પરના મૂલ્યો દર્શાવવા માટે. હવે, આ નાની માત્રામાં માહિતી માટે ઉપયોગી છે, પરંતુ આપણે મોટી સંખ્યામાં ડેટા વાંચવા અને લખવા માંગીએ છીએ. તેમને હાથ દ્વારા લખવું અથવા તેમને આ સ્ક્રીનમાં સ્ક્રોલ પાસ તરીકે જોવા માટે અવિભાજ્ય છે. તેથી, મોટા ડેટા માટે અમને ડિસ્ક પર રહેલી ફાઇલો સાથે વ્યવહાર કરવા માટે ફરજ પાડવામાં આવે છે. તેથી, અમારે મોટી સંખ્યામાં ડેટા વાંચવો પડશે જે ડિસ્કમાં પહેલાથી જ ફાઇલ પર લખાયેલો છે અને જે આપણે ગણતરી કરીએ છીએ તે ખાસ કરીને ડિસ્ક પર બીજી ફાઇલમાં પાછો આવે છે. હવે, ડિસ્ક સાથે વ્યવહાર કરતી વખતે ધ્યાનમાં રાખવા માટેની એક વસ્તુ એ છે કે ડિસ્ક વાંચવા અને લખવા માટે મેમરી વાંચવા અને લખવા કરતાં ઘણો ધીમી છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 00:48)

આની આસપાસ જવા માટે મોટાભાગની સિસ્ટમો મોટા બ્લોક્સમાં ડેટા વાંચશે અને લખશે. કલ્પના કરો કે તમારી પાસે વિશાળ સ્ટોરેજ સુવિધા છે જેમાં તમે વસ્તુઓને મોટી કાર્ટનમાં સંગ્રહિત કરો છો. હવે, જ્યારે તમે જાઓ અને કોઈ વસ્તુ લાવશો ત્યારે તમે એક સમયે એક કાર્ટન લાવશો, પછી ભલે તમે માત્ર શોધી રહ્યા હોવ, તે બેટોનનો એક પુસ્તક કહે, તમે સ્ટોરેજ સુવિધામાંથી એક બટનોની બહાર એક પુસ્તક લઈ જશો નહીં. સંપૂર્ણ પૂંઠું અને પછી જ્યારે તમે વસ્તુઓને ફરીથી પાછું મૂકવા માંગો છો ત્યારે તમે તેમને એક પૂંઠું માં ભેગા કરો અને તેમને પાછા મૂકો. તે જ રીતે, જે રીતે મેમરી અને ડિસ્ક વચ્ચે આગળ અને પાછળની માહિતી વહે છે તે બ્લોકો કહેવાય છે. તેથી, જો તમે માત્ર એક જ મૂલ્ય અથવા ફક્ત એક લીટી વાંચવા માંગતા હોવ તો તે વાસ્તવમાં ડિસ્કમાંથી મોટા જથ્થામાં માહિતી મેળવે છે અને તેને બફર તરીકે કહેવામાં આવે છે અને પછી તમે બફરમાંથી જે જોઈએ તે વાંચી શકો છો. તેવી જ રીતે, જ્યારે તમે ડિસ્કમાં લખવા માંગો છો ત્યારે તમે બફરમાં તમારો ડેટા ભેગા કરો છો જ્યારે બફર ડિસ્ક પર લખવામાં આવે તેટલા પ્રમાણમાં હોય છે, પછી ડેટા અથવા બ્લોકના એક ભાગને ડિસ્ક પર પાછા લખવામાં આવે છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 01:49)

જ્યારે આપણે ડિસ્કમાંથી વાંચીએ છીએ અને લખીએ છીએ ત્યારે આ બફર સાથે જોડાયેલો પ્રથમ વસ્તુ છે. આને ફાઇલ ખોલવાનું કહેવામાં આવે છે. તેથી, જ્યારે આપણે કોઈ ફાઇલ ખોલીએ છીએ ત્યારે આપણે કોઈ ફાઇલ હેન્ડલ નામની વસ્તુ બનાવીએ છીએ અને તમે કલ્પના કરી શકો છો કે આ બફરની એક્સેસ મેળવવા જેવી છે કે જેમાંથી તે ફાઇલમાંથી મેમરી મેમરીમાં વાંચી શકાય છે અથવા બેક અપ લેબલ કરી શકાય છે. હવે, આ ફાઇલ ખોલી, ફાઇલ સાથે જે કંઈ કરીએ તે બધું જ હેન્ડલ કરે છે, વાસ્તવમાં આ ફાઇલ હેન્ડલના સંદર્ભમાં થાય છે. તેથી, અમે ડિસ્કમાંથી વાંચવા અને લખવાનો સીધી પ્રયાસ કરતા નથી, તેના બદલે આપણે આ ફાઇલ હેન્ડલની મદદથી ખુલ્લી બફરમાંથી વાંચીએ છીએ અને લખીએ છીએ અને છેલ્લે, જ્યારે અમે અમારા પ્રોસેસિંગ સાથે પૂર્ણ કરીએ છીએ ત્યારે અમારે ખાતરી

કરવી જરૂરી છે કે તમામ ડેટા કે અમે લખ્યું છે પાછા જાય છે. તેથી, આ ફાઇલ બંધ કરીને કરવામાં આવે છે. તેથી, ફાઇલને બંધ કરવું બે અસરો છે; પ્રથમ અસર એ છે કે તે ખાતરી કરવા માટે કે જે બધા ફેરફારો જે અમે ફાઇલમાં બનાવવાનો હેતુ ધરાવે છે. કોઈપણ ડેટા જે આપણે ફાઇલમાં લખવા માંગીએ છીએ તે વાસ્તવમાં બફરમાં લેવામાં આવે છે અને ડિસ્ક પર મૂકવામાં આવે છે અને આ તકનીકી રીતે બફરને ફ્લશ કરે છે. તેથી, ફાઇલને બંધ કરવાથી આઉટપુટ બફરને ફ્લશ કરવામાં આવે છે તેની ખાતરી કરો કે બધા હક્કી ફાઇલમાં પાછા જાય છે અને હારી નહીં જાય અને બીજું વસ્તુ તે કરે છે કે તે અમુક અર્થમાં આ બફરને દૂર કરે છે. તેથી, તે ફાઇલ હેન્ડલ ડિસ્કનેક્ટ કરે છે જે અમે હમણાં જ સેટ કરી છે. હવે, આ ફાઇલ હવે અમારી સાથે કનેક્ટેડ નથી, જો આપણે તેને ફરીથી વાંચી અથવા લખવી હોય તો ફરીથી તેને ખોલવું પડશે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 03:11)

ફાઇલ ખોલવા માટેની આદેશ ફક્ત 'ખુલ્લી' છે તમે આપેલ પ્રથમ દલીલ તમારા ડિસ્ક પર વાસ્તવિક ફાઇલ નામ છે. હવે, આ તમે જે સિસ્ટમનો ઉપયોગ કરી રહ્યા છો તેના પર થોડો આધાર રાખશે, પરંતુ સામાન્ય રીતે તેનો પ્રથમ ભાગ અને એક્સટેન્શન છે. ઉદાહરણ તરીકે, આ આદેશો, જુસીડી ડોટ પીઇ ફાઇલ ખોલવા માટે. હવે સર્વથા, જો તમે માત્ર ફાઇલનું નામ આપો તો તે વર્તમાન ફોલ્ડર અથવા ડિરેક્ટરીમાં દેખાશે જ્યાં તમે સ્ક્રિપ્ટ ચલાવી રહ્યા છો. તેથી, તમે એક ફાઇલ નામ આપી શકો છો જે પાથ આપીને તમારી ડાયરેક્ટરી પદાનુક્રમના જુદા ભાગને અનુસરે છે અને તમે કેવી રીતે પાથનું વર્ણન કરો છો તે તેના પર આધાર રાખે છે કે તમે વિન્ડો અથવા યુનિક્સ પર કામ કરી રહ્યા છો, તમે કઈ ઓપરેટિંગ સિસ્ટમનો ઉપયોગ કરી રહ્યા છો. હવે, તમે જુઓ છો ત્યાં બીજી દલીલ છે, જે અક્ષર 'આર' છે. આ આપણને કહે છે કે કેવી રીતે ફાઇલ ખોલવી છે. તેથી, તમે કલ્પના કરી શકો છો કે જો તમે કોઈ ફાઇલમાં ફેરફાર કરી રહ્યા છો અને તેને લખીને લખો છો, તો તે મૂંઝવણ બનાવી શકે છે. તેથી, અમારે શું કરવાનું છે તે અગાઉથી નક્કી કરે છે કે આપણે કોઈ ફાઇલમાંથી વાંચી જશું અથવા તેને લખીશું, તો અમે બન્નેને કરી શકતા નથી. ત્યાં કોઈ રીત નથી કે આપણે એક સાથે ફાઇલમાંથી વાંચી શકીએ અને તે ખુલ્લું હોવા પર તેને સંશોધિત કરી શકીએ. તેથી, 'r' દ્વારા વાંચવામાં આવે છે હવે, બે સ્વરૂપોમાં લખો, આપણે શરૂઆતથી નવી ફાઇલ બનાવી શકીએ છીએ. આ કિસ્સામાં આપણે 'w' અક્ષરનો ઉપયોગ કરીએ છીએ. તેથી, 'ડબલ્યુ' એક નવી ફાઇલ લખી લેવા માટે છે, આપણે આ અંગે થોડી સાવચેત રહેવું જોઈએ કારણ કે જો આપણે કોઈ ફાઇલ લખીએ જે પહેલાથી જ બહાર નીકળે તો વધુ 'ડબલ્યુ' સાથે ખોલીને તે વિષયવસ્તુ પર ફરીથી લખીશું જે અમારી પહેલાથી જ હતી. બીજી વસ્તુ જે કરવું ઉપયોગી છે તે ફાઇલ લેવાની છે જે પહેલાથી અસ્તિત્વમાં છે અને તેમાં કંઈક ઉમેરવું છે. તેને ઉમેરવામાં આવે છે તેથી, આપણી પાસે બે લેખન સ્થિતિઓ છે; 'W' લખવા માટે અને 'એ' માટે ઉમેરો. શું ઉમેરું હશે તે ફાઇલ લેશે જે પહેલાથી અસ્તિત્વમાં છે અને ફાઇલના અંતે લખતી નવી સામગ્રીને ઉમેરશે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 05:02)

એકવાર અમારી પાસે ફાઇલ ખુલ્લી છે, ચાલો જોઈએ કે કેવી રીતે વાંચવું. તેથી, આપણે ફાઇલ હેન્ડલ દ્વારા વાંચવા આદેશનો ઉપયોગ કરીએ છીએ. આ અન્ય ફંક્શનો જેવું છે જે તમે શબ્દમાળાઓ સાથે જોયું હતું અને તેથી, જ્યાં આપણે ઓબ્જેક્ટમાં કાર્ય જોડીએ છીએ. તેથી, એફએચ ફાઇલ હેન્ડલ છે જે આપણે ખોલી છે, આપણે તેમાંથી વાંચવા માંગીએ છીએ. તેથી, આપણે એફએચ ડોટ વાંચીએ છીએ, એફએચ ડોટ શું વાંચે છે તે ફાઇલને એકલ સ્ટ્રિંગ તરીકે ગણી જાય છે અને તે રીટર્ન કરે છે અને પછી આપણે તેને કોઈ પણ નામ પર સોંપી શકીએ છીએ, અહીં આપણે નામ સમાવિષ્ટોનો ઉપયોગ કરીએ છીએ. તેથી, સમાવિષ્ટો હવે સમગ્ર ડેટાને અસાધન કરે છે જે ફાઇલ હેન્ડલમાં છે જે એક શબ્દમાળામાં એફએચ દ્વારા નિર્દેશ કરે છે. હવે, આપણે ફાઇલનો ઉપયોગ પણ કરી શકીએ છીએ, આપણે ખાસ કરીને ટેક્સ્ટ ફાઇલો સાથે વ્યવહાર કરીએ છીએ. તેથી, ટેક્સ્ટ ફાઇલ સામાન્ય રીતે લીટીઓ ધરાવે છે; ઉદાહરણ તરીકે,

લીટીઓ પછી લીટીઓ પછી લીટીઓ હોય છે. કુદરતી એકમ પાઠોનો એક ટોળું છે જે નવા રેખા પાત્ર સાથે સમાપ્ત થાય છે. જો તમને યાદ છે કે આ ઇનપુટ કમાન્ડ શું કરે છે, તો ઇનપુટ કમાન્ડ તમારા માટે રાહ જુએ છે અને તમે કંઈક ટાઇપ કરો છો અને પછી તમે નવી રિટર્ન છે તે રીટર્ન દબાવો છો અને વળતર સુધી તમે જે કંઈ ટાઇપ કરો છો તે ઇનપુટ દ્વારા નામના સ્ટ્રિંગ તરીકે ટ્રાન્સમિટ થાય છે. ઇનપુટને સોંપેલું છે તેથી, રીડલાઇન એવી છે, પરંતુ રીડલાઇન અને ઇનપુટ વચ્ચે તફાવત એ છે કે, જ્યારે તમે રેખા વાંચશો તો તમને ઇનપુટ સ્ટ્રિંગ સાથે છેલ્લા નવા રેખા પાત્ર મળશે. જ્યારે તમે ઇનપુટ કહી શકો છો ત્યારે ફક્ત તમને જે અક્ષરો મળે છે તે છેલ્લા નવી લીટી પહેલાં આવે છે નવી લીટી શામેલ નથી, પરંતુ રેડલાઇનમાં તમે નવા રેખાના અક્ષર મેળવી શકો છો. તેથી, તમારે યાદ રાખવું પડશે કે તમારી સ્ટ્રિંગના અંતમાં તમારી પાસે અતિરિક્ત અક્ષર છે. તેથી, આ પરંપરાગત રીતે જાણીતી બેકસ્લેશ n છે. બેકસ્લેશ n એ એક સંકેત છે જે એક અક્ષરને દર્શાવે છે તેમ છતાં બે અક્ષર દેખાય છે. આ નવા રેખા પાત્ર હોવાનું માનવામાં આવે છે. હવે, વાસ્તવિક નવી લાઇન અક્ષર ઓપરેટિંગ સિસ્ટમ્સ પર એકથી બીજા પર અલગ પડે છે, પરંતુ પાયથોન માં જો આપણે બેકસ્લેશ n નો ઉપયોગ કરીએ છીએ અને તે બધી સિસ્ટમોમાં યોગ્ય રીતે ભાષાંતરિત થશે જે તમે ઉપયોગ કરી રહ્યાં છો. ત્રીજી રીત જે તમે ફાઇલમાંથી વાંચી શકો છો તે શબ્દોની એક સૂચિમાં એક પછી એક જ રેખાઓ વાંચવા માટે છે. તેથી, રેડલાઇનને બદલે, જો હું રેડીલાઇન્સ કહું તો તે સમગ્ર ફાઇલોને ઝાડની સૂચિ તરીકે વાંચે છે. દરેક શબ્દમાળા સૂચિમાં એક વસ્તુ છે અને યાદ રાખો કે આ રેખાઓમાંના દરેક બેકસ્લેશ n સમાવેલ છે. તેથી, વાંચો, વાંચવા અને વાંચતી લીટીઓ, તેમાંના કોઈપણ ખરેખર બેકસ્લેશ n ને દૂર કરશે નહીં. તેઓ તમારા ઇનપુટના ભાગરૂપે વિશ્વાસુ રહેશે. બીજા શબ્દોમાં કહીએ તો, જો તમે આને એક ફાઇલમાંથી બીજામાં સ્થાનાંતરિત કરવા જઈ રહ્યા હોવ, તો તમે બેકસ્લેશ n ને ફરી દાખલ કરવા માટે ચિંતા કરવાની ઇચ્છા નથી કારણ કે આ પહેલેથી જ ત્યાં છે તેથી, તમે આ ઇનપુટ સીધું સીધું ઉપયોગ કરી શકો છો, પરંતુ બીજી બાજુ, જો તમે સ્ટ્રિંગના કેટલાક મેનીપ્યુલેશન કરવા માંગતા હો તો તમને યાદ રાખવું જોઈએ કે આ બેકસ્લેશ n ત્યાં છે અને તમારે તેની સાથે યોગ્ય રીતે કામ કરવું જોઈએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 07:49)

ફાઇલો વાંચવાનું સ્વાભાવિક રીતે ક્રમિક કામગીરી છે. હવે, અલબત્ત, જો આપણે મૂળભૂત આદેશ વાંચીએ, તો તે સમગ્ર સામગ્રી વાંચે છે. તેથી દેખીતી રીતે, તે શરૂઆતથી અંત સુધી વાંચે છે, પરંતુ જો તમે એક સમયે એક રેખા વાંચી રહ્યા છો તો તે જે રીતે કાર્ય કરે છે તે એ છે કે જ્યારે આપણે ફાઇલ ખોલીએ છીએ ત્યારે શરૂઆતમાં ફાઇલની શરૂઆતમાં જ છે. તેથી, તમે આ લાલ તીરની જેમ એક નિર્દેશકની કલ્પના કરી શકો છો જે આગળ જણાવે છે કે આપણે ક્યાં આગળ વાંચીએ છીએ. તેથી, શરૂઆતમાં જ્યારે આપણે ખોલો, અમે શરૂઆતથી વાંચી રહ્યા છીએ, હવે દરેક રીડલાઇન અમને આગળ લઈ જશે. જો હું આ બિંદુ પર એક રેડલાઇન કરીશ તો તે મને આગામી બેકસ્લેશ n સુધી લઈ જશે. યાદ રાખો રેખા એ સંખ્યા છે જે બેકસ્લેશ n દ્વારા સીમાંકિત છે. તેથી, આપણી પાસે એક લીટી હોઈ શકે જે 100 અક્ષરો ધરાવે છે, આગળની લીટીમાં 3 અક્ષરો અને તેથી વધુ હોઈ શકે છે. તે એક બેકસ્લેશ n થી આગળ છે જે એક લીટી છે, તેથી આ એક નિશ્ચિત લિંક નથી. તેથી, અમે બેકસ્લેશ n સુધી ત્યાં સુધી એક અક્ષરને આગળ વધારીશું, પછી બેકસ્લેશ n સુધી બધું રીટલાઇન અને પોઇન્ટર દ્વારા આગળના અક્ષર પર જવા માટે સ્ટ્રિંગ રીફર્ન તરીકે પરત કરવામાં આવશે. હવે, અમે બીજી રીટેક્લીનને કદાચ બીજી લાઇનથી આગળ વધારીએ છીએ. તેથી, આ રીતે આપણે શરૂઆતથી અંત સુધી જઈએ છીએ જો આપણે વાસ્તવમાં વ્યૂહરચનામાંથી કાઢી નાખવા માંગીએ છીએ, તો ત્યાં એક આદેશની જરૂર છે, જે પોઝિશન લે છે, એક પૂર્ણાંક n, અને સીધી જ સ્થાને ખસેડે છે n તમે જ્યાં હોવ તેનાથી. પણ ફરે અથવા એક ફાઇલમાં આસપાસ ફેરવાનું એક રીત છે જે વાક્ય દ્વારા સળંગ વાક્ય વાંચવા કરતાં. છેલ્લે, આપણે વાંચેલ સ્ટેટમેન્ટમાં ફેરફાર કરી શકીએ છીએ, આખી ફાઇલ વાંચવા માટે નહીં, પરંતુ અક્ષરોની ઠીક સંખ્યા વાંચવા માટે. હવે, આ ઉપયોગી થઈ શકે છે જો તમારી અક્ષર વાસ્તવમાં તમારી ફાઇલ ખરેખર ફિક્સ બ્લોકોના ડેટા ધરાવે છે. તેથી, તમે કદાચ કહી શકો છો, ઉદાહરણ તરીકે, પેન નંબરો જે સામાન્ય રીતે 10 અક્ષર લાંબાં હોય છે અને તમે તેમને કોઈ પણ નવા લીટીઓ વગર એક લાંબી ટેક્સ્ટ તરીકે સંગ્રહિત કરી

શકો છો, કેમ કે દરેક પેન નંબર 10 અક્ષરો છે. તેથી, જો આપણે કહીએ એકએચ ડોટ 10 વાંચ્યું છે, તો તે આગામી પેન નંબર વાંચશે અને ચાલુ રાખશે અને આ તમને લાંબા ગાળે અમુક જગ્યા બચાવે છે. તેથી, એવી પરિસ્થિતિઓ છે કે જ્યાં તમે આનો ઉપયોગ કરી શકો છો જ્યાં તમે અક્ષરોની ઠીક સંખ્યા વાંચી શકો છો.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 09:56)

જ્યારે આપણે ફાઇલને વધારી રૂપે વાંચી રહ્યા છીએ, ફાઇલ ક્યારે સમાપ્ત થઈ જાય છે તે જાણવા માટે ઉપયોગી છે કારણ કે આપણને અગાઉથી ખબર નથી પડતી કે કેટલા ફાઇલો છે અથવા કેટલા લીટીઓ ફાઇલ છે તેથી, જો તમે વાક્ય દ્વારા ફાઇલ રેખા વાંચી રહ્યા હોવ તો પછી આપણે જાણી શકીએ કે ફાઇલ ક્યારે સમાપ્ત થઈ ગઈ છે. તેથી, ત્યાં બે પરિસ્થિતિઓ છે જ્યાં આપણે આને જાણશું. તેથી, એક એ છે કે આપણે વાંચેલા આદેશનો ઉપયોગ કરીને વાંચવાનો પ્રયત્ન કરીએ છીએ અને આપણને કંઈપણ પાછું મળી નથી, આપણને એક ખાલી શબ્દ મળે છે જેનો અર્થ એ છે કે ફાઇલ સમાપ્ત થઈ ગઈ છે, આપણે ફાઇલનો અંત સુધી પહોંચી ગયા છીએ. એ જ રીતે, જો આપણે રેખા વાંચવા પ્રયત્ન કરીએ છીએ અને આપણને ખાલી શબ્દ મળે છે તો તેનો અર્થ એ કે આપણે ફાઇલના અંત સુધી પહોંચી ગયા છીએ. તેથી, જો તે ખાલી શબ્દ પાછો આવે તો વાંચો અથવા વાંચી લો તેનો અર્થ એ છે કે આપણે ફાઇલના અંત સુધી પહોંચી ગયા છીએ. યાદ રાખો, અમે શરૂઆતથી અંત સુધી અનુક્રમે જઈ રહ્યા છીએ તેથી, અમે ફાઇલના અંત સુધી પહોંચી ગયા છીએ અને આ ફાઇલમાં વાંચવા માટે વધુ કંઈ નથી

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 10:44)

ફાઇલમાંથી વાંચ્યા પછી બીજી વસ્તુ જે આપણે કરવા માંગીએ છીએ તે ફાઇલમાં લખી છે. તેથી, અહીં તમે અહીં ફાઇલ લખી શકો છો જેમ કે તમે આદેશ વાંચી લીધો છે તમારી પાસે લખવાની આજ્ઞા છે, પરંતુ હવે તે વાંચવાથી વિપરીત છે કે ફાઇલ પર કંઈક અનિચ્છનીય રીતે લઈ જાય છે અને તમને આપે છે, અહીં તમારે તેને ફાઇલમાં મૂકેલ છે. તેથી, લખો એક દલીલ લે છે જે શબ્દમાળા છે. જ્યારે તમે કહો છો, લખી લો, તો શબ્દમાળાને લો અને ફાઇલમાં લખો. હવે, બે વસ્તુઓ છે; એક એ આ છે કે બેકસ્લેશ n હોઈ શકે નહીં, તેમાં એક કરતાં વધુ બેકસ્લેશ n હોઈ શકે છે. તો, કશું જ નથી, આ તમને કહે છે કે આ એક લીટીની એક લીટી ભાગ છે જે તમને ફાઇલ પર લખેલું રસ્તો મુજબ તમારે લખવું પડશે, જો તમે તેને એક લીટીમાં રાખવા માંગો છો તો તમારે ખાતરી કરવી જોઈએ તે બેકસ્લેશ n સાથે અંત થાય છે. અને આ લખી ખરેખર લખેલા અક્ષરોની સંખ્યા આપે છે. હમણાં, આ કરવા માટે એક વિચિત્ર વસ્તુ જેવી લાગે છે, શા માટે તે તમને કહી કારણ કે તમે અક્ષરની સંખ્યા શું છે લખાવેલ છે, પરંતુ આ ઉપયોગી છે જો, ઉદાહરણ તરીકે, ડિસ્ક ભરેલી છે. જો તમે લાંબા સ્ટ્રિંગ લખવાનો પ્રયત્ન કરો અને શબ્દનો માત્ર ભાગ લખો તે લખેલું છે અને આ સંકેત છે કે ત્યાં લખવા સાથે સમસ્યા હતી. તેથી, કેટલીક વાર જાણવા માટે કે જે અક્ષરો લખવાનો પ્રયત્ન કરે છે તેમાંથી કેટલા અક્ષરો ખરેખર લખાયા છે. અન્ય વસ્તુ જે ફાઇલમાં મોટા પ્રમાણમાં લખે છે તે રીટેલલાઈન કહેવાય છે. તેથી, આ શબ્દોની યાદી લે છે અને તેને એક પછી એક ફાઇલમાં લખી છે. હવે છતાં તે લખી લીટીઓ કહે છે કે આ વાસ્તવમાં લીટીઓ નથી. તેથી, જો તમે તેમને રેખાઓ કરવા માંગતા હો તો તેનું નામ ગેરમાર્ગે દોરે છે, તમારે ખાતરી કરવી પડશે કે તમારી પાસે દરેક બેકસ્લેશ n દ્વારા સમાપ્ત થયેલ છે. જો તે ન હોય તો તેઓ માત્ર એક લાંબી લાઇનની વસ્તુ બનાવવા માટે કાસ્કેડ કરશે. તેથી, જો કે તે રીટેઈલિનને કહે છે તે શબ્દોની યાદી લખવી જોઈએ, તે જોઈએ, તે આ કાર્ય માટે વધુ યોગ્ય નામ છે, તે માત્ર શબ્દમાળાઓની સૂચિ લે છે અને ફાઇલ હેન્ડલ દ્વારા નિર્દેશિત ફાઇલમાં તેને લખે છે.

(સ્લાઈડ ટાઇમ નો ઉલ્લેખ કરો: 12:40)

અને છેલ્લે, આપણે એકવાર એક ફાઇલ સાથે પૂર્ણ કરી લીધા પછી, આપણે તેને બંધ કરવું પડશે અને ખાતરી કરો કે ફાઇલ સાથે સંકળાયેલ બંધો છે, ખાસ કરીને જો તમે કોઈ ફાઇલમાં લખી રહ્યા હોવ તો તે ફલશ છે તેથી, એફએચ ડોટ બંધ, ફાઇલ હેન્ડલ એફએચ બંધ કરશે અને આ બિંદુએ તમામ બાકી લખો ડિસ્કમાં કોપિ કરેલા છે. તેનો અર્થ એવો પણ થાય છે કે એફએચ ફાઇલ સાથે સંકળાયેલ નથી જે અમે સાથે વ્યવહાર કરી રહ્યા છીએ. તેથી, આ પછી જો આપણે એફએચ પર કામગીરી શરૂ કરવાનો પ્રયાસ કરીએ તો તે પાયથોન માં અવ્યાખ્યાયિત નામ હોવા જેવા છે. હવે, કેટલીક વખત એવા પરિસ્થિતિઓ છે કે જ્યાં બંધરને ફાઇલ બંધ કર્યા વિના ફલશ કરવું જોઈએ. અમે ફક્ત ખાતરી કરવા માંગીએ છીએ કે આ બિંદુ સુધીના તમામ લખાણો વાસ્તવમાં ડિસ્ક પર પ્રતિબિંબિત થયા છે. તેથી, એક આદેશ ફલશ છે જે આ કરે છે. જો આપણે ફલશ કહીએ છીએ, તો તે માત્ર એટલું જ કહે છે કે કોઈ બાકી લેખો છે તો કૃપા કરીને તેમને તમામ ડિસ્ક પર મૂકી દો, જોખમોની રાહ જોતા નથી જ્યાં સુધી બંધર પૂર્ણ ન થાય ત્યાં સુધી રાહ જુઓ અને પછી તમે જે રીતે સામાન્ય રીતે ડિસ્ક પર હોવ તે લખો.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 13:42)

અહીં એક સામાન્ય વસ્તુ છે કે તમે પાયથોન માં કરવા માંગો છો, જે વાક્ય દ્વારા તે પ્રક્રિયા કરવા માટે છે. આ કરવા માટેની કુદરતી રીત એ લીટીઓ એક સૂચિમાં વાંચવી અને ત્યારબાદ સૂચિબદ્ધ સૂચિ પર પ્રક્રિયા કરવી. તેથી, તમે કહો કે સામગ્રી એ એફએચ ડોટલાઇન્સ છે અને પછી દરેક લાઇન અને વિષયવસ્તુ માટે તમે તેની સાથે કંઈક કરો છો. તમે વાસ્તવમાં આને વધુ સઘન રીતે કરી શકો છો, તમે નામ સામગ્રીઓથી દૂર કરી શકો છો અને કાર્ય ફંક્શન ડીએટ ટાઇપલાઇન્સ દ્વારા સીધી દરેક લીટી રીટર્ન વાંચો. તેથી, આ એક જ લૂપની સમાન રચના છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 14:25)

ઉદાહરણ તરીકે, લીટી પ્રોસેસિંગ દ્વારા આ લાઇનનો ઉપયોગ કેવી રીતે કરવો, ચાલો આપણે કલ્પના કરીએ કે આપણે ફાઇલ input dot txt ફાઇલના સમાવિષ્ટોને ફાઇલ આઉટપુટ ડોટ ટેક્સ્ટમાં કોપી કરવી છે. તેથી, આપણે જે પ્રથમ વસ્તુ કરવાની જરૂર છે તે એ છે કે આપણે તેને યોગ્ય રીતે ખોલીએ. આપણે ખરેખર 'w' મોડ સાથે ફાઇલ ફાઇલ ખોલવું જોઈએ અને ફાઇલ મોડ 't' માં. આ કહે છે કે હું ઈન્ફાઇલમાંથી વાંચીશ અને આઉટફાઇલ લખીશ. હવે, ઈન્ફાઇલ પરની રીડલાઇન દ્વારા પરત આવતા દરેક લીટી માટે, યાદ રાખો કે જ્યારે હું રેખલાઈનથી રેખાચિત્ર મેળવતો હોઉં ત્યાં બેકસ્લેશ n પહેલેથી જ છે, જો હું બેકસ્લેશ n ને કાંઈ ન કરું, તો હું તેને બરાબર એ જ રીતે લખી શકું છું. તેથી, દરેક લીટી માટે જે મેં યાદીમાંથી વાંચી છે, ઇન્ફિલ ડોટ ટાઇપલાઇન્સ હું તેને માત્ર આઉટફાઇલમાં લખી અને છેવટે, હું બન્ને ફાઇલોને બંધ કરું છું. ઇનપુટથી આઉટપુટમાંથી એક ફાઇલને કોપિ કરવાનો આ એક રીત છે.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 15:25)

અલબત્ત, આપણે તેને એક જ શોટમાં પણ કરી શકીએ છીએ કારણ કે આદેશ લખાય છે, જે શબ્દમાળાઓની યાદી લે છે અને તેમને એક શોટમાં લખે છે. તેથી, લીસ્ટની લીટી દ્વારા લીટીમાં જવાની બદલે આપણે સમગ્ર સૂચિ વિષયવસ્તુ લઈ શકીએ છીએ અને માત્ર રીટેઈલની મદદથી સીધા જ આઉટપુટ કરી શકીએ છીએ, આ એક વૈકલ્પિક રીત છે જ્યાં મેં સ્થાને લીધું છે. આ મૂળભૂત રીતે માટે બદલી છે. તેથી, ઇન્ફાઈલ્સમાં દરેક લાઇન માટે કહીને બદલે હું તેને સીધી રીતે લખી શકું છું.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 16:02)

આપણે જે વસ્તુઓ વિશે વાત કરી રહ્યા છીએ તે આ નવું રેખા પાત્ર છે જે થોડી ચીડ છે જો આપણે કોઈ નવી લીટી પાત્ર સાથે મેળવવા માંગીએ છીએ, તો યાદ રાખો કે આ ફક્ત એક શબ્દમાળા છે અને નવા વાક્ય અક્ષર આ શબ્દમાળામાં છેલ્લો અક્ષર બનશે. તેથી, મેળવવાનો એક રસ્તો એ છે કે શબ્દમાળાનો ટુકડો ખાલી કરવા માટે, પરંતુ છેલ્લો અક્ષર શામેલ નથી. હવે, યાદ રાખો કે જ્યારે આપણે ગણતરી કરીએ છીએ કે પાછલું બાદ 1 એ છેલ્લો અક્ષર છે. જો આપણે સ્લાઇડસને 0 થી લઈને બાદ 1 સુધી લઈશું તો તે યોગ્ય રીતે છેલ્લા અક્ષરને સ્ટ્રિંગથી બાકાત કરશે. તેથી, રેખા કોલોન બાદ 1 બરાબર છે, લીટી અને છેલ્લા પાત્રની સ્ટ્રીપ લેશે જે સામાન્ય રીતે બેકસ્લેશ એન છે જે આપણે મેળવીએ છીએ, જ્યારે આપણે લીટીલાઇન્સ કરીએ છીએ. હવે, સામાન્ય રીતે અમારી પાસે અન્ય જગ્યાઓ હોઈ શકે છે તેથી, યાદ રાખો કે જ્યારે તમે ઘણીવાર ટેક્સ્ટને લખો છો ત્યારે આપણે રેખાના અંતની જગ્યાઓ જોઈ શકતા નથી કારણ કે તે અમને અદ્રશ્ય છે. આ તે સફેદ જગ્યા કહેવાય છે. તેથી જગ્યાઓ, ટેબ્સ, નવી લીટીઓ, તે અક્ષરો છે જે સ્ક્રીન પર પ્રદર્શિત થતા નથી, ખાસ કરીને જગ્યાઓ અને ટેબ્સ અને ત્યાં લીટીના અંતે, આપણે જાણતા નથી કે લીટી એ છેલ્લો અક્ષર સાથે અંત થાય છે જે આપણે જોશું કે શબ્દો પછી જગ્યા છે. તેથી, r સ્ટ્રિટ એ સ્ટ્રિંગ કમાન્ડ છે જે ખરેખર સ્ટ્રિંગ લે છે અને પાછળની સફેદ જગ્યાને દૂર કરે છે, બધી સફેદ જગ્યાઓ લીટીના અંતે છે. ખાસ કરીને માત્ર એક બેકસ્લેશ n છે અને તે બેકસ્લેશ n માં છીનવી લેશે. તે અન્ય ફૂદકામાં ફૂદકા કરે છે ત્યાં બેકસ્લેશ n પહેલાં કેટલાક જગ્યાઓ અને ટેબ્સ છે અને તે પરત કરે છે. તેથી, રેખાના ડોટ રે સ્ટ્રિટની સમાન છે જે સફેદ જગ્યાની જમણી બાજુથી સ્ટ્રીપ લીટી છે. આ અગાઉના લીટીમાં સમાન વસ્તુ છે સિવાય કે તે વધુ સામાન્ય છે કારણ કે તમામ સફેદ જગ્યાને છેલ્લા બેકસ્લેશ n દ્વારા નહીં, પરંતુ લીટીના તમામ સફેદ જગ્યાઓ અંત. અમે એલ સ્ટ્રીપનો ઉપયોગ કરીને ડાબેથી પણ છીનવી શકીએ છીએ અથવા જો આપણે કોઈ પણ પાત્રાલેખન । અથવા r સિવાય સ્ટ્રીપ કહીએ તો બંને બાજુએ છીનવી શકીએ છીએ. આ સ્ટ્રિટ મેનીપ્યુલેશન ફંક્શન્સ છે અને અમે તેમાંના કેટલાકને જોશું, પરંતુ આ ફક્ત ઉપયોગી છે જે ફાઇલ હેન્ડલિંગના સંદર્ભમાં તરત જ આવે છે. તેથી, આગળ વધતાં પહેલાં આપણે આ બધી વસ્તુઓની કેટલીક ઉદાહરણો જોઈએ જે આપણે અત્યાર સુધી જોયેલી છે.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 18:05)

આપણે ઇનપુટ ડોટ નામની એક ફાઇલ બનાવી છે જે એક રેખા ધરાવે છે, તો ઝડપી બ્રાઉન શિયાળ આક્રમક ફૂતરા પર ફૂદકા કરે છે. હવે, ચાલો પાચથોન ને ઇન્ટરપ્રીટર ખોલીએ અને આ ફાઇલની લીટીઓ વાંચવાનો પ્રયત્ન કરીએ અને તેને છાપીએ. તેથી, આપણે કહી શકીએ, દાખલા તરીકે, એક વાંચવા માટે input dot txt ને ખોલવા બરાબર છે. હમણાં, મેં ફાઇલ ખોલી છે અને હવે હું કહી શકું છું, ઉદાહરણ તરીકે, f ડોટ ટાઇડલાઇન પ્રિન્ટ લાઇનમાં લીટી માટે. હવે, તમે અહીં જે કંઈક રસપ્રદ બનશે તે જોશો, તમે જોશો કે હવે આપણી ફાઇલની દરેક લાઇન વચ્ચે એક ખાલી લીટી છે. હવે, શા માટે અમારી ફાઇલમાં દરેક લીટી વચ્ચે ખાલી લીટીઓ શા માટે છે? કારણ કે જ્યારે આપણે વાંચી લીધું છે ત્યારે આપણે રેખાથી બેકસ્લેશ n અક્ષર મેળવીએ છીએ તેથી, ઝડપી ભુરો, બેકસ્લેશ n સાથે પ્રથમ રેખા અંત, શિયાળ બેકસ્લેશ n સાથે અને ત્યારબાદ ઉપર અને ઉપર છે જો તમને યાદ છે કે પ્રિન્ટ સ્ટેટમેન્ટ તેના પોતાના બેકસ્લેશ n ને ઉમેરે છે. તેથી, વાસ્તવમાં પ્રિન્ટ આમાંના દરેક માટે ખાલી લીટીઓ પર મૂકે છે. હવે, ચાલો ફરીથી પ્રયાસ કરીએ અને આ પ્રમાણે કરીએ, હું આ વસ્તુને પુનરાવર્તન કરું છું અને હવે હું આ ફરીથી કરું છું, હવે કંઈ પણ થાય નથી કારણ કે કંઈ નથી કારણ કે અમારી પાસે આ ક્રમાંકિત વાંચન ફાઇલ છે. તેથી, પ્રથમ વખત આપણે ડોટ ટાઇપલાઇન કર્યું છે, તે એક સમયે એક લીટી વાંચે છે અને હવે આપણે વાસ્તવમાં ફાઇલના અંત તરફ સંકેત આપી રહ્યા છીએ.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 19:49)

દાખલા તરીકે જો આ બિંદુએ, આપણે કહીએ છીએ કે ટેક્સ્ટને એફએચ ડોટ વાંચવા માટે, માફ કરશો ડોટ ડોટ વાંચી શકાશે, તો ટેક્સ્ટ ખાલી સ્ટ્રીંગ હશે. આ એ સંકેત છે કે આપણે વાસ્તવમાં ફાઇલના અંત સુધી

પહોંચી ગયા છીએ. એ જ રીતે, જો આપણે ફરીથી રેડલાઇન કહેવાનો પ્રયાસ કરીએ તો ટેક્સ્ટ ખાલી સ્ટ્રીંગ હશે. તેથી, યાદ રાખો કે આપણે કહ્યું છે કે જો વાંચી અથવા વાંચી લીટી ખાલી શબ્દ આપે તો આપણે ફાઇલના અંત સુધી પહોંચી ગયા છીએ. આ એક જ રસ્તો આપણે આને પૂર્વવત્ કરી શકીએ છીએ તે છે કે ફાઇલો બંધ કરીને ફરીથી શરૂ કરો. તેથી, આપણે શું કહીએ છીએ તે F આ ફાઇલ બંધ. હવે, જો તમે Fot વાંચવા માટે પ્રયત્ન કરો તો આપણે કહી શકીએ છીએ કે આ વ્યાખ્યાયિત કરવામાં આવી રહ્યું નથી. તેથી, હવે અમારી સાથે f નથી. તેથી, ફરીથી આપણે કહીએ છીએ f ખુલ્લું input dot txt r છે અને હવે આપણે દરેક લીટી માટે f ડોટ ટાઇપલાઇન્સમાં લીટી માટે કહી શકીએ છીએ. ધારો કે, હવે અમે તે યુક્તિનો ઉપયોગ કર્યો છે જે અમારી પાસે છેલ્લી વખત છે, જે કહે છે કે અંતિમ સ્ટ્રિંગની સમાન છે જે દરેક પ્રિન્ટ સ્ટેટમેન્ટ પછી કંઈપણ શામેલ ન કરે. હવે, જો તમે આ કરો તો તમે જુઓ છો કે આપણે ચોક્કસ ઇનપુટ ફાઇલો પર પાછા આવીએ છીએ કારણ કે તે વધારાની ખાલી લીટીઓ વગર છે કારણ કે પ્રિન્ટ હવે આ વધારાની રેખાઓ બનાવતી નથી.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 21:10)

ચાલો આપણે ઇનપુટ ડોટ ટેક્સ્ટને આઉટપુટ ડોટ ટેક્સ્ટમાં કોપી કરવા ઈચ્છીએ, આપણે કહીએ છીએ એફ એ પહેલા input.pt.txt ખોલવા બરાબર છે અને આપણે કહીએ છીએ કે output is open dot txt w બરાબર છે. અને હવે આપણે f ડોટ ટ્રીલાઇન્સમાં લીટી માટે કહીએ છીએ, g dot write લાઈન. હવે, નોંધ લો કે તમને સંખ્યાઓનો ક્રમ મળે છે શા માટે આપણે સંખ્યાઓનું અનુક્રમ મેળવીએ છીએ, કારણ કે દરેક વખતે આપણે કંઈક લખીએ તો તે સંખ્યાબંધ અક્ષર લખે છે અને તે ચાલુ થશે, જો તમે લીટીઓ ઝડપી બ્રાઉન લક્ષ, વગેરે વગેરે જોશો, ઉદાહરણ તરીકે, બીજી લાઈન ફક્ત શિયાળ છે, શિયાળને ત્રણ અક્ષરો છે, પરંતુ જો તમે બેકસ્લેશ n નો સમાવેશ કરો તો તેને 4 અક્ષરો લખ્યા છે. એટલા માટે ઝડપી બ્રાઉન 15 અક્ષરો વત્તા બેકસ્લેશ n, શિયાળ 3 વત્તા બેકસ્લેશ n હતી. તેથી, આ વાક્ય દ્વારા રેખા છે હવે, જો હું આ ફાઇલો યોગ્ય રીતે બંધ કરીશ તો આમાંથી બહાર નીકળો, પછી output dot txt બરાબર ઇનપુટ dot txt જેવું જ છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 22:26)

આપણે શું જોયું છે તે સારાંશ આપવા માટે, જો તમે ફાઇલો સાથે ક્રિયાપ્રતિક્રિયા કરવા માગીએ છીએ તો અમે તે ફાઇલ હેન્ડલ્સ દ્વારા કરી શકીએ છીએ, જે વાસ્તવમાં બફરોને અનુરૂપ છે જેનો ઉપયોગ અમે મેમરી અને ફાઇલ વચ્ચે ક્રિયાપ્રતિક્રિયા કરવા માટે કરીએ છીએ. ડિસ્ક આપણે એક ફાઇલને ત્રણમાંથી એક મોડમાં ખોલી શકીએ; વાંચો, લખો અને ઉમેરો અમે વાસ્તવમાં એપેન્ડ સાથે કોઈ ઉદાહરણ કર્યું નથી, પરંતુ અમે તે શું ઉમેરશે તે ઉમેરો કરીશું, ફાઇલ પહેલાથી જ અસ્તિત્વમાં છે ત્યાંથી આગળ લખવાનું રહેશે, અન્યથા લખશે ફાઇલ ભૂંસી નાખશે અને શરૂઆતથી શરૂ થશે અમે જોયું કે આ વાંચવા, વાંચવા અને વાંચતી લીટીઓ, આનો ઉપયોગ કરીને આપણે સમગ્ર ફાઇલ સ્ટ્રિંગના એક શોટમાં વાંચી શકીએ છીએ અથવા રેખા દ્વારા તે રેખા વાંચી શકીએ છીએ. એ જ રીતે, આપણે કાં તો શબ્દમાળા લખી શકીએ છીએ અથવા આપણે શબ્દોની યાદી પણ લખી શકીએ છીએ. તેથી, અમારી પાસે રીટેઈલેઇન્સમાં લખવાની આદેશ છે અને રીટેઈલિંગ્સ વધુ સારી રીતે યોગ્ય છે જેથી દુભાષીઓ શબ્દોની યાદી લખી શકે. છેલ્લે, આપણે હેન્ડલને બંધ કરી શકીએ છીએ જ્યારે આપણે ફ્લશ કમાન્ડનો ઉપયોગ કરીને બફરને ફ્લશ કરી શકીએ છીએ અને આપણે જોયું કે સ્ટ્રીમ ઓપરેશન માટે સફેદ જગ્યા છીનવાઈ છે અને આ પાછળના બેકસ્લેશ n ને દૂર કરવા માટે ઉપયોગી છે. જ્યારે પણ તમે ટેક્સ્ટ ફાઇલોની પ્રક્રિયા કરી રહ્યા હો ત્યારે આવો.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રો. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 05 લેક્ચર - 04 સ્ટ્રિંગ પ્રોસેસિંગ

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 00:02)

છેલ્લો વ્યાખ્યાન અમે વાંચ્યું છે લખાણ ફાઇલો લખો અને લખાણ વાંચવું અને લખવું અચૂકપણે શબ્દમાળાઓ પર પ્રક્રિયા કરે છે જે આપણે વાંચીએ છીએ અને લખીએ છીએ. અને તેથી, પાયથોન પાસે સંખ્યાબંધ સ્ટ્રિંગ પ્રોસેસિંગ ફંક્શન્સ છે જે આ સામગ્રીને સંશોધિત કરવાનું સરળ બનાવે છે. તેથી સામાન્ય રીતે, તમે આ ફાઇલો સાથે કંઈક કરવા માટે ફાઇલો વાંચી અને લખી રહ્યાં છો, અને આ સાથે કંઈક કરવા માટે તમે શબ્દમાળા કાર્યોમાં ઉપયોગ કરી શકો છો જે તદ્દન શક્તિશાળી છે. અન્ય વસ્તુઓ પૈકી, તમે આ શબ્દમાળા કાર્યો સાથે શું કરી શકો છો ઉદાહરણ તરીકે, ટેક્સ્ટ માટે શોધ કરો અથવા શોધ કરો અને તેને બદલો. ફાઇલ માટે સ્ટ્રિંગ પ્રોસેસિંગનો એક લાક્ષણિક ઉપયોગ એ છે કે જ્યારે આપણે કોઈ સ્પેડશીટ જેવું કંઈક લઈએ છીએ અને તેને ટેક્સ્ટ તરીકે નિકાસ કરીએ છીએ. અલ્પવિરામથી વિભાજિત મુલ્ય ફોર્મેટ CSV નામની કોઈ વસ્તુ છે, જ્યાં કોલમ્સ ટેક્સ્ટ તરીકે અલ્પવિરામ દ્વારા અલગ થયેલ છે. હવે, આપણે સ્ટ્રિંગ ફાઇલ સાથે શું કરી શકીએ તે છે કે રેખા દ્વારા આવી ફાઇલ લાઇન વાંચવી અને દરેક લાઇનમાં અલ્પવિરામ વચ્ચે વાંચીને ટેક્સ્ટની કોલમ કાઢવા. તો, આ લેક્ચરમાં આપણે આ બધું જોશું.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 01:04)

સ્ટ્રિંગ કમાન્ડનું પહેલું ઉદાહરણ કે જે આપણે છેલ્લે જોયું છે તે ખાલી જગ્યા ખાલી કરવા માટે આદેશો છે. અમારી પાસે રસ્તાની રેફ્રેશ છે, જે અમે ઉદાહરણ તરીકે ઉપયોગમાં લીધેલ સફેદજગ્યાના બેકસ્લેશ n ને અમારા રેખાઓમાંથી દૂર કરવા માટે વાપરી છે, અને અમારી પાસે અગ્રણી સફેદજગ્યાને દૂર કરવા માટે ફૂદકો લગાવ્યો હતો, અને અમારી પાસે સ્ટ્રીપ છે જે તેને બંને દિશામાં દૂર કરે છે. ચાલો જોઈએ કે આ કેવી રીતે કામ કરે છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 01:30)

ચાલો એક સ્ટ્રિંગ બનાવીએ જે પહેલા અને પછીથી સફેદજગ્યાના છે, તેથી ચાલો આપણે કેટલીક જગ્યાઓને ટેબ બનાવીએ અને પછી હેલ્લો શબ્દ અને પછી બે ટેબ્સ. અમારી પાસે એક સ્ટ્રિંગ છે જે સફેદજગ્યા છે અને તમે ટેબ્સને બેકસ્લેશ ટી અને બ્લેન્ક્સ દ્વારા દર્શાવેલ છે તે જોઈ શકો છો. હવે, જો આપણે માત્ર જમણાથી રદ કરવા ઈચ્છીએ છીએ તો આપણે કહી શકીએ છીએ કે t બરાબર s dot rstrip છે. યાદ રાખો આ સ્ટ્રીપ કમાન્ડ સ્ટ્રીંગ્સ અવિરત અધિકાર છે, તે તે બદલશે નહીં, તે નવી સ્ટ્રિંગ પાછી આપશે, જો હું કહું કે ટી એ ડોટ આર સ્ટ્રીપ છે તો તે જમણી બાજુ સફેદ જગ્યા પર છીનવીશ અને મને ટી આપો, જો હું ટી જુઓ તે બધા સુધી હેલ્લો સુધી છે પરંતુ તે ટેબ નથી અને પછીથી જગ્યા. એ જ રીતે, જો હું કહું કે ટોલ ડોટ લી સ્ટ્રીપ છે તો તે ડાબી બાજુથી દૂર કરશે, હવે હેલ્લો સાથે શરૂ થશે, પરંતુ અંતે તે સફેદ જગ્યા વાંચશે. છેલ્લે, જો હું કહું કે ટી એ ડોટ સ્ટ્રીપ છે તો બન્ને પક્ષો બગડ્યા છે અને મને તે શબ્દ મળશે જે મને ઈચ્છે છે. આ ઉપયોગી છે કારણ કે જ્યારે તમે લોકોને વસ્તુઓ અને સ્વરૂપો ટાઈપ કરવા માટે પૂછો ઉદાહરણ તરીકે, સામાન્ય રીતે જો તેઓ પહેલાં અને પછી કેટલાક ખાલી જગ્યા છોડે છે, તેથી જો તમે પહેલા ખાલી થવા પહેલાં બધું જ કરવા માંગો છો અને છેલ્લું ખાલી માત્ર પછી વચ્ચે લખાણ રાખો

lstrip, rstrip સંયોજનનો ઉપયોગ કરી શકો છો અથવા ફક્ત વાસ્તવિક ડેટાને કાઢવા માટે સ્ટ્રિટ કરી શકો છો કે જે તમે ફાઇલમાંથી ઇચ્છો છો.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 02:43)

આગામી વસ્તુ જે આપણે કરવા માગીએ છીએ તે શબ્દમાળામાં ટેક્સ્ટ જોવાનું છે. શોધવા માટેની મૂળભૂત આદેશ છે તેથી, જો શબ્દમાળા અને પેટર્ન અન્ય શબ્દમાળા છે જે હું s માં શોધી રહ્યો છું, તો ડોટ શોધ પેટર્ન s માં પ્રથમ સ્થાન આપશે જે પેટર્ન ઉત્પન્ન થાય છે. અને જો પેટર્ન ઉત્પન્ન થતું નથી, તેથી સ્થાનો ત્યાં ચોક્કસપણે 0 અને s min length 1 ની વચ્ચે હશે. અમે પહેલાથી જ આ પહેલાંના આપણા પોતાના અમલીકરણને લખી છે. તેથી, જો તે ન થતું હોય તો તે તમને બાદબાકી આપશે. કેટલીક વખત તમે સંપૂર્ણ સ્ટ્રિંગ શોધી શકતા નથી, તેથી પેટર્ન એક વૈકલ્પિક દલીલને શરૂ કરે છે અને અંત થાય છે, જેમાં તે સંપૂર્ણ સ્ટ્રિંગમાંથી પેટર્ન શોધીને બદલે દેખાય છે સામાન્ય સંમેલનથી શરૂઆતથી અંત સુધી એક સ્લાઇડ પર કે આ શૂન્યથી શરૂ થવાની શરૂઆત છે. આ આદેશનું બીજું સંસ્કરણ ઇન્ડેક્સ કહેવાય છે. શોધ અને અનુક્રમણિકા વચ્ચેનું તફાવત એ છે કે જ્યારે પેટર્ન ન મળે તો શું થાય છે. પેટર્ન ન મળે ત્યારે શોધી કાઢો ત્યારે તમને અનુક્રમણિકામાં એક બાદબાકી 1 મળશે, જ્યારે પેટર્ન ન મળે તો તમને આ કિસ્સામાં એક વિશિષ્ટ પ્રકારની ભૂલ મળે છે જેમાં મૂલ્યની ભૂલ છે. તો ચાલો આપણે જોઈએ કે આ વસ્તુઓ ખરેખર કેવી રીતે કાર્ય કરે છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 03:54)

તેથી, આપણી પાસે અહીં એક સ્ટ્રિંગ છે જેમાં શબ્દ "બ્રાઉન ફોક્સ ગ્રે ડોર બ્રાઉન ફોક્સ" છે. હવે જો હું તેને "બ્રાઉન" શબ્દની પ્રથમ ઘટના શોધવા માટે કહીશ, તો તે પદ 0 ને પાછું આપશે કારણ કે તે સ્ટ્રિંગની શરૂઆતમાં બરાબર છે. જો બીજો બાજુ હું આ સ્થિતિ નથી ઇચ્છતો, પણ હું માનું છું કે પોઝિશન 5 થી શરૂ કરવું અને ઉદાહરણ તરીકે ઓની લંબાઈ જવાનું છે, તો તે 19 કહેશે અને જો તમે ગણતરી કરશો તો તમને મળશે કે ભૂરા રંગની બીજી ઘટના છે પોઝિશન 19. જો બીજો બાજુ હું કંઈક કે જે "બિલાડી" જેવું નથી ત્યાં જોવા મળશે તો શોધવા માઇનસ 1 આપશે, તેથી ઓછા 1 એ ભૂલ નથી પરંતુ સંકેત છે કે શબ્દમાળા મળી નથી. અનુક્રમણિકામાં તફાવત એ છે કે જો હું ઇન્ડેક્સને બાદ 1 ની જગ્યાએ એક જ વસ્તુ આપીશ તો મને એક મૂલ્યની ભૂલ મળે છે જે કહે છે કે સબસ્ટ્રિંગ યોગ્ય રીતે થતું નથી તે આ છે કે કેવી રીતે શોધ અને ઇન્ડેક્સ કામ કરે છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 04:53)

શોધ બાદ અને પછી સ્થાનાંતરિત પછીની કુદરતી વસ્તુ. જો હું કંઈક બદલવા માંગું છું તો હું તેને બે સ્ટ્રિંગ્સ આપીશ જે હું શોધી રહ્યો છું અને હું તેને બદલી રહ્યો છું અને તે પ્રથમ શબ્દમાળાની દરેક ઘટના સાથે બીજા શબ્દમાળા દ્વારા બદલાશે. હવે આને નીચેની રીતે નિયંત્રિત કરી શકાય છે; ધારો કે, હું દરેક ઘટના માટે નથી માંગતા, પરંતુ હું માત્ર પ્રથમ ઘટના અથવા પ્રથમ ત્રણ ઘટનાઓ કહેવું માંગો છો. તેથી, હું તેને વૈકલ્પિક દલીલ આપીને કહું છું કે શરૂઆતથી શરૂ થતી આવી ઘણી ઘટનાઓને બદલવી જોઈએ. તે કહે છે કે મોટાભાગની પ્રથમ એન કોપિઓને બદલો અને નોંધો કે ગમે અને સ્ટ્રીપ કરો અને તે બધું અહીં છે, કારણ કે કંઈક બદલીને આ શબ્દ બદલી રહ્યો છે, તે એ બદલાશે નથી કારણ કે શબ્દમાળાઓ અસંબદ્ધ છે, અમને પરિવર્તન પાછું લાવવાનું છે. શબ્દમાળા તો ચાલો એક ઉદાહરણ જોઈએ.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 05:45)

ફરી એક વાર આપણે આપણા જૂના ઉદાહરણને જોઈએ; એ "ભુરો શિયાળ ગે ફૂતરો ભુરો શિયાળ" છે અને હવે હું "કાળો" દ્વારા "ભુરો" બદલવા માંગું છું, પછી મને 'બ્લેક ફોક્સ ગે ડોગ બ્લેક ફોક્સ' મળે છે. જો હું કહું કે માત્ર 1 ને બદલવાની જરૂર છે તો મને 'બ્લેક ફોક્સ ગે ડોગ' મળે છે જ્યાં બીજું ભુરો યથાવત રહે છે. હવે જો તમે આ પેટર્ન ધરાવો છો તો શું થશે તે પૂછી શકો છો, જો મારી પાસે ભુરો ઓવરલેપની જુદી જુદી નકલો હોય તો તે સરસ રીતે વિભાજિત થતી નથી. ધારો કે મારી પાસે "અબાબા" જેવી મૂર્ખ સ્ટ્રિંગ છે અને હવે હું કહું છું કે "ડીડી" દ્વારા "એબીએ" ને બદલો. હવે પ્રશ્ન એ છે કે, તે બે એબીએ અથવા 1 એબીએ મળશે, કારણ કે ત્યાં એબીએ સ્થાને 0 થી શરૂ થાય છે, ત્યાં બીજો સ્થાને એબીએ પણ છે જે સ્થાને 2 થી શરૂ થાય છે. પ્રશ્ન એ છે કે તે બંનેનો ચિહ્નિત થશે આ અને તેમને ડીડી દ્વારા બદલવું, સારું, તે નથી કારણ કે તે ક્રમશઃ કરે છે તેથી તે પ્રથમ પ્રથમ એબા લે છે, તેને ડીડી દ્વારા બદલે છે, આ બિંદુએ બીજા એબીએ નાશ કરવામાં આવ્યો છે. તેથી તે તેને શોધી શકશે નહીં. જ્યારે મેં દાખલા તરીકે આ જોડે બે નકલો કરી હોય તો તે આ સાચી રીતે મળી હોત અને મને ડીડી પછી ડીડી આપવામાં આવશે. તેથી, શબ્દમાળાઓ ઓવરલેપ થવાની કોઈ સમસ્યા નથી, તે માત્ર તેને જમણેથી ડાબેથી કરે છે અને તે ખાતરી કરે છે કે ઓવરલેપ સ્ટ્રિંગ પ્રથમ લખાયેલ છે, તેથી તે બીજો નકલ રૂપાંતરિત થશે નહીં.

(સ્લાઇડ સમયનો સંદર્ભ લો: 07:17)

આગલી વસ્તુ જે આપણે જોવા માંગીએ છીએ તે એક સ્ટ્રિંગને વિભાજન કરે છે. હવે જ્યારે આપણે સ્પ્રેડશીટ લઈએ છીએ અને તેને ટેક્સ્ટ તરીકે લખીએ છીએ, ત્યારે સામાન્ય રીતે શું થાય છે કે આપણી પાસે આઉટપુટ હશે જે આના જેવું દેખાય. પ્રથમ સ્તંભ લખવામાં આવશે પછી અલ્પવિરામ પછી બીજા સ્તંભ, તેથી જો અમારી પાસે ત્રણ સ્તંભ હશે તો પ્રથમ સ્તંભ 6 સેકન્ડનો સ્તંભ 7 સેટ કરશે અને ત્રીજો શબ્દ હેલ્લો શબ્દ છે, પછી આપણે તેને લખીશું કે તમે 6, 7 મળશે. અને "હેલ્લો" વાસ્તવમાં "હેલ્લો" થોડી સમસ્યા છે કારણ કે તેમાં ડબલ અવતરણચિહ્નો છે, ચાલો આપણે હેલ્લોનો ઉપયોગ ન કરીએ, અમને સરળ કંઈક ઉપયોગ કરીએ. તો ચાલો આપણે કહીએ કે અમારી પાસે 3, 6, 7 અને 8 નો દાખલો છે. હવે, આપણે શું કરવા માંગીએ છીએ તે આપણે આ માહિતીને બહાર કાઢવા માંગીએ છીએ. તેથી, આપણે વ્યક્તિગત 6, 7 અને 8 ને કાઢવા માંગીએ છીએ ત્રણ મૂલ્યો હતા તેથી આપણે શું કરવાની જરૂર છે તે આ શબ્દને શબ્દમાળાઓ વચ્ચે જુએ છે, તેથી આપણે સ્તંભમાં વિભાગોને અલ્પવિરામ વચ્ચે વિભાજિત કરવાનું વિચાર્યું છે અને આ સ્પ્લિટ કમાન્ડનો ઉપયોગ કરીને કરવામાં આવે છે. તેથી, વિભાજિત એક શબ્દમાળા લે છે અને એક અક્ષર લે છે અથવા વાસ્તવમાં કોઈપણ શબ્દમાળા હોઈ શકે છે અને તે સ્તંભોને વિભાજિત કરે છે જે તમને અલ્પવિરામ વચ્ચેના ભાગો લઈને આવતી મૂલ્યોની સૂચિ આપે છે. તેથી, પ્રથમ અલ્પવિરામ સુધી પ્રથમ વસ્તુ છે તેથી કૉલમ એ ફક્ત એક નામ છે જેનો અમે ઉપયોગ કર્યો છે, તે કોઈ સૂચિ હોઈ શકે છે. સૂચિની પ્રથમ આઇટમ પ્રથમ અલ્પવિરામ સુધી હશે જે પછી પ્રથમ અને બીજા અલ્પવિરામ વચ્ચે અને તેથી વધુ અને છેલ્લા કોમા પછી. આ કિસ્સામાં અલ્પવિરામ એક ખૂબ જ વિશિષ્ટ વસ્તુ નથી જે તમે કોઈ વિભાજક શબ્દમાળાનો ઉપયોગ કરીને વિભાજિત કરી શકો છો. અને ફરી સ્થાનાંતરણની જેમ આપણે કેટલી વખત બદલીએ તે નિયંત્રિત કરી શકીએ છીએ, અહીં આપણે નિયંત્રિત કરી શકીએ કે તમે કેટલા સ્પ્લિટ કરો છો. તેથી, તમે કહી શકો છો કે આ સ્ટ્રિંગ નોટિસ પ્રમાણે વિભાજિત થઈ શકે છે કે આ કોઈપણ શબ્દમાળા હોઈ શકે છે તેથી અહીં આપણે તેને જગ્યા કોલોન સ્પેસ મુજબ વિભાજિત કરી રહ્યા છીએ. પરંતુ અમે કહીએ છીએ કે વધારે સંખ્યામાં ભાગ ન કરો, જો આપણી પાસે સંખ્યા કરતા વધુ સંખ્યા હોય અથવા જે અમુક હિસ્સાઓ જે કોઈ ચોક્કસ બિંદુથી આગળ આવી હોય તો આપણે તેને એક બાકીની સ્ટ્રિંગ તરીકે ગણી બનાવીશું અને તેને અમારી સાથે રાખીશું. તો ચાલો આપણે જોઈએ કે આ કેવી રીતે કાર્ય કરે છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 9:22)

ધારો કે આ અમારી ટેક્સ્ટની લાઇન છે જે હું CSV રેખાને બોલાવીશ, તે અલ્પવિરામ દ્વારા અલગ પાડવામાં આવેલા મૂલ્યોનો ક્રમ છે, જો તે શબ્દમાળા છે. હવે જો હું કહું છું કે સીવીવી લાઇન ડોટ સ્પ્લિટ અલ્પવિરામને અલગ પાડનાર તરીકે ઉપયોગ કરે છે અને પછી હું સ્ટ્રિંગ 6, સ્ટ્રિંગ 7, સ્ટ્રિંગ 8 ની સૂચી મેળવી શકું છું. યાદ રાખો કે આ બરાબર છે કે આપણે ઇનપુટ વિશે શું કહ્યું છે, તે તમને મુલ્ય ન મળી શકે. ફોર્મમાં જે તમે ઇચ્છો છો તે પછી તમારે તેમને પૂર્ણાંકનો ઉપયોગ કરીને કન્વર્ટ કરવો પડશે અથવા આ હજુ પણ શબ્દમાળાઓ છે. તેથી, તે માત્ર એક લાંબી શબ્દમાળા લે છે અને તેને નાના શબ્દોની યાદીમાં વિભાજિત કરે છે. હવે અહીં ત્રણ ઘટકો છે તેથી જો હું ઉદાહરણ તરીકે કહું તો મને ફક્ત 0, 1, 2 ની સ્થિતિ જોઈએ છે. તેથી, જો હું કહું કે મારે માત્ર એક જ કરવું છે તો મને પ્રથમ 6 મળશે, પરંતુ તે પછી 7 અને 8 વિભાજિત થતું નથી. કારણ કે તે માત્ર એક જ વખત નાંખે છે હવે, જો હું આને કંઈક વધુ ફેન્સી ગણાતો હોઉં તો હેશ કમ્પની પ્રશ્ન ચિહ્ન તો હવે મારી પાસે એક અલગ વિભાજક છે, તે એક અક્ષર નથી, પણ હેશ પ્રશ્ન ચિહ્ન છે, હું હેશ પ્રશ્ન ચિહ્ન પ્રમાણે વિભાજિત કરી શકું છું અને આ મને તે જ વસ્તુ આપશે. તમે કોઈપણ સ્ટ્રીંગના આધારે વિભાજિત કરી શકો છો, તે માત્ર એકસમાન શબ્દમાળા છે. ત્યાં વધુ ફેન્સી વસ્તુઓ છે જે તમે નિયમિત અભિવ્યક્તિ અને તે સાથે કરી શકો છો, પરંતુ અમે તે માટે હવે આવરી લેવામાં આવશે નહીં. જ્યાં સુધી તમારી પાસે એક નિશ્ચિત સ્ટ્રિંગ હોય છે જે તમારી વસ્તુને અલગ કરે છે તમે તે નિશ્ચિત સ્ટ્રિંગ મુજબ વિભાજિત કરી શકો છો.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 10:49)

તેથી, વિભાજનનું વ્યસ્ત ઓપરેશન શબ્દમાળાઓ સાથે જોડાવાનું રહેશે. ધારો કે, અમારી પાસે શબ્દમાળાઓનો સંગ્રહ છે અને હું તેને એક અલગ સ્ટ્રિંગમાં ભેગા કરવા માંગું છું જે દરેકને આપેલ વિભાજક દ્વારા અલગ કરે છે. તેથી ઉદાહરણ તરીકે, આપણે ધારવું કે આપણે કેટલાક સી.એસ.વી. આઉટપુટ લે છે અને આપણે તેને અલ્પવિરામ પર કોલમમાં વિભાજિત કરીએ છીએ અને પછી આપણે શબ્દમાળા સાથે જોડાઈ શકીએ છીએ અને તેને મૂલ્ય અલ્પવિરામમાં મુકીએ છીએ અને પછી તેનો ઉપયોગ કોલમ સાથે જોડાય છે. હવે આ થોડી ગૂંચવણમાં મૂકે છે, તેથી જોડાવા કાર્ય છે જે શબ્દમાળા સાથે સંકળાયેલ છે. આ કિસ્સામાં સંબંધિત શબ્દમાળા અલ્પવિરામ છે. તેથી તે કહે છે, વધુ કે ઓછું તમે અલ્પવિરામ ડોટ જોડાણો કોલમ કહેતા હોય છે જે કોલમ સાથે જોડાવા માટે અલ્પવિરામનો ઉપયોગ કરે છે. તેથી, તમે હમણાં જ અહીં એક નામ આપ્યું છે જે શબ્દમાળા જોડે અલ્પવિરામ બરાબર છે અને પછી CSV રેખા શબ્દમાળા dot join કોલમ જોડાય છે. તેથી આ શું કહે છે, અલ્પવિરામનો ઉપયોગ અલગ કરવા માટે કરો જેથી જો છેલ્લાં 6, 7 અને 8 ની જેમ મેં છેલ્લું સમય મેળવ્યું હતું, તો તે હવે તેમને 6 અલ્પવિરામ 7 અલ્પવિરામ 8 ને એક જ શબ્દમાળામાં ફેરવી નાખશે. અહીં બીજું ઉદાહરણ છે, અહીં આપણી પાસે તારીખ 16 એક મહિના 08 અને વર્ષ 2016 શબ્દમાળાઓ તરીકે આપવામાં આવે છે અને હું તેને તારીખમાં એકસાથે સ્ટ્રિંગ કરવા માંગું છું, જેમ કે આપણે સામાન્ય રીતે હાઇફન્સ સાથે ઉપયોગ કરીએ છીએ. અહીં હાઇફન્સમાં મધ્યવર્તી નામ આપવાની જગ્યાએ અને પછી હાઇફન્સ ડોટ જોડીને કહીને હું સીધા જ આ શબ્દનો ઉપયોગ કરું છું, માત્ર એટલું જ સમજવું છે કે તમે સીધા જ આ શબ્દમાળાને પોતાની રીતે સતત સ્ટ્રિંગ તરીકે ઉપયોગ કરી શકો છો અને આનો ઉપયોગ મૂલ્યોની સૂચિમાં જોડાવા માટે કરો. તમને ખાતરી કરવાની જરૂર છે કે તમારી પાસે દલીલમાં જોડાવાની અંદર શું છે શબ્દમાળાઓની સૂચિ છે અને તમે જે શબ્દ પર લાગુ કરો છો તે શબ્દમાળા છે જેનો ઉપયોગ તેમની સાથે જોડાવા માટે કરવામાં આવશે. ચાલો આપણે તપાસીએ કે આપણે ખરેખર કરવાના હેતુથી કામ કર્યું છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 12:38)

ચાલો બીજું ઉદાહરણ સીધું કરીએ. ધારો કે તારીખ 16 છે, યાદ રાખો કે આ બધી સ્ટ્રિંગ્સ મહિના 08 છે, 2016 નું વર્ષ છે, અને હવે હું કહેવા માંગુ છું કે આ ત્રણ વસ્તુઓને ડેશ દ્વારા અલગ પાડનાર તરીકે

ઉપયોગમાં લેવાની અસર શું છે અને મને 16 ડેશ 08 નાં દશામાં 2016 મળે છે. સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 13:09)

તેથી ઘણા અન્ય રસપ્રદ વસ્તુઓ છે જે તમે ઉદાહરણ તરીકે શબ્દમાળાઓ સાથે કરી શકો છો, તમે ઉચ્ચ કેસ અને નીચલા કેસને ચાલાકી કરી શકો છો. જો તમે કહી શકો કે મૂડીકરણ કરવું, તો તે શું કરશે, તે પ્રથમ અક્ષરને ઉપલા કેસમાં રૂપાંતરિત કરશે અને બાકીનાને લોઅર કેસ તરીકે રાખશે, જો તમે કહેતા હોવ તો તે બધા ઉપરના કેસને લોઅર કેસમાં રૂપાંતરિત કરશે, જો તમે કહી શકો છો કે ઉપરના ઉપર તમામ લોઅર કેસને અપર કેસમાં રૂપાંતરિત કરશે અને તેથી. ડોટ ટાઇટલ જેવી અન્ય ફેન્સી વસ્તુઓ પણ છે. તેથી, શીર્ષક દરેક શબ્દ ઉઠાવે છે પુસ્તક અથવા ફિલ્મના શીર્ષકમાં એવું સામાન્ય રીતે દર્શાવવામાં આવે છે. એસ ડોટ સ્વેપ કેસ લોઅર કેસને અપર કેસ અને ઉચ્ચ કેસને લોઅર કેસમાં ફેરવશે અને તેથી. તેથી શબ્દમાળા વસ્તુમાં વિધેયોનું સંપૂર્ણ સંગ્રહ છે જે ઉપલા કેસ, નીચલા કેસ અને આ વચ્ચે પરિવર્તન કેવી રીતે કામ કરે છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 13:52)

બીજી વસ્તુઓ કે જે તમે શબ્દમાળાઓ સાથે કરી શકો છો તે તમારે જે ઇચ્છે છે તે ફિટ કરવા માટે તેનું માપ બદલો. તેથી જો તમે સ્ટ્રિંગ માગો છો કે જે ચોક્કસ પહોળાઈના સ્તંભ તરીકે સ્થિત થયેલ હોય તો આપણે કહી શકીએ કે તે n એ બ્લોકમાં કેન્દ્ર છે. તો આ શું કરે છે તે નવી સ્ટ્રિંગ આપશે જે લંબાઈ n ની છે અને તેમાં કેન્દ્રિત છે. હવે કેન્દ્રિત કરીને આપણે શું અર્થ કરીએ તે છે કે કાં તો બાજુ પર ખાલી બ્લેન્ક્સની જગ્યાએ ખાલી જગ્યાઓ હશે, તમે તારા જેવો કે તારાઓ, અથવા નાની વસ્તુઓને મૂકી શકો છો. તમે એક અક્ષર આપી શકો છો જે ખાલી જગ્યાને બદલે ખાલી જગ્યા ખાલી કરવા માટે ઉપયોગમાં લેવાશે. હવે તમે તેને કેન્દ્રીત ન કરી શકો અથવા તમે ડાબે અથવા જમણે માગતા ન હોઈ શકો, જેથી તમે ઉદાહરણ તરીકે બગાડ્યા દરમ્યાન ન્યાયી ઠરાવી શકો છો અથવા અચકાવું દરમિયાન ન્યાય કરી શકો છો અને ફરીથી તમે એક વૈકલ્પિક પાત્ર આપી શકો છો અને આ રીતે. એસ, અમે હમણાં જ આમાંના એક કે બેમાંથી ફક્ત એક કરી શકીએ છીએ કે તેઓ કેવી રીતે કામ કરે છે. ધારો કે આપણે 'હેલ્લો' જેવી ટૂંકા શબ્દકથન કરીએ છીએ અને હવે આપણે તેને 50 ના મોટા બ્લોકમાં કેન્દ્રિત કરીએ છીએ. આપણે કહીએ છીએ કે ડોટ સેન્ટર 50, તો પછી આ આપણને ઘણા ખાલી સાથે હેલ્લો આપે છે. ક્યાં તો બાજુ પર જગ્યાઓ હવે આપણે તે ખાલી જગ્યાઓને બદલી શકીએ છીએ, તેથી આપણે માઇનસ સાઇન કહીએ છીએ અને તે પછી આપણે ઓછા નિશાનીઓ અથવા હાઇફન્સની સ્ટ્રિંગ મેળવીશું. હવે અમે એમ પણ કહી શકીએ કે મારે આ વસ્તુને કેન્દ્રમાં નહીં રાખવી જોઈએ. તેથી જો હું તે કરું તો મને શરૂઆતમાં અને બાદબાકીના સંકેતોનું એક ટોળું મળશે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 15:23)

કેટલાક અન્ય પ્રકારનાં કાર્યો જે આપણે શબ્દમાળાઓ સાથે સંકળાયેલા છીએ તેને શબ્દમાળાઓના ગુણધર્મો તપાસવું છે. શું માત્ર એટલું જ નહીં કે ઝેડ માટેના અક્ષરો અને મૂડી ઝેડમાં મૂડી. તેથી તે શું છે તે આલ્ફા કહે છે તે એક આલ્ફાબેટીક સ્ટ્રિંગ છે, જો તે સાચું હોય તો તેનો અર્થ એ છે કે જો તે ઓછામાં ઓછા એક બિન આલ્ફાબેટીક અક્ષર ધરાવે છે. તેવી જ રીતે તે સંપૂર્ણપણે અંકો છે, આંકડાકીય અમને જણાવશે જો તે સંપૂર્ણપણે અંકો છે. તેથી, ત્યાં સંખ્યાબંધ સ્ટ્રીંગ ફંક્શનો છે અને આ વસ્તુમાં તે બધામાંથી પસાર થતું કોઈ બિંદુ નથી, જો આપણે તેમની સાથે જઈએ તો અમે તેમને વાપરીશું અને તેમને સમજાવશે. પરંતુ તમે સ્ટ્રીંગ ફંક્શન્સ હેઠળ પાયથોન દસ્તાવેજીકરણ દેખાવને જોઈ શકો છો અને તમને ઉપયોગી ઉપયોગીતાઓનો સંપૂર્ણ યજમાન મળશે જે તમને સ્ટ્રીંગ્સને સરળતાથી હેંધલ કરવા દે છે. અને આ એક કારણ છે કે પાયથોન લોકપ્રિય ભાષા છે કારણ કે તમે આ પ્રકારની સરળ ટેક્સ્ટ પ્રક્રિયા કરી શકો છો. તેથી તમે તેને એક ફોર્મેટમાંથી બીજામાં ઝડપથી રૂપાંતરિત કરવા માટે તેનો ઉપયોગ કરી શકો છો અને તમને ખબર છે કે તે જે રીતે જુએ છે અથવા તેને ફરીથી કદમાં ફેરવવું તે બદલવું અને તેથી

વધુ. સ્ટ્રિંગ ફંક્શન્સ પાઇથોન્સ યુટિલિટીનો એક મહત્વનો ભાગ છે, જે એક ફોર્મેટમાં બીજી વસ્તુઓને રૂપાંતરિત કરવા માટે ગુંદર ભાષા છે.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રો. માધવન મુકુંદ,
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 05 લેક્ચર - 05 ફોર્મેટિંગ પ્રિન્ટેડ આઉટપુટ

(સ્લાઇડ ટાઇ નો ઉલ્લેખ કરો: 00:01)

આ સપ્તાહના વ્યાખ્યાનમાં, અમે જણાવ્યું હતું કે પ્રિન્ટ દ્વારા આઉટપુટ કેવી રીતે પ્રદર્શિત થયું તે આપણે મર્યાદિત રીતે નિયંત્રિત કરી શકીએ. હવે ડિફોલ્ટ પ્રિન્ટ દ્વારા પ્રિન્ટ કરવા માટેની વસ્તુઓની સૂચિ લે છે, તેને ખાલી જગ્યાઓ દ્વારા અલગ કરે છે અને અંતે નવી લીટી મૂકે છે. જો કે, તે વૈકલ્પિક દલીલ અંતને સ્ટ્રિંગ સમાન લે છે જે આપણે અંતે શું બદલ્યું છે, ખાસ કરીને જો આપણે ખાલી સ્ટ્રિંગ મૂકીએ તો તેનો અર્થ એ કે તે એક નવી લીટી શરૂ કરતું નથી, તેથી આગામી પ્રિન્ટ સમાન લાઇન પર ચાલુ રહેશે. એ જ રીતે આપણે અલગ જગ્યાને અલગ જગ્યા બદલી શકીએ છીએ અને ખાસ કરીને જો કોઈ જગ્યા ન હોય તો આપણે તેમને પોતાને મૂકી શકીએ છીએ અને ફક્ત કહી શકીએ કે વિભાજક કંઈ નથી - ખાલી શબ્દમાળા.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 00:43)

હવે, કેટલીકવાર તમે થોડો વધુ ચોક્કસ બનવા માંગો છો, તેથી આ માટે આપણે ફોર્મેટ પદ્ધતિનો ઉપયોગ કરી શકીએ છીએ જે ખરેખર સ્ટ્રિંગ લાઇબ્રેરીનો એક ભાગ છે. તેથી, જે વસ્તુઓ આપણે સ્ટ્રિંગ્સ સાથે કરી શકીએ છીએ, છેલ્લા, છેલ્લી લેક્ચરમાં, આપણે અન્ય વસ્તુઓ પર જોયું છે, આપણે સ્ટ્રિંગ કરી શકીએ છીએ, શોધી શકીએ છીએ, બદલી શકીએ છીએ અને આ બધી વસ્તુઓ કરી શકીએ છીએ, તેથી આ તે સમાન વર્ગમાં છે. યાદ રાખો કે જ્યારે તમે પ્રિન્ટ કરી રહ્યા છો, ત્યારે તમે વાસ્તવમાં શબ્દમાળા છાપી રહ્યા છો. તેથી, સ્ટ્રિંગને સુધારવા માટે તમે જે કંઈ પણ કરી શકો છો તે તમને બીજી સ્ટ્રીંગ આપશે જે તમે છાપી રહ્યા છો. તેથી, અહીં સ્ટ્રિંગ વાસ્તવમાં ફોર્મેટ પદ્ધતિને કોલ કરવા જઈ રહ્યું છે. તેથી, આ કરવા માટે સૌથી સરળ રીત ઉદાહરણ છે. આપણી પાસે બેઝ સ્ટ્રિંગ અહીં છે, જે પ્રથમ, સેકન્ડ છે અને કૌંસમાં આ બે રમૂજ વસ્તુઓ છે. કૌંસમાં રમૂજ વસ્તુઓને વિધેયમાં દલીલોના સમકક્ષ માનવામાં આવે છે, આ વસ્તુઓને વાસ્તવિક મૂલ્યો દ્વારા બદલવામાં આવે છે અને પછી શું થાય છે એ છે કે જ્યારે તમે આ શબ્દમાળા આપો છો અને તમે બંધારણ પદ્ધતિને લાગુ કરો છો તો 0 નો સંદર્ભ આપે છે પ્રથમ દલીલ અને 1 એ બીજા દલીલનો ઉલ્લેખ કરે છે. તો આપણે શું કરી રહ્યા છીએ, આપણે સ્થાને બદલી રહ્યા છીએ, તેથી જો હું વાસ્તવમાં આ શબ્દમાળા લઈશ અને તેને પાયથોન માં પસાર કરું છું તો પરિણામી વસ્તુ પ્રથમ કોલન 47, બીજા કોલન 11 છે, કારણ કે પ્રથમ દલીલ, બ્રેસ 0 ને લીધું છે 47 ના ફોર્મેટમાં પ્રથમ દલીલ અને બીજા સ્થાને છે. હવે સ્થાનો નામો નક્કી કરે છે જેથી તેઓ એ જ ક્રમમાં ઉપયોગમાં ન હોય. તેથી, આપણે પહેલી દલીલ છાપી શકીએ છીએ અને પછી દ્વિતીય ઉદાહરણ શો તરીકે દલીલ છાપો. અનિવાર્યપણે, ફોર્મેટનું આ સંસ્કરણ અમને ફોર્મેટ વસ્તુમાં તેમની સ્થિતિ દ્વારા શબ્દમાળામાં વસ્તુઓ પસાર કરવાની મંજૂરી આપે છે. તેથી અમે સ્થિતિ દ્વારા દલીલો બદલી રહ્યા છીએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 02:34)

હવે આપણે આ જ વસ્તુ નામ દ્વારા કરી શકીએ છીએ. આ એક ફંક્શનને વ્યાખ્યાયિત કરવા બરાબર છે જે યાદ છે, અમે કહ્યું હતું કે અમે ફંક્શન દલીલો આપી શકીએ છીએ અને તે નામ દ્વારા પસાર કરી શકીએ છીએ. તેથી, તે જ રીતે અહીં આપણે ફોર્મેટમાં આર્ગ્યુમેન્ટ્સના નામોને સ્પષ્ટ કરી શકીએ છીએ, આપણે

કહી શકીએ એક 47 બરાબર છે, 11 બરાબર 11 છે, તે પ્રથમ અને સેકન્ડ છે. હવે આપણે કહી શકીએ એક એક અને બે ઓ હવે અહીં લાભ પદ દ્વારા પરંતુ નામ દ્વારા નથી. જો હું આ બે વસ્તુઓ લઈશ અને હું તેમને બદલી કરું છું, તેથી જો હું બીજી દલીલ એક અને પ્રથમ દલીલ કરું છું, અને હું તેને એક જ શબ્દમાળામાં પસાર કરું છું તો f યોગ્ય રીતે 47 અને s 11 તરીકે પકડવામાં આવશે, તેથી અહીં ઉપયોગ કરીને નામ સ્થિતિ નથી તેથી, જે ફોર્મેટમાં તમે વસ્તુઓને ઓફર કરો છો તે કોઈ બાબત નથી.

(સ્લાઇડ સમય જણાવો: 03:20)

તો, આ બિંદુ સુધી આપણે કોઈપણ ફોર્મેટિંગ કર્યું નથી. આપણે જે કર્યું છે તે આપણે એક સ્ટ્રિંગ લીધું છે અને આપણે સ્ટ્રિંગમાં સ્થાન ધારકો માટે મૂલ્ય કેવી રીતે બદલવું તે અમને જણાવ્યું છે. ત્યાં કોઈ વાસ્તવિક ફોર્મેટિંગ નથી થયું કારણ કે જે કંઈ કર્યું છે તે અમે પહેલાથી કર્યું હોતહાલના પ્રિન્ટ સ્ટેટમેન્ટનો ઉપયોગ કરીને જે આપણે જોઈએ છીએ. હવે પ્રત્યક્ષ ફોર્મેટિંગ આ સ્થાન ધારકોને કેવી રીતે પ્રદર્શિત કરવું તે અંગે વધારાની સૂચનાઓ આપીને આવે છે. તો અહીં આપણી પાસે 0 ની કેટલીક રમૂજી વસ્તુ છે, અમારી પાસે આ ખાસ કોલોન છે અને કોલોન એ ફોર્મેટિંગ સૂચના છે પછી શું આવે છે. આનો બે ભાગ અહીં છે, આપણે 3 અને જાહેરાત જોઈએ છીએ અને તેનો અર્થ અલગ અલગ વસ્તુઓ છે. તેથી સમગ્ર 3 ડી આપણને કહે છે કે કિંમત કેવી રીતે પ્રદર્શિત કરવી તે અહીં પસાર થવાની છે, તે પહેલી વસ્તુ છે. ડી એક કોડ છે જે સ્પષ્ટ કરે છે, મને લાગે છે કે તે દશાંશ જેટલો છે, તેથી ડી સ્પષ્ટ કરે છે કે 4 ને પૂર્ણાંક મૂલ્ય તરીકે ગણવા જોઈએ. તેથી, આપણે વાસ્તવમાં તેને એક સામાન્ય પૂર્ણાંક મૂલ્ય તરીકે દર્શાવવું જોઈએ, એટલે કે તે બેઝ દસ પૂર્ણાંક છે. છેલ્લે, 3 કહે છે કે આપણે ફોર્મેટ 4 કરવું જોઈએ જેથી તે ત્રણ જગ્યાઓ લઈ શકે. તે ત્રણ જગ્યાઓની સમકક્ષ હોવા છતાં તે સિંગલ ડિજ છે. તેથી જો હું આ બધું કરું તો શું થાય છે, મને મૂલ્ય મળે છે, હવે નોંધ લો કે પહેલેથી જ અહીં એક જગ્યા છે, પછી તે ત્રણ જગ્યાઓ લઈ જવાનો છે તેથી હું બે જગ્યાઓ ખાલી કરીશ અને પછી 4, તેથી જ શા માટે છે આપણી પાસે આ લાંબુ છે, તેથી તે વાસ્તવમાં ત્રણ ખાલી જગ્યાઓ અને પછી 4 છે. આ સમગ્ર બાબત, તેનો આ ભાગ બંધારણમાંથી આવે છે. તેથી મારી પાસે ખાલી જગ્યા છે, ખાલી જગ્યા અને 4, કારણ કે મને 4 ની પહોળાઈમાં 4 મૂકવા કહેવામાં આવે છે અને તેને એક સંખ્યા તરીકે લાગે છે, તેથી તેનું સંખ્યા જમણી બાજુ જાય છે.

(સ્લાઇડ સમય જણાવો: 05:06)

ચાલો બીજા ઉદાહરણ જોઈએ. ધારો કે મારી પાસે સંખ્યા હતી જે પૂર્ણાંક નથી, પરંતુ ફ્લોટિંગ બિંદુ નંબર છે, તેથી તે 47.523 છે. હવે અહીં પ્રથમ વાત એ છે કે આપણી પાસે v ની જગ્યાએ છે. આપણે ફ્લોટિંગ બિંદુ માટે f છે. તેથી, f, 6.2f, કોલોનની જમણી બાજુથી આ સમગ્ર બાબત મને કહે છે કે કિંમતો કેવી રીતે ફોર્મેટ કરવી તે અહીંથી આવે છે. નીચે પ્રમાણે 6.2f તોડે છે; એક, તેનો પત્ર ભાગ હંમેશા મને કહે છે કે મૂલ્ય પ્રકાર શું છે? તેથી, તે કહે છે કે 47.523 ને ફ્લોટિંગ બિંદુ મૂલ્ય તરીકે ગણવા જોઈએ. અને બીજી વાત એ છે કે તે કહે છે કે આ 6 મને કહે છે કે મને જે લખવાનું છે તે લખવા માટે કેટલું જગ્યા છે. તેથી તે દશાંશ ચિહ્ન સહિત કુલ મૂલ્ય દર્શાવે છે, બધું 6 અક્ષરોથી વિશાળ રહેશે. છેલ્લે, 2 કહે છે કે દશાંશ ચિહ્ન પછી કેટલા અંકો બતાવવા. જો હું આ બધું લાગુ કરું તો સૌ પ્રથમ તો, કારણ કે મારી પાસે દશાંશ ચિહ્ન પછી ફક્ત બે અંકો છે, 3 આ બોલ પર કોઈ રન નોંધાયો નહીં, અને પછી કારણ કે તે 6 અક્ષર વાપરવા માટે સુયોજિત છે, હવે જો હું જમણીથી ગણતરી કરું છું, આ 1, 2, 3, 4, 5 અક્ષરો, પરંતુ તે 6 અક્ષરોનો ઉપયોગ કરવા માટે સુયોજિત છે, એટલા માટે અહીં વધારાની ખાલી જગ્યા છે જો તમે અહીં નોંધ કરો છો, તો ખાલી એક ખાલી છે, પરંતુ અહીં બે ખાલી જગ્યા છે. બીજા ખાલી ફોર્મેટ સ્ટેટમેન્ટમાંથી આવે છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 06:27)

દુર્ભાગ્યવશ આ બરાબર વપરાશકર્તા મૈત્રીપૂર્ણ અથવા તમે સરળતાથી યાદ રાખી શકો તે કંઈક નથી, પરંતુ અન્ય મૂલ્યો માટે કોડ્સ છે આપણે એક અને ડી જોયો છે, તેથી કોડ માટે ઓ જેવા સ્ટ્રીંગ્સ, અને ઓક્ટેકલ માટે ઓ અને હેક્સાડેસિમ માટે x છે. આ ફોર્મેટ કરેલ વસ્તુઓનો ઉપયોગ કરીને આ બધા મૂલ્યો પ્રદર્શિત કરી શકાય છે. અને તમે અન્ય વસ્તુઓ પણ કરી શકો છો જેને તમે તેને કહી શકો છો કે તે તેને ડાબી બાજુ પર મૂકવા નહીં, જેથી તમે મૂલ્યને યોગ્ય ઠરાવી શકો. ઉદાહરણ તરીકે પહોળાઈ 5 માં, જો તમે સ્ટ્રિંગ મૂકવા માંગતા હો તો તમે કહી શકો છો કે શબ્દમાળા ડાબી બાજુથી આવે છે, જમણેથી નહીં પછી તમે અગ્રણી શૂન્યતાઓ ઉમેરી શકો છો, જેથી તમે નંબર 4 નો પહોળાઈ 3 નહીં 4 તરીકે, પરંતુ 004 તરીકે પ્રદર્શિત કરી શકો. તો, આ બધી વસ્તુઓ તમે કરી શકો છો. જેમ મેં કહ્યું હતું કે આ વસ્તુઓને ફોર્મેટ કરવાની સંપૂર્ણ ઝૂ છે જે તમે આ સાથે કરી શકો છો. આ તમામ ભાષા સી અને 'સી' માં 'printf' તરીકે ઓળખાતા સ્ટેટમેન્ટ છે, તેથી અમારા 0, 3 ડી અને 6.2f અને તેઓ જેનો અર્થ થાય છે તે બધામાં ચોક્કસ ફોર્મેટ સ્ટેટમેન્ટ. તે શ્રેષ્ઠ છે કે તમે પાચથોન દસ્તાવેજો જુઓ, તમારે તેના તમામ ભિન્નતાઓની જરૂર ન પડી શકે, તમને જેની જરૂર હોય તે તમે જોઈ શકો છો, પરંતુ તે જાણવા માટે ઉપયોગી છે કે આ પ્રકારની ફોર્મેટિંગ કરી શકાય છે.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ અને એલ્ગોરિધમ્સનો ઉપયોગ કરીને પાચથોન

પ્રો. માધવન મુકુન્દ

ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,

ઇન્ડિયન ઇન્સ્ટિટ્યૂટ ઓફ ટેકનોલોજી, મદ્રાસ

અઠવાડિયું - 05 લેક્ચર - 06 પાસ, ડેલ () અને કોઈ નહીં

(સ્લાઇડ ટાઇ નો ઉલ્લેખ કરો: 00:01)

છેલ્લા ભાષણ માટે આ અઠવાડિયે આપણે કેટલીક ઉપયોગી બાબતોને જોવી જોઈએ જે પાક કરશે અને જે ક્યાંય પણ ફિટ ન હોય તેથી અમે આ એક જ પ્રસ્તુતિમાં તેમને એક સાથે જોડીએ. તેથી, જ્યારે અમે ઇનપુટ કરી રહ્યાં છીએ અને છાપવા માટે વપરાશકર્તાને કેવી રીતે પ્રોમ્પ્ટ કરી રહ્યા છે તે વિશે એક ઉદાહરણ હતું, જો તેઓએ અયોગ્ય એન્ટ્રી કરી હોય અમે એક નંબર વાંચવાનો પ્રયાસ કરી રહ્યા હતા અને આપણે શું કહ્યું હતું કે વપરાશકર્તા દ્વારા પૂરી પાડવામાં આવતી કેટલીક સ્ટ્રિંગ ઇનપુટ થશે, તો તેમને સંદેશો આપીને કહીએ કે તેઓ અમને શબ્દમાળા આપીને નંબર દાખલ કરે છે અને પછી આપણે તેને ઇન ફંક્શનનો ઉપયોગ કરીને રૂપાંતરિત કરવાનો પ્રયાસ કરીએ છીએ. અને આ પૂર્ણાંક કાર્ય નિષ્ફળ જશે જો વપરાશકર્તાએ સ્ટ્રિંગ પ્રદાન કરી છે જે એક માન્ય પૂર્ણાંક નથી, જે કિસ્સામાં આપણને મૂલ્યની ભૂલ મળશે. અને આપણને મૂલ્યની ભૂલ મળે છે, અમે ફરી એક વાર કહેતા સંદેશો છાપીએ છીએ અને પાછા જઈએ છીએ. છેલ્લે, જો આપણે સફળ થવું હોય, તો તે આ ઇન્ટ કાર્યોની સફળતાનું કામ કરે છે અને પછી આમાંથી અવરોધિત કરો અને બીજા ભાગો પર જાઓ અને બીજું આંટીઓમાંથી બહાર નીકળી જશે. તેથી, આ આપણે પહેલેથી જોયું હતું. હવે પ્રશ્ન એ છે કે, જો આપણે આ બદલવા માંગીએ છીએ જેથી આપણે કંઈ કરવું નહીં કરીએ. બીજા શબ્દોમાં કહીએ તો જો વપરાશકર્તા તેમને કહેવાની જગ્યાએ માન્ય મૂલ્ય પ્રસ્તુત કરતા નથી કે શા માટે અમે ફક્ત એક નંબર દાખલ કરવાનું કહીએ છીએ, તેનો અર્થ એમ કે અમે આ સંખ્યા ઘણી વાર જોઈ છે, તમે કેટલાક વપરાશકર્તા ઇન્ટરફેસ પર જાઓ છો અને તમે કંઈક ખોટું લખો છો, તે નથી. તમને ખોટું શું છે તે જણાવો કે તે ફક્ત પાછળ અને પાછો અને પાછળ જવાનું ચાલુ રાખે છે અને ફરી ટાઇપ કરવાનું પૂછે છે. તો, તમે કેવી રીતે વાસ્તવમાં કંઈક તરીકે તે તરફેણકારી હોવ? આપણે શું કહેવા માંગીએ છીએ, જો હું આ મૂલ્યની ભૂલ પર આવીશ તો કંઈ નહીં હવે પાચથોન સાથેની સમસ્યા એ છે કે તમે જ્યાં પણ આ પ્રકારની કોલોન કરો ત્યાં તે પછી કંઈક અપેક્ષા રાખશે, તમારી પાસે ખાલી બ્લોક નહીં હોઈ શકે. તેથી જો હું કોલોન કરું તો તેના પછી ઓછામાં ઓછું એક નિવેદન હોવું જોઈએ. આ પાચથોન નું એક વાક્યરચના નિયમ છે, જે તમારી પાસે ખાલી બ્લોક નથી. પણ અહીં હું કંઈ કરવાનું નથી ઈચ્છું, મારે ત્યાં એક મૂલ્યની ભૂલ છે તે ઓળખવું છે અને પછી અહીં પાછા જાઓ તે સાફ છે, પણ હું બીજું કશું કરવા નથી માગતું. હું પાચથોન માં કંઈ કંઈ કરું? તેથી જવાબ છે, કે પાસ તરીકે ઓળખાતું વિશિષ્ટ સ્ટેટમેન્ટ છે પાસ એક નિવેદન છે જે ફક્ત જગ્યાઓ ભરવા માટે જ અસ્તિત્વ ધરાવે છે જે ભરવા જરૂરી છે. તેથી જ્યારે તમે બ્લોકોને સિવાય અથવા બીજા જેવા હોય છે, જો તમે બ્લોકનું નામ ત્યાં મૂકો છો તો તમે તેને ખાલી છોડી શકતા નથી. આ કિસ્સામાં તમે કશું કરવા માટે શબ્દ પાસનો ઉપયોગ કરી શકો છો.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 02:28)

ધારો કે મારી પાસે એક સૂચિ છે અને હું સૂચિ મધ્યમાં એક તત્વ દૂર કરવા માંગુ છું. તેથી આ કરવા માટેનો એક રસ્તો એ છે કે, તે પદ પરથી સ્લાઇસ અપ લેવું, તે સ્થાનમાંથી આ સ્લાઇસ પછી વત્તા અને તેથી ઉપયોગ કરીને તેને એકસાથે ગુંદર કરો. પરંતુ જો હું સીધી આ કરવા માંગુ છું તો શું? તે તારણ આપે છે કે ડેલ નામના આદેશ છે. જો હું કહું કે ડેલ એલ 4 અને તે શું કરે છે તો તે વર્તમાન કિંમતોના મૂલ્યમાંથી અસરકારક રીતે દૂર કરે છે, અને તે સ્વયંચાલિત સૂચિને ઠીક કરે છે અને 5 દ્વારા વાઈસથી

લઈને ડાબી બાજુએ સ્થાન 5 સુધી બધું કરે છે. તેથી ચાલો આપણે આ કાર્યો જે રીતે તે કામ કરવાનો દાવો કરે છે

(સ્લાઇડ સમયનો સંદર્ભ લો: 03:09)

ધારો કે, અમે અમારી સૂચિ 0 થી 10, 0 થી 9 સુધીના મૂલ્યોની રેન્જમાં સેટ કરીએ છીએ, અને હવે હું કહું છું કે લીલ 4 પછી 1 0, 1, 2, 3, 5, 6, 7, 8, 9, કારણ કે 1 4 એ મૂલ્ય 4 હતું, યાદ કરો કે પદ 0 થી શરૂ થાય છે. અને તેથી તે ખરેખર ચોથા ક્રમાંકનું મૂલ્ય કાઢી નાખ્યું છે. આ શબ્દકોશો પણ કામ કરે છે. જો આપણે કી કી સાથે સંકળાયેલ મૂલ્ય દૂર કરવા માંગીએ છીએ, તો તમે ડેલ ડીકે કરી શકો છો અને તે કી કીને દૂર કરશે અને તેની સાથે સંકળાયેલ મૂલ્યો અને તેમાંથી દૂર કરવું પડશે. તેથી, કી હવે એક અવ્યાખ્યાયિત કી તરીકે ગણવામાં આવશે.

(સંદર્ભ લોસ્લાઇડ ટાઇમ: 03:56)

સામાન્ય રીતે આપણે એક્સ જેવા નામ લઈ શકીએ અને ડેલ એક્સને કહી શકીએ અને આ શું કરે છે તે એક્સને અવ્યાખ્યાયિત બનાવે છે. ધારો કે, તમે કેટલાક જંક કોડ લખ્યા છે જેમ આપણે x બરાબર 7 સેટ કર્યું છે અને પછી આપણે કહીએ છીએ ડેલ એક્સ, અને પછી આપણે y વત્તા 5 વત્તા 5 ની બરાબર કહીએ છીએ. હવે આ બિંદુ કારણ કે x એ અવ્યાખ્યાયિત છે, તેમ છતાં તેની પાસે 7 નું મૂલ્ય છે અભિવ્યક્તિત x વત્તા 5 નું ગણતરી કરી શકાતું નથી કારણ કે x પાસે હવે મૂલ્ય નથી અને તમે જે સાથે અંત આવશે તે ભૂલ છે જે આપણે પહેલા નામની ભૂલથી કહીએ છીએ કે નામ x વ્યાખ્યાયિત નથી.

(સ્લાઇડ સમય જણાવો: 04:29)

જો તમે નામ વ્યાખ્યાયિત થયેલ છે તે ચકાસવા વિશે તમે કેવી રીતે જાઓ છો, અલબત્ત તમે અપવાદ હેન્ડલિંગનો ઉપયોગ કરી શકો છો. ધારો કે, જો આપણે x ને અવ્યાખ્યાયિત હોય તો જ નામ x ને વેલ સોંપવા માંગીએ છીએ, તો આપણે કહી શકીએ x તેથી આ x સાથે કંઈક કરવા માટે પ્રયાસ કરી રહ્યું છે. યાદ રાખો કે નામો કંઈ હોઈ શકે છે, તે કાર્યો હોઈ શકે છે. તેથી, તમે ફક્ત એક્સ લખી શકો છો કારણ કે જો તે વર્તમાનમાં કાર્યના નામે છે જે કોઈ દલીલો લેતા નથી તો અમે તે કાર્ય ચલાવવાનો પ્રયત્ન કરીશું, તેથી તે ફક્ત x લખવા માટે સંપૂર્ણપણે માન્ય છે. પરંતુ x જો તેની પાસે કોઈ કિંમત નથી તો તે નામની ભૂલ આપશે. તેથી, તમે કહી શકો છો એક્સ પ્રયાસ કરો અને જો તમને કોઈ નામની ભૂલ શોધવાનું થયું હોય તો તેને 5 માં મૂકી દો. જો x પાસે પહેલેથી મૂલ્ય છે તો તે x ને કંઈ પણ કરશે નહીં, જો x પાસે મૂલ્ય નથી, તો તે x થી 5 ની કિંમતને અપડેટ કરશે.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 05:15)

હવે, સામાન્ય રીતે શું થાય છે તે આપણે કરવા માંગીએ છીએ તપાસો કે કંઈક ક્યાંથી નક્કી કરવામાં આવી છે કે નહીં. તે ફક્ત તેને અવ્યાખ્યાયિત રાખવાનું અને તે કરવા માટે અપવાદ હેન્ડલિંગનો ઉપયોગ કરવાનું એક સારું વિચાર નથી, કારણ કે તમે વાસ્તવમાં વિચિત્ર વસ્તુઓ થઈ રહ્યાં છો. તેથી, પાયથોન આપણને મૂડી એન સાથે વિશિષ્ટ મૂલ્ય આપે છે, જે કંઈ વ્યાખ્યાયિત કરવા માટે વિશિષ્ટ મૂલ્ય છે - ખાલી મૂલ્ય અથવા નલ મૂલ્ય. જ્યારે આપણે આપણા પોતાના ડેટા સ્ટ્રક્ચર્સને વ્યાખ્યાયિત કરીએ છીએ ત્યારે તેના માટે અમે તેનો ખૂબ સારો ઉપયોગ કરીશું, પરંતુ અત્યારે કોઈ વિશિષ્ટ મૂલ્ય તરીકે વિચારીશું નહીં, ત્યાં કોઈ એકમાં માત્ર એક જ મૂલ્ય છે અને તે કંઈ નથી સૂચવે છે. તેથી, લાક્ષણિક ઉપયોગ એ છે કે જ્યારે આપણે તપાસવું કે નામનું માન્ય મૂલ્ય છે કે નહીં તે આપણે તેને કોઈ પણને પ્રારંભ કરી શકીએ છીએ અને પછીથી તે તપાસી શકે છે કે તે હજુ પણ કંઈ નથી. તો, શરૂઆતમાં આપણે કહીએ છીએ x એ કોઈની બરાબર છે અને છેવટે આપણે આગળ વધીએ છીએ અને કહીએ છીએ, જો x

એ કંઈ નથી તો y બરાબર x ને સુયોજિત કરો, x બરાબર બીજો એક શબ્દ એક્ઝેક્યુટ થશે નહીં જો x એ હજુ પણ કંઈ નથી તો હવે જુઓ, વિશિષ્ટ શબ્દ નથી. તો, આપણે ઉપયોગ કરી રહ્યા છીએ તે બરાબર નથી. તેથી જગ્યામાં પાચથોનમાં બરાબર એક મૂલ્ય નથી. બધા નોન્સ એ જ વસ્તુ પર નિર્દેશ કરે છે યાદ રાખો કે જ્યારે અમે 1 1 છે, 2 ની તપાસ કરી રહ્યા છીએ, ત્યારે આપણે ચકાસી રહ્યા છીએ કે શું 1 1 અને 1 2 બિંદુ એ જ સૂચિ ઓબ્જેક્ટ છે. અમે પૂછતા હતા કે શું એક્સ અને સતત કંઈ પણ એ જ કોઈ વસ્તુને નહીં અને ત્યાં માત્ર એક જ મૂલ્ય નથી. તો, x એ કોઈની સમાન નથી જે x બરાબર ને સમાન છે, તેથી x એ કંઈ નથી જે કોઈ પણ ના બરાબર x નથી. તેથી, x એ કંઈ આગળ નહીં, કોઈની બરાબર નથી વાંચવું સહેલું છે. તેથી જ આપણે આ કહેવું લખીશું.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 06:58)

આપણે ત્રણ ઉપયોગી વસ્તુઓ જોઈ છે જે સમયાંતરે આપણે આગળ વધીએ છીએ. એક એ નિવેદન પાસ છે જે વિશિષ્ટ સ્ટેટમેન્ટ છે જે કંઈ કરે છે અને જ્યારે તમને કોઈ ખાલી બ્લોકની જરૂર હોય ત્યારે તેનો ઉપયોગ કરી શકાય છે. પછી અમે આદેશ ડેલ જોયું જે કિંમત લે છે અને તે અવ્યાખ્યાયિત છે. ડેલનો ઉપયોગ કરવા માટેની સૌથી સામાન્ય રીત સૂચિ અથવા શબ્દકોશમાંથી કંઈક દૂર કરવા માટે છે, તમે સામાન્ય રીતે ફક્ત નામને અવગણવા માંગતા નથી જે સરળ મૂલ્ય ધરાવે છે. પરંતુ કોઈ શબ્દકોશ અથવા સૂચિમાંથી આપણે કોઈ કીને દૂર કરવા અથવા જો કોઈ સ્થાનને દૂર કરવા માંગતા હોઈએ અને ડેલ તે માટે ખૂબ જ ઉપયોગી છે. છેલ્લે, આપણે જોયું કે વિશિષ્ટ મૂલ્ય કંઈ નથી જે નલ મૂલ્ય સૂચવે છે, તે એક અનન્ય નલ મૂલ્ય છે અને આનો ઉપયોગ ચકાસવા માટે ચલો પ્રારંભ કરવા માટે થઈ શકે છે કે કેમ તે પછી કોડમાં યોગ્ય મૂલ્યો સોંપવામાં આવ્યા છે કે કેમ.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન,
પ્રોફેસર માધવન મુકુન્દ,
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
અઠવાડિયું - 06 લેક્ચર - 01 બેકટ્રેકિંગ

એન ક્વીન્સ ઘણી સમસ્યાઓ માટે, આપણે શક્યતાઓના સમૂહ દ્વારા શોધવાનું છે ઉકેલ શોધવા

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 00:02)

કોઈ સ્પષ્ટ ઉકેલ નથી કે જે અમે સીધા જ પહોંચી શકીએ. તેથી, આપણે તેને શોધવા માટે વ્યવસ્થિત રીતે શોધવાનું છે. અમે એક સમયે એક પગલાથી ઉમેદવાર ઉકેલોનું નિર્માણ કરવાનું ચાલુ રાખીએ છીએ. હવે કદાચ એવું બની શકે કે અમે જે ઉકેલ મેળવવાનો પ્રયાસ કરી રહ્યા છીએ તે કામ કરતું નથી. તેથી, અમે એક મૃત અંત હિટ, અને પછી અમે છેલ્લા પગલું પૂર્વવત્ અને આગામી વિકલ્પ પ્રયાસ કરો. દાખલા તરીકે કલ્પના કરો જો તમે સુડોકુને હલ કરો છો. તેથી, તમારી પાસે ગ્રીડ છે અને પછી તમે વસ્તુઓ ભરવાનું પ્રારંભ કરો છો અને કેટલાક બિંદુઓ છે કે તમે સમજો છો કે તમે કશું મૂકી શકશો નહીં. પછી તમે પાછા જાઓ અને તમને પહેલાં જે કંઈ કર્યું તે બદલવું પડશે. તેથી, આપણે પાછળ પાથરીએ છીએ, આપણે સમસ્યાને હલ કરવાનો પ્રયાસ કરી આગળ જવું પડશે; અને અમુક સમયે જ્યારે આપણે ખ્યાલ રાખીએ છીએ કે અમે અટવાઈ ગયા છીએ તો અમે ફરીથી સમસ્યાનું નિરાકરણ કરી શકતા નથી, આપણે પાછા જવું પડશે અને જે કંઈપણ કર્યું તે પહેલાં આપણે તેને બદલવું જોઈએ અને કંઈક બીજું કરવાનો પ્રયત્ન કરવો પડશે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 00:53)

આ પ્રકારના ક્લાસિક સમસ્યાઓ પૈકી એક આઠ રાણીઓ સમસ્યા કહેવામાં આવે છે. સમસ્યા ચેસ બોર્ડ પર 8 રાણીઓ મૂકવા માટે છે જેથી તેમાંથી કોઈ એકબીજા પર હુમલો ન કરે. હવે, જો તમે ક્યારેય ચેસ રમ્યો હોય, તો તમને ખબર હોત કે રાણી એક વિશિષ્ટ ટુકડો છે જે તે કોઈ પંક્તિ, સ્તંભ અથવા વિકર્ણ સાથે કોઈપણ વર્ગને ખસેડી શકે છે - દાખલા તરીકે, જો આપણે રાણીને અહીં ત્રીજી હરોળમાં મૂકીએ અને ત્રીજા કોલમ, પછી તે ત્રીજા પંક્તિ પર ગમે ત્યાં ડાબે અથવા જમણે ત્રીજા કોલમ ઉપર ખસેડી શકે છે, અને બે ત્રાંસા સાથે, જેના પર ચોરસ ત્રણ અલ્પવિરામ ત્રણ ખોટા છે કારણ કે તે આ કોલમ્સ સાથે આગળ વધી શકે છે કારણ કે આ પંક્તિઓ સાથે આવેલ કોઈપણ ભાગને પણ પકડી શકે છે. આ તમામ ચોરસ પર હુમલો કરવા માટે રાણીને કહેવામાં આવે છે. રાણી ખસેડી શકે તેવા ચોરસ રાણી દ્વારા હુમલો કરવામાં આવે છે. તેથી, અમારો ધ્યેય રાણીઓ રાખવી એ છે કે, તેઓ એકબીજા પર હુમલો કરતા નથી, તેથી જો આપણી પાસે રાણી હો, તો આપણે કોઈ અન્ય રાણીને કોઈ પણ લાલ ચોકમાં મૂકી શકતા નથી, આપણે તેને બીજે ક્યાંય મૂકવી પડશે. દાખલા તરીકે, આપણે નવી રાણી મૂકી શકીએ છીએ; દાખલા તરીકે, આ બરાબર હશે; અથવા અહીં. અને પછી, જો હું અહીં રાણી મૂકીશ તો તે વધુ ટુકડાઓ પર આક્રમણ કરશે, તે આ ચોરસ પર હુમલો કરશે અને તમે કેટલાક વધુ વિકલ્પો બહાર કાઢશો જેથી હું ત્યાં રાણીઓ મૂકી શકશો નહીં. તેથી, આપણે જોઈ શકીએ છીએ કે આપણે

8 રાણીઓ મૂકી શકીએ છીએ. હવે અમે 8 થી વધુ રાણીઓ મૂકી શકતા નથી; કારણ કે, જો તમે 9 રાણીઓ મૂકો તો ફક્ત 8 પંક્તિઓ છે, 2 એ જ પંક્તિ અથવા તે જ સ્તંભમાં અને તે જ સ્તંભમાં હશે. તેઓ એકબીજા પર હુમલો કરવો પડશે. તેથી, 8 સ્પષ્ટપણે મર્યાદા છે, પ્રશ્ન એ છે કે આપણે ખરેખર 8 મૂકી શકીએ?

(સ્લાઇડ સમયનો સંદર્ભ લો: 02:28)

તેથી, અમે આ પ્રશ્નને સામાન્ય બનાવતા હોઈએ છીએ અને 8 માટે નથી માગતા, પરંતુ એન. ધારો કે મારી પાસે ચેસબોર્ડ છે જેમાં એન પંક્તિઓ અને N કોલમ છે. શું હું આવા ચેસબોર્ડ પર એન રાણીઓ મૂકી શકું? હવે એક માટે N બરાબર પ્રશ્ન તુચ્છ છે, કારણ કે તમારે 1 ચોરસ પર 1 રાણી મૂકવો પડશે. હવે, એ જોવાનું સરળ છે કે એન 2 બરાબર અશક્ય છે, કારણ કે જો મારી પાસે 2 ચોરસ હોય અને ગમે ત્યાં, હું રાણીને કહું છું કે અહીં તે બાકીના ચોરસ પર હુમલો કરશે. કોઈ બાબત જ્યાં હું રાણી મૂકી નથી, દરેક અન્ય ચોરસ પંક્તિ, કોલમ અથવા તે રાણી વિકર્ણ પર હશે. અને તેથી બીજી રાણી મૂકવાની કોઈ શક્યતા નથી. તે તારણ આપે છે કે ત્રણ પણ અશક્ય છે. ધારો કે અમે ટોચની ડાબા ખૂણામાં રાણી મૂકીને શરૂ કરીએ છીએ પછી આપણે જોશું કે તે પ્રથમ સ્તંભ, પ્રથમ પંક્તિ અને મુખ્ય ત્રાંસાને અવરોધે છે. આ બીજી રાણી માટે બે સ્લોટસ ખોલે છે, પણ આપણે જ્યાં પણ મૂકીએ છીએ, તેમાંથી બેમાંથી જે પણ આપણે મૂકીએ છીએ, તે અન્ય એકને અવરોધિત કરશે. એકવાર અમે તે સ્લોટમાંથી એક રાણી મૂકી દઈએ તો એક જ ત્રાંસા પર છે અને ત્રીજી રાણી માટે કોઈ મુક્ત સ્લોટ નથી. તેથી, સંપૂર્ણ વિશ્લેષણ દ્વારા આપણે તે બતાવી શકીએ કે, ત્રણ બરાબર એ ખરેખર અશક્ય છે N માટે 4 માટે 4 4 4 દ્વારા 4 બોર્ડ, તે શક્ય છે બહાર ચાલુ. અમે ખૂણા પર શરૂ ન થવું જોઈએ, પરંતુ ખૂણાઓ એક. ધારો કે આપણે તેને બીજા સ્તંભમાં મૂકીએ છીએ, તો પછી આપણે આ સ્ક્વેરના બ્લોક સ્ક્વેરને મેળવીએ છીએ. પછી આપણે અંતે બીજી પંક્તિ પર ખાલી સ્લોટ શોધી શકીએ છીએ. તેથી, અમે એક રાણી મૂકી છે ત્યાં તે છેલ્લા સ્તંભના અમુક ચોરસમાં અને તે કર્ણમાં ચોક્કસ બ્લોકો છે, પરંતુ આ ત્રીજી પંક્તિમાં હજી એક સ્લોટને છોડી દે છે, દુર્ભાગ્યે ત્રીજા રાણી ચોથા પંક્તિ પર છેલ્લા બે સ્લોટને અવરોધિત કરતી નથી અને અમારી પાસે આ પ્રકારના સપ્રમાણ પેટર્ન છે જ્યાં બધું એક ખૂણામાં છે જેમાં રાણીઓ કોઈ એકબીજા પર હુમલો કરે નહીં. હવે, તે તારણ આપે છે કે એક વાર આપણે 5, 6, 7, 8 માટે એન બરાબર 4 વડે પસાર કરી શકીએ, તમે બતાવી શકો કે ઉકેલ શક્ય છે. અમારું કાર્ય આવા ઉકેલ શોધવાનું છે. N માટે 4 અથવા તેથી વધુ સમાન માટે ઉકેલ કેવી રીતે શોધી શકીએ?

(સ્લાઇડ સમયનો સંદર્ભ લો: 04:24)

તેથી, જેમ તમે જોયું તેમ, પ્રથમ વસ્તુ જે તમને ખબર છે તે પ્રથમ છે કે દરેક પંક્તિ અને દરેક કોલમમાં બરાબર એક રાણી હોઈ શકે છે કારણ કે રાણીઓ સ્તંભ અને પંક્તિ પર હુમલો કરે છે જેના પર તેઓ આવેલા છે. જો અમારી પાસે એક જ પંક્તિ અથવા તે જ સ્તંભ પર બે રાણીઓ હશે તો તેઓ એકબીજા પર હુમલો કરશે. 8 થી ચેસબોર્ડનું ક્લાસિકલ કદ છે, ચાલો આપણે 8 રાણીઓ માટે ખાસ કરીને અમારા ઉદાહરણને જોઈએ. તેથી, આપણે પંક્તિઓ દ્વારા હરોળને હવે પંક્તિઓ મૂકવા માંગીએ છીએ. અમે જાણીએ છીએ કે દરેક પંક્તિમાં બરાબર એક રાણી છે ચાલો પહેલા રાણીને પ્રથમ પંક્તિમાં મૂકીએ, પછી તેના આધારે બીજી રાણીમાં રાણી મૂકવામાં આવે છે અને તે જ રીતે આપણે 4 બી 4 કેસ માટે કર્યું છે જે આપણે પાછલી સ્લાઇડમાં જોયું હતું. તેથી, દરેક પંક્તિમાં અમે રાણીને પ્રથમ ઉપલબ્ધ સ્તંભમાં મૂકી દઈશું, જે રાણીઓને પહેલેથી જ મૂકવામાં આવી છે,

ઉપલબ્ધ દ્વારા અત્યાર સુધી અમે એવા ચોરસનો અર્થ રાખીએ છીએ જેનો અત્યાર સુધી હુમલો થયો નથી. તેથી, અમે 8 બાય 8 બોર્ડથી શરૂ કરીએ છીએ અને પ્રથમ પંક્તિમાં હવે બધું ઉપલબ્ધ છે. અમારા વિશ્લેષણ દ્વારા આપણે રાણીને પ્રથમ ઉપલબ્ધ સ્તંભમાં મૂકીશું, એટલે કે આપણે તે પછી ટોચની ડાબી બાજુએ; તે પ્રથમ પંક્તિ અને કોલમ અને મુખ્ય કર્ણ બહાર અવરોધે છે. તેથી, બધા શેડમાં ચોક હવે હુમલો હેઠળ છે. અમે બીજી હરોળમાં ખસેડીએ છીએ અને અમે રાણીને પ્રથમ ઉપલબ્ધ સ્તંભમાં મૂકવાનો પ્રયાસ કરીએ છીએ જે આ ત્રીજા ભાગ છે અને આ બદલામાં પંક્તિઓ, કોલમ અને વિકર્ણ વર્ગના અન્ય સમૂહ પર હુમલો કરશે. હવે, અમે ત્રીજા પંક્તિ તરફ જઈએ છીએ અને 5 મી સ્તંભમાં આપણે રાણી મૂકી શકીએ છીએ. અને આ ફરી એકવાર કેટલાક ચોરસ પર હુમલો કરે છે. તેથી, અમે સૂચવવા માટે કેટલાક રંગો ઉમેર્યા છે, કારણ કે નવી રાણી દ્વારા દરેક નવા રાણીને મૂકવામાં આવે છે, તેમાંના કેટલાંક નવા રાણી દ્વારા હુમલો કરવામાં આવે છે, તેમાંના કેટલાકને ઘણી રાણીઓ દ્વારા હુમલો કરવામાં આવે છે. દાખલા તરીકે, પીળા રાણી કર્ણ પર વાદળી ચોરસ પર હુમલો કરે છે, જે પહેલા રાણી દ્વારા પહેલેથી હુમલો કરાયો હતો. તો, આપણે હવે તે વાદળી છોડી દઈશું. આ રીતે આપણે આગળ વધારી શકીએ છીએ. તેથી, અમે 4 થી પંક્તિ પર 4 થી રાણી મૂકીએ છીએ, અને પછી આ એક ભૂલ છે કે આ પહેલાથી જ આ રાણી દ્વારા હુમલો કરવો જોઈએ અને પછી અમે 5 મી રાણી મૂકીશું અને પછી 6 થી અને ત્યારબાદ 7 મી અને હવે આપણે તે બધા 8 મી પંક્તિના વર્ગ ખરેખર અવરોધિત છે. 8 મી રાણીને મૂકવા માટે આ ઉકેલ લાવવાનો કોઈ રસ્તો નથી. તેથી, આ વિશે કંઈક કરવું પડશે, અમે રાણીને 8 મી પંક્તિમાં મૂકી શકતા નથી.

(સ્લાઈડ ટાઇમ નો ઉલ્લેખ કરો: 06:36)

આપણે 8 મી પંક્તિમાં રાણી સાથે સ્થાન ન મૂકી શકીએ તેથી આપણે પાછા જવું જોઈએ અને હવે પહેલાં જે કંઈ કર્યું તે બદલવું પડશે. અમે જે છેલ્લી વસ્તુ હતી તે 7 મી રાણીની જમણી બાજુ મૂકવા હતી. તેથી, અમે તે કરીએ છીએ અને કમનસીબે 7 થી રાણીની શોધમાં અમારો એક વિકલ્પ હતો. તેથી, અમારી પાસે 7 મી રાણી માટે કોઈ અન્ય વિકલ્પ નથી. જો 7 મી રાણી કોઈ ઉકેલ તરફ દોરી શક્યો ન હતો, પણ તે 7 મી રાણીની પસંદગી ન હતી, જેણે વાસ્તવમાં સમસ્યા ઊભી કરી હતી, પરંતુ તે અગાઉની વાત હતી. પછી અમે પાછા જાઓ અને 6 થી રાણી ખસેડવાનો પ્રયાસ કરો. તેથી, ફરી એક વખત જો તમે 6 થી રાણીને દૂર કરો તો આ થોડા ચોરસને તોડી નાંખે છે, પરંતુ તે જ સમયે 6 થી પંક્તિ પર 6 થી રાણી મૂકવા માટે કોઈ અન્ય જગ્યા નથી. તેથી, ફરી આ એક અનન્ય પસંદગી હતી જે અમે બનાવી હતી. હવે જો આપણે 5 મી રાણી પર પાછા જઈએ તો આપણે શોધી કાઢીએ કે 5 મી રાણી મૂકવાનો માર્ગ છે. એક અલગ સ્થાને એટલે કે તે આ સ્લોટમાં ખસેડશે. તેથી, અમે આ 5 મી રાણીને એક સ્લોટને જમણે ખસેડી શકીએ છીએ અને ફરીથી પ્રયાસ કરી શકીએ છીએ. તેથી, 8 મી ચોરસથી પાછો ફર્યો અને, 8 મી પંક્તિ જે સંપૂર્ણપણે બ્લોક કરવામાં આવી છે, તે 7 મી પંક્તિની છે જે ફક્ત એક જ વિકલ્પ છે, જે 6 થી પંક્તિની છે, જે ફક્ત એક જ વિકલ્પ છે જે અમે 5 મી પંક્તિ પર પાછા આવીએ છીએ અને હવે અમે 5 મી પંક્તિ માટે આગામી પસંદગી. જો આપણે 5 મી પંક્તિ માટે આગળની પસંદગી કરવાનો પ્રયાસ કરીએ - તો આપણે ચોરસની આ પેટર્ન મેળવીએ છીએ અને હવે આપણે ઉદાહરણ તરીકે જોઈએ છીએ, કે આપણે 6 થી વસ્તુ મૂકી શકતા નથી. તેથી, 5 મી પંક્તિ માટેની બંને પસંદગીઓ ખરેખર ખરાબ હોવાનું ચાલુ કરે છે. તેથી, તમારે હવે પાછા જવું પડશે અને 4 થી હરોળ માટે અલગ પસંદગી કરવાનો પ્રયાસ કરવો પડશે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 07:55)

જો આપણે પહેલાની ચાલને પૂર્વવત્ કરી ન શકીએ અને ફરી પ્રયાસ કરીએ તો, આપણે આગળના પગલાંનો ઉકેલ વધારવાનો પ્રયાસ કરીએ છીએ, અને આ રીતે અમે સંપૂર્ણ રીતે તમામ સંભવિત ઉકેલો, પરંતુ અમે તે એક વ્યવસ્થિત રીતે કરીએ છીએ કે આપણે પાછા ન જઈએ અને ચોક્કસપણે એક પગલે પાછા ફરો અને અગાઉના પગલાંને પૂર્વવત્ કરીએ તે પહેલાં અમે કેટલીક પસંદગીઓને ફેરબદલ કરીએ છીએ. તેથી, દરેક પગલામાં અમારી પાસે ઘણી બધી પસંદગીઓ છે, જે આપણે તેમને વ્યવસ્થિત રીતે લઈએ છીએ, દરેક વિકલ્પ માટે અમે ઉકેલ વિસ્તૃત કરવાનો પ્રયાસ કરીએ છીએ જો ઉકેલ વિસ્તૃત ન થાય તો અમે પાછા આવીએ છીએ અમે આગળની પસંદગીનો પ્રયાસ કરીએ છીએ અને જ્યારે આ સ્તરે તમામ પસંદગીઓને એક્ઝોસ્ટ કરીએ છીએ અમે પાછલા સ્તરની જાણ કરીએ છીએ કે અમે નિષ્ફળ ગયા પછી તેઓ તેમની આગામી પસંદગી અને તેથી આગળ પ્રયાસ કરશે. બેક્ટ્રેકિંગની ચાવી એક સમયે એક સ્તર આગળ અને પાછળની તરફ જઈને તમામ શક્યતાઓ દ્વારા વ્યવસ્થિત શોધ કરવાનું છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 08:47)

તો, આપણે આ પ્રકારનાં અભિગમને કેવી રીતે એન્કોડ કરીશું? ખાસ કરીને, 8 રાણીઓની સમસ્યા માટે, તેથી અમારું પ્રથમ પ્રશ્ન એ છે કે બોર્ડનું પ્રતિનિધિત્વ કેવી રીતે કરે છે કારણ કે બોર્ડ તે છે જે આપણે ચાલને બદલીએ છીએ અને તેમને પૂર્વવત્ કરીએ છીએ. એન રાણીના ઉકેલ માટેનો સૌથી સ્પષ્ટ રસ્તો શાબ્દિક રીતે N ગ્રીડ દ્વારા એન તરીકે રજૂ કરે છે. અને ત્યારથી પાંચથોન નંબર્સની સૂચિ પદ પરથી 0 ની N ગ્રીડ દ્વારા N છે અને આપણે 1 થી N ના કોલમની સંખ્યા, પરંતુ 0 થી N ની 1 ની સંખ્યા, તેથી તેમની પંક્તિઓ 0 થી N ની 1 ઘાત અને કોલમ 0 થી N ઓછા 1 હશે. હવે આપણે મૂલ્ય 1 અથવા 0 અથવા સાચા કે ખોટા દર્શાવવા માટે કરી શકીએ છીએ કે ચોરસમાં હું રાણી છું કે નહીં; હું એક પંક્તિ છે, j એ સ્તંભ છે. તેથી, આપણી પાસે બે પરિમાણીય સૂચિ, બોર્ડ અથવા ચાદીઓની સૂચિ હોઈ શકે છે, જેમાં એન બાદ 1 નો N બાદ 1, 0 થી N ઓછા 1 અને 0 થી N ઓછા 1 માન્ય સૂચકાંકો તરીકે છે અને અમે કહીએ છીએ કે બોર્ડ આઈજે 1 છે સૂચવે છે કે રાણી હું અલ્પવિરામ જે. અને તેથી જો તે 0 હોય તો તે દર્શાવે છે કે કોઈ રાણી નથી. દરેક ચોરસ માટે બે શક્ય મૂલ્યો છે. અલબત્ત, આપણે પણ જાણીએ છીએ કે પંક્તિ દીઠ માત્ર એક રાણી છે આ ચોક્કસ બાબત છે કે એન એનસ સ્ક્વેર એન્ટ્રીઓમાં એન બાદ એન નો સમાવેશ થાય છે, તે કોઈ પણ સમયે ફક્ત એન હશે. તેથી, આપણે 0 થી N ની 1 ની એન્ટ્રી સાથે ફક્ત એક સૂચિ બનાવીને સહેજ ઓપ્ટિમાઇઝ કરી શકીએ છીએ જ્યાં આપણે કહીએ છીએ કે i th એન્ટ્રી i th પંક્તિ સાથે સંકળાયેલ છે અને આપણે સ્તંભ નંબર રેકોર્ડ કરીએ છીએ. તેથી, જો બોર્ડ ઓફ આઈ બરાબર છે તો તેનો અર્થ એ છે કે હારમાળામાં રાણી સ્તંભ j છે. રાણી પોઝિશન i અલ્પવિરામ જે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 10:36)

તેથી, આવા ડેટા માળખું સાથે આ અમારી વ્યૂહરચના કેવી રીતે કાર્ય કરે છે તે રૂપરેખા છે. તેથી, આપણે શું કરવું છે તે દરેક રાણી એક સમયે એક જ મૂકશે. તેથી, આપણે ફક્ત એક કાર્ય લખી રહ્યા છીએ જે સળંગમાં રાણી મૂકવાનો પ્રયાસ કરે છે અને મેં બોર્ડની વર્તમાન સ્થિતિને આપેલ છે. તેથી, આપણે તેને એક દલીલ તરીકે બોર્ડની હાલની સ્થિતિને પસાર કરીએ છીએ અને અમે તે પંક્તિ નંબરને પસાર કરીએ છીએ જે અમે કરવા જઈ રહ્યા છીએ. તેથી, અમે શરૂઆતમાં ખાલી બોર્ડ સાથે અને 0 ના દાયકાથી શરૂ કરીશું. હવે અમે દરેક સ્તંભમાં દોડીએ છીએ અને તપાસ કરીએ છીએ કે પંક્તિ કોલમ પોઝિશન જે

ચોરસ આઠ કોમા સી છે તે ઉપલબ્ધ છે, જો ઉપલબ્ધ હોય તો અમે ત્યાં રાણી મૂકીશું અને અમે અલબત્ત, બોર્ડ સુધારવા માટે છે તેથી, અમે એક મિનિટમાં પાછા આવીશું, પરંતુ અમારા કિસ્સામાં અમારા બોર્ડને સુધારવાનો અર્થ ફક્ત બોર્ડની સેટિંગ એટલે કે 1 ની સમાન હોય છે જો અમારી પાસે એક પરિમાણીય પ્રતિનિધિત્વ હોય. હવે, જો આપણે ખરેખર છેલ્લી રાણી મૂકી, જો હું એન બાદ 1 હોત તો આ છેલ્લો રાણી જમણી છે. તેથી, જો તે 8 રાણીની સમસ્યા છે, તો જ્યારે અમે 0 થી શરૂ કરતા રાણી નંબર 7 મૂકીશું તો અમે કરીશું. તેથી, અમે સાચા પાછા આવી શકીએ છીએ; જો કે, જો આ છેલ્લી રાણી નથી, તો અમારે ચાલુ રાખવો પડશે. તેથી, અમારે શું કરવાની જરૂર છે હવે એક વધુ રાણી મૂકવા માટે નવા બોર્ડ સાથે. તેથી, આપણે આ ફંક્શનને પંક્તિથી 1 ને વત્તા એક સાથે સુધારિત બોર્ડ સાથે બોલાવીએ છીએ જે આપણે હમણાં મૂકી છે અને આ સાચું કે ખોટું આપણે, તેના આધારે કે તે સફળ થાય છે કે નહીં. તેથી, આપણે રેકોર્ડ કરીએ છીએ કે નામ પરત આપવાની પ્રક્રિયામાં વળતર મૂલ્ય છે. તે સફળ થાય છે કે નહીં તે અમે તેના પર નિર્ભર છીએ કે જો વિસ્તરણ ઉકેલ એ સાચું છે કે વર્તમાન સ્થિતિ અંત સુધી પહોંચે છે. હવે, ક્યારે તે સાચું હશે; જો તે N નીબાજુ 1 અને એન બાદ 1 ની બધી રીત તરફ જવા માટે સફળ થઈ તેથી, જ્યારે N બાદ 1 એ સાચું આવે છે ત્યારે એન બાદ 2 સાચું પાછું આપણે. પછી અમારા સ્તરે હું પણ કિંમત સાચું મળશે. પછી અમે પણ સાચું પાછા આવી શકી છો. તેથી, જો ઉકેલને વળતર આપવામાં આવે તો અમે સાચા કહીએ છીએ કે અત્યાર સુધી હું સારી છું. બીજી તરફ જો ઉકેલ લાવવી તે ખોટા વળતર આપે છે તો તેનો અર્થ એ છે કે વર્તમાન સ્થિતિને મેં પંક્તિ I માટે પસંદ કરી છે, આને પૂર્ણ ઉકેલ માટે વિસ્તારવા માટે વધુ કંઈ કરવાનું નથી. આ સ્થિતિ પૂર્વવત્ કરવી જોઈએ તેથી, આપણે આ ચાલને પૂર્વવત્ કરવો પડશે. તેથી, બોર્ડમાં અપડેટ કરવા પહેલાં અમે જે કર્યું તે આપણી પાસે છે. આ અપડેટને આ બિંદુએ ઉલટાવી શકાય છે. તેથી, આપણે આઠસી પર મૂકવાની અસરને ઉલટાવી છે અને પછી, જ્યારે આપણે આ કરીશું આપણે પાછા જઈશું અને આપણે આગામી સીને અજમાવીશું અને હવે જો આપણે ખરેખર તમામ સી દ્વારા ચલાવીશું અને અમે કોઈપણ સમયે સાચું પાછી નથી કર્યું, પછી પાચથોન આ બીજું છે જે કહે છે કે લૂપ માટે વચ્ચે વચ્ચે બહાર આવતા વગર સમાપ્ત થાય છે. આ માટે લૂપ સામાન્ય રીતે સમાપ્ત થાય છે એનો અર્થ એ કે અમે ઉપલબ્ધ દરેક શક્ય સી દ્વારા ચલાવી છે અને તેમાંથી કોઈએ સાચું કર્યું નહીં; તેનો મતલબ એવો થાય છે કે હાલમાં કોઈ પણ રાણી પર પંક્તિ મૂકવાની કોઈ રીત નથી જે મેં આપેલ બોર્ડને આપેલ છે. તેથી, આપણે ખોટા કહેતા પાછા જવું જોઈએ કે જે બોર્ડ અમને મળી છે તે સારી નથી, પછીની પંક્તિ હવે ખોટી બની જશે અને અમે આગળની સ્થિતિ અને તેથી આગળ પ્રયાસ કરીશું. આ એક રિકર્સિવ ઉકેલ છે કે જે આપણને મળે છે અમે એક વાસ્તવિક પાચથોન એક્ઝેક્યુશનને જોશું, પરંતુ વાસ્તવમાં આ કેવી રીતે અમલમાં મૂકવું તે સમજવા માટે અમારે થોડો વધુ કાર્ય કરવું પડશે. અમલીકરણમાંની મહત્વની બાબત એ છે કે આપણે પહેલાંના એકને જોયું છે, કે જ્યારે આપણે રાણી રાખીએ છીએ ત્યારે બોર્ડને અપડેટ કરવું પડશે અને બોર્ડને અપડેટ કરવું જોઈએ જ્યારે અમે તેને પૂર્વવત્ કરીએ છીએ અને અમે પણ તપાસ કરવી પડશે કે શું આઠસી ઉપલબ્ધ છે કે નહીં.

(સ્લાઈડ ટાઇમ નો ઉલ્લેખ કરો: 14:01)

તેથી, અમે બે પ્રતિનિધિઓ, 0 સે અને રાશિઓ સાથે એક બે પરિમાણીય પ્રતિનિધિત્વ અને એક ડાયમેન્શનલ પ્રતિનિધિત્વ કર્યું હતું જે અમને દરેક પંક્તિ માટે સ્તંભની સ્થિતિ આપે છે જે બોર્ડમાં રાણીઓનું ધ્યાન રાખે છે, પરંતુ ચોરસ મુક્ત છે કે નહીં તે નિર્ધારિત કરવા માટે ક્રમમાં, અમે ગણતરી કરવા માટે વધુ સારી રીત હોવા જોઈએ કે કેવી રીતે ચોરસ રાણીઓ દ્વારા હુમલો કરવામાં આવે છે. એક સરળ રીત ફક્ત એમ કહી શકાય કે બોર્ડના બે

પરિમાણીય પ્રતિનિધિત્વની સાથે અમે બતાવીએ છીએ કે ઉદાહરણ તરીકે આપણે ચિત્રાત્મક રીતે કામ કર્યું છે, આપણે જે રીતે કામ કર્યું છે તે આપણે આ પ્રકારની રંગીન ચોરસ તરીકે ઓળખાવ્યા છે તે દર્શાવવું કે ચોરસ હુમલો કરવામાં આવે છે. તેથી, અમે કહીએ છીએ કે હુમલો ij 1 છે, જો તે રાણી દ્વારા હુમલો કરવામાં આવે તો અન્યથા તે 0 છે. હવે તેની સાથે સમસ્યા એ છે કે આપેલ ચોરસ આઈજે પર એકથી વધુ રાણી અધિકાર દ્વારા હુમલો કરી શકાય છે. તેથી, જ્યારે આપણે રાણીને પૂર્વવત્ કરીએ છીએ, તે સ્પષ્ટપણે, ઘણા ચોરસ પર હુમલો કરશે, પરંતુ તે તમામ ચોરસ તે રાણીને દૂર કરીને મુક્ત નહીં થાય, કારણ કે કેટલાક ચોરસ અન્ય રાણીઓ દ્વારા પણ હુમલો કરવામાં આવે છે, જે અમે અગાઉ મૂક્યા હતા. તેથી, અમને સાવચેત રહેવાની જરૂર છે, જ્યારે ચોવીસ ચિહ્નિત કરવા માટે રાણીને દૂર કરવામાં આવે છે, જે મફત તરીકે હુમલો કરવામાં આવ્યા હતા. ઠીક છે, આમ કરવા માટેનો એક રસ્તો વાસ્તવમાં છે, રાણીઓની સંખ્યા અને પ્રારંભિક રાણી છે જે દરેક ચોરસ પર હુમલો કરે છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 15:28)

તેથી, અમે કહીએ છીએ કે હુમલો ij એ k છે જો ij પર પ્રથમ રાણી દ્વારા હુમલો કરવામાં આવ્યો અને ij એ હુમલો એકથી ઓછો છે જો ij મફત છે તેથી, જ્યારે અમે રાણી k ને દૂર કરીએ છીએ, તો વેલ્યૂ k થી બાદ 1 પર હુમલો ij રીસેટ કરીએ છીએ અને અન્ય તમામ ચોરસ હજુ પણ અગાઉ રાણીઓ દ્વારા હુમલો કરવામાં આવે છે. તેથી, આપણે તે ચિત્ર સાથે ખૂબ જ સહેલાઈથી સમજાવી શકીએ છીએ જે આપણે પહેલા કર્યું હતું.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 15:58)

અહીં આપણે વાદળી રાણી મૂકી ત્યારે અમારી બોર્ડનું પ્રતિનિધિત્વ કર્યું છે કે આપણે વાદળી રાણીના તમામ ચોકને વાદળી સોલિડ ચોરસ જેવા વાદળી પર હુમલો કર્યો છે, જ્યારે અમે લાલ રાણી મૂકીએ છીએ ત્યારે અમે ફક્ત ત્યારે જ હુમલો કરીએ છીએ અમે લાલ તે ચોરસ નવા ચોરસ સાથે ચિહ્નિત થયેલ છે જે ઉદાહરણ તરીકે હુમલો કરવામાં આવે છે, આ ચોક્કસ સ્ક્વેર, જે લાલ અને વાદળી બંને દ્વારા હુમલો કરવામાં આવ્યો છે તે પહેલા વાદળી દ્વારા હુમલો કરવામાં આવ્યો હતો તેથી, અમે તેને ચિહ્નિત કર્યું નથી. તેથી, દરેક નવી રાણી સાથે આ રીતે હું અમે મૂકી છે કે અમે માત્ર રાણી દ્વારા હુમલો કરવામાં આવે છે જે ચોરસ ચિહ્નિત કરો. અહીંના રંગો રાણી નંબરોનું પ્રતિનિધિત્વ કરે છે. વાદળી ચોરસ રાણી 0 છે, લાલ ચોરસ રાણી છે, પીળા ચોરસ રાણી બે અને તેથી વધુ છે. તેથી, દાખલા તરીકે, જ્યારે આપણે આને રદ કરવા માટે આવે છે, ત્યારે હવે આપણે આ ખાસ વસ્તુને પૂર્વવત્ કરવા માંગીએ છીએ, જ્યારે આપણે તેને મૂકીએ ત્યારે માત્ર એક સફેદ ચોરસ હતું, આ સિવાયના કોઈ મુક્ત ચોરસ ન હતા. તેથી, અમે કોઈ નવા હુમલાનો ઉમેરો કર્યો નથી. તેથી, તેને દૂર કરવું એ હુમલોની સ્થિતિ અંગે વાસ્તવમાં કંઈ બદલી નાંખતું નથી માત્ર તે જ ચોક્કસ ચોરસથી મુક્ત કોઈ અન્ય સ્ક્વેરને છૂટી પાડતું નથી. હવે, જ્યારે આપણે આ નારંગી રાણીને દૂર કરીએ છીએ, ત્યારે આપણે આ નારંગી ચોરસને દૂર કરવી પડશે જે આ રાણીને ઉમેર્યા પછી જ હુમલો હેઠળ મૂકવામાં આવે છે અને તે નીચેની પંક્તિ પર આ બે હોવાનું બહાર આવે છે. તેથી, જ્યારે આપણે આને પૂર્વવત્ કરીએ છીએ, ત્યારે આપણે તે બેને પૂર્વવત્ મેળવીશું. એ જ રીતે જ્યારે આપણે જાંબલીને પૂર્વવત્ કરીએ છીએ તેથી, અમે જે કરી રહ્યા છીએ તે ખરેખર ચોક્કસ છે કે કેવી રીતે વસ્તુઓને કેવી રીતે રાખવી તે કેવી રીતે કરવી તે કેવી રીતે કરવી તે આક્રમણ હેઠળ છે.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 17:22)

તેથી, હવે અમે એટલેક રાખીશુંકે એ એરે જે કહે છે કે હુમલો i, j એ K છે, જો તે પ્રથમ રાણી કે દ્વારા હુમલો કરવામાં આવે છે અને જ્યારે અમે રાણી k ને દૂર કરીએ છીએ, તો અમે એમ કહીએ છીએ કે, ચોરસ એ ચોક્કસપણે મુક્ત છે જો મૂલ્ય હાલમાં છે. હવે આ માત્ર મુશ્કેલી જ છે કે તે એન ચોરસ અવકાશની જરૂર છે, આપણે જોયું કે આપણે બોર્ડને રેખિક વસ્તુથી 0 થી અને એરો સાથે N દ્વારા એક રેખીય વસ્તુથી બદલી શકીએ છીએ, આપણે તે એક એરે દ્વારા બદલી શકીએ છીએ જે બોર્ડ પર હતા. સમાન હોઈ પ્રશ્ન એ છે કે આપણે એક સુરેખ એરે દ્વારા હુમલોને બદલી શકીએ છીએ, હવે યાદ રાખવું એક બાબત એ છે કે જોકે હુમલો પોતે એન સ્ક્વેર્ડ એરેનો હુમલો છે, હુમલાને પૂર્વવત્ કરવાની જરૂર નથી કારણ કે એકવાર અમે રાણીને ઠીક ઠીક કરવા માટે તમામ એન સ્ક્વેર્ડ એન્ટ્રીઓ જોવાની જરૂર નથી. પૂર્વવત્ કરો, આપણે ફક્ત તેની પંક્તિ, સ્તંભ અને ત્રાંસા સાથે જોવાની જરૂર છે અને તે પંક્તિના સ્તંભ અને કર્ણ પરની બધી રાણીઓની કિંમત સાથેની તમામ એન્ટ્રીઝને દૂર કરો. અપડેટ્સ કોઈ સમસ્યા નથી કે જે સુધારાઓ રેખીય છે, રાણી ઉમેરી રહ્યા છે અને દૂર કરવા માટે માત્ર આ એરેમાં કોષોની એક રેખીય સંખ્યા જોવાની જરૂર છે, પરંતુ એરે પોતે વર્ગાત્મક છે, તેથી અમે ફક્ત ઓર્ડર એન સ્પેસનો ઉપયોગ કરવા માટે અમારા પ્રતિનિધિત્વને સુધારી શકીએ છીએ. .

(સ્લાઈડ ટાઇ નો ઉલ્લેખ કરો: 18:35)

આ કરવા માટે આપણે સમસ્યાની નજીક થોડું નજીક જોવું પડશે. તેથી, કેટલા રાણીઓ અમે હરોળ પર હુમલો કરીએ છીએ જો આપણે હરોળને જોતા રહીએ છીએ તો આપણે યાદ રાખીએ છીએ કે દરેક હરોળમાં એક રાણી અને દરેક સ્તંભમાં તેથી, હરોળ પરની રાણી હું વાસ્તવમાં પંક્તિ પર હુમલો કરું છું, તેવી જ રીતે માત્ર એક જ રાણી કોલમમાં છે. તેથી, સ્તંભ જમાની માત્ર રાણી છે જે તે સ્તંભ પર હુમલો કરે છે. જો આપણે વ્યક્તિગત સ્ક્વેરને જોવો તો, જો આપણે દાખલા તરીકે આ કેન્દ્રમાં હોવ તો આ ચોક્કસ ચોરસને 4 દિશાઓથી હુમલો કરી શકાય છે, તે સ્તંભમાંથી હુમલો કરી શકાય છે કે જેમાં તે અથવા તે પંક્તિ છે કે જેમાં તે છે અથવા તે આ મુખ્ય કર્ણ અથવા બંધ કર્ણ પરથી હુમલો કરવામાં આવશે. મુખ્ય ત્રિકોણ એ ટોચની ડાબી બાજુમાંથી એક છે જે ઉત્તર-પશ્ચિમ અને એક કહેવાય છે, દક્ષિણ દિશામાંથી આવતી કર્ણ છે. ત્યાં 4 શક્ય રાણીઓ છે જે આ સ્ક્વેર પર હુમલો કરી શકે છે. ત્યાં 4 દિશાઓ છે જેમાં એક ચોરસ હુમલો હેઠળ હોઈ શકે છે. આ ચાર દિશાઓને બદલે ચોરસની પ્રતિનિધિત્વ કરવાનું વધુ સારું હોઈ શકે છે, આપણે જે પ્રતિનિધિત્વ કર્યું છે તે હવે કહેવાનું છે કે આ ચોક્કસ ચોરસ રાણી કે દ્વારા હુમલો કરવામાં આવે છે, પરંતુ તે અમને કઈ દિશામાં રાણી કીએ અધિકાર પર હુમલો કરી રહ્યું છે તે કહો નથી અમને જણાવો કે રાણી કે તે પંક્તિ અથવા સ્તંભ અથવા કર્ણમાંથી હુમલો કરી રહી છે.

(સ્લાઈડ સમય જણાવો: 20:05)

તેથી, પંક્તિઓ અને કોલમ્સ કુદરતી રીતે 0 થી 7 સુધી ગણવામાં આવે છે, પરંતુ કર્ણ વિશે શું? હવે જો આપણે ઉત્તર પશ્ચિમથી ત્રાંસા પર જોશું તો ચાલો આ દિશાઓ ઉત્તર પશ્ચિમ, દક્ષિણ પશ્ચિમ, ઉત્તર પૂર્વ અને દક્ષિણ પૂર્વમાં કહીએ. જો તમે વિકર્ણ કર્ણમાં ઘટાડાને જોતા હોવ કે જે આનાથી ઉપરથી નીચે સુધી જાય છે, તો આપણે શું શોધીએ છીએ કે આ તફાવત કોલમની બાદબાકી પંક્તિ કંઈક છે જે તે કર્ણ પર દરેક ચોરસ પર સમાન હશે, દાખલા તરીકે, જુઓ આ કર્ણ તે અહીં શરૂ થાય છે અહીં સ્તંભની સંખ્યા 2 છે અને પંક્તિ 0 છે. 2 ઓછા 2 છે, જો આપણે ત્રાંસાની આગળની વસ્તુ પર જઈએ તો 3 ઓછા 1 જે ફરીથી 2 છે તો પછી 4 ઓછા 2 ફરીથી 2 અને તેથી વધુ. તેથી, જો

આપણે આ બધા ચોરસ માટે આ કર્ણ સાથે જઈએ તો c ઓછા r જ્યાં c એ સ્તંભ નંબર છે અને r એ પંક્તિની સંખ્યા છે જે તફાવત 2 બરાબર છે અને તમે ચકાસી શકો છો કે આ ગ્રીડ પરના સ્કવેર પર બીજું કોઈ સાચું નથી, અન્ય ઉદાહરણ તરીકે જો તમે આ ચોક્કસ વસ્તુ જુઓ છો આપણી પાસે 0 ઓછા ચાર છે. આ તફાવત ઓછા છે 4 અને આ જ રીતે 3 ઓછા 7 પણ ઓછા 4 છે. તેથી, આ ચોક્કસ કર્ણ સાથેના દરેકમાં તફાવત 4 ઓછા છે. હવે, જો આપણે અન્ય માર્ગે જવા માટેના કર્ણ જોઈશું તો આપણે શોધી કાઢીએ છીએ કે આ રકમ અવિભાજ્ય છે દાખલા તરીકે, આપણી પાસે 6 વત્તા 0 અથવા 5 વત્તા 1 અથવા 4 વત્તા 2 અને 2 વત્તા 3, 3 વત્તા 3 અને તેથી વધારે છે. તેથી, આ જાંબલી વિકર્ણ સી વત્તા r બરાબર 6 બરાબર છે, અને આ લીલો કર્ણ સાથે અમારી પાસે 7 વત્તા 5, 6 વત્તા 6 અને 5 વત્તા 7 છે. તેથી, c વત્તા r બરાબર 12 છે. તેથી, આપણે હવે પૂર્ણ કરી શકીએ છીએ. કે જે સ્થાને આઈજ પર ચોરસ પર હુમલો કરવામાં આવે છે, જો રાણી દ્વારા અથવા રામની જમણામાં રાણી દ્વારા હુમલો કરાયો હોય અથવા જો તે ત્રાંસા સાથે હોય છે જેના તફાવત એ જ ઓછા છે અથવા જો તે ત્રાંસા સાથે છે જેના તફાવત એ જ વત્તા છે જેની રકમ છે j વત્તા i .

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 22:00)

તો, હવે આપણે એક પ્રતિનિધિત્વ સાથે આવી શકીએ છીએ જે માત્ર હરોળ, સ્તંભો અને કર્ણ પર હુમલો કરે છે અને જેનો હુમલો આવેલો છે અને તેમાંથી આપણે જાણી શકીએ છીએ કે, ચોરસ હુમલો હેઠળ છે. તેથી, અમે કહીએ છીએ કે પંક્તિ 1 એ 1 છે, જો પંક્તિ i એ હુમલો હેઠળ હોય છે જ્યાં i 0 થી એન બાદ 1 ની જેમ જ રેન્જ ધરાવે છે; આપણી પાસે એક એરે હોઈ શકે છે જે કહે છે કે મેં સ્તંભ i પર હુમલો કર્યો છે અને પછી સ્તંભ i ને 1 પૂરા પાડવામાં આવેલ કોલમ પર સેટ કરવામાં આવ્યો છે, જેના પર i ફરીથી 0 અને એન નીચું વચ્ચેનો હુમલો કરું છું. હવે જ્યારે આપણે વિકર્ણને જુઓ, તો આપણી પાસે આ બે પ્રકારના કર્ણ છે. ઉત્તર પશ્ચિમમાં દક્ષિણ પૂર્વ કર્ણ એ એક જ છે જ્યાં તફાવત એ જ છે અને જો તમે તફાવતો જોશો, જો તમે પાછા જાઓ તો તમે અહીં આ કર્ણ પરના તફાવતો જુઓ, તફાવત 7 ઓછા છે 7 અને અહીં તફાવત છે 0 ઓછા 7 ની ઘાત 7 છે.

(સ્લાઇડ ટાઇ નો ઉલ્લેખ કરો: 22:48)

તે વત્તા એન બાદ એક થી બાદબાકી N નો 1 નીકળે છે. બીજી તરફ, જો તમે બીજી રીતે જાઓ તો આ બિંદુ પર રકમ 0 વત્તા 0 એ 0 છે, અને અહીંની રકમ 7 વત્તા 7 બરાબર 14 છે. આ કર્ણ સાથે સરવાળો 0, 1, 2, 3, 4 અને તેથી વધુ છે. આ એક છે 2 આ 3 છે અને તેથી આગળ.

(સ્લાઇડ ટાઇ નો ઉલ્લેખ કરો: 23:11)

તો આપણી પાસે આ ઉત્તર પશ્ચિમમાં દક્ષિણ પૂર્વીય કર્ણ છે, જે ઓછામાં ઓછા છે. ઓછા નિમ્ન 1 થી એન ઓછા 1. આ મને જો નંબર આપે છે. આ તફાવત છે જો તફાવત 6 કહે છે તો i જાણું છું કે તફાવત શું છે જો તફાવત 3 હશે. i જાણું છું કે ચોરસ કોણ છે અને મૂલ્યની શક્ય શ્રેણી ઓછા 7 થી વત્તા 7 ઓછા એન બાદ 1 થી વત્તા એન બાદ 1 અને બીજી દિશા માટે તે 0 થી 2 ગુણ્યા N ની 1 ઘાત 1 આપણા કિસ્સામાં બે ગુણ્યા એન બાદ 1 બે ગુણ્યા 7 છે જે 14 છે. તેથી, 0 થી 14, પરંતુ જો આપણી પાસે એન દ્વારા એન હોય તો આપણી પાસે બે છે. વખત એન બાદ 1. આ અમને હુમલો હેઠળ ચોરસ

ઓર્ડર ઓ રજૂઆત આપે છે. તેથી, જો આપણે જોવા ઈચ્છીએ છીએ કે આઈજ ચોરસ હુમલા હેઠળ છે તો અમે તપાસીએ છીએ કે તે પંક્તિ I છે એક કે સ્તંભ J 1 અથવા J ઓછા 1 છે, ત્રાંસી 1 છે અથવા હું વત્તા જે ત્રાંસા 1. જો આમાંથી કોઈ 1, તો તે આક્રમણ હેઠળ છે જો આ તમામ 0 હોય તો તે આક્રમણના અધિકાર હેઠળ નથી.

(સ્લાઈડ ટાઇ નો ઉલ્લેખ કરો: 24:07)

તેથી, ij એ ફી પૂરું પાડવામાં આવેલ પંક્તિ I કોલમ છે જે ઉત્તર પશ્ચિમથી દક્ષિણ પૂર્વ ત્રાંસા જે-માઈન -1 અને દક્ષિણ-પશ્ચિમથી પૂર્વ દિશામાં ત્રિકોણીય જે. વત્તા હું તમામ 0 ની બરાબર છું. જ્યારે આપણે આઈજ પર રાણી પ્રથમ આપણે બોર્ડનું પ્રતિનિધિત્વ કરીએ છીએ કે અમને જણાવવું કે ત્યાં છે, હવે i th પંક્તિ એ જથ્થા સ્તંભ પર સેટ કરેલી છે અને આ ચોરસને અનુરૂપ યોગ્ય પંક્તિ, કોલમ અને કર્ણ માટે આપણે તેમને બધાને હુમલો હેઠળ ગોઠવવાની જરૂર છે. તેથી, પંક્તિ હું હુમલો હેઠળ બની, કોલમ J હુમલો હેઠળ બની જાય છે અને ઘાતા કર્ણ પર એક ત્રિજ્યા J જોઈએ અને વધતી કર્ણ પર ત્રિજ્યા મીટર બધા બધા એક પર સેટ કરો; અને પૂર્વવત્ પણ એ જ રીતે, સરળ છે કે આપણે પહેલા બોર્ડ મૂલ્યને રીસેટ કરવું પડશે કે i th રાણી મૂકવામાં નથી. તેથી, આપણે ઓછા બાદ કહી શકીએ કે આ એક માન્ય મૂલ્ય નથી કારણ કે મૂલ્યો 0 થી N ની 1 ઘાત છે. તેથી, બાદ 1 સૂચવે છે કે i th રાણી આ ક્ષણે મુકવામાં આવ્યો નથી અને અમે આ પંક્તિ ફરીથી સેટ કરીએ છીએ અને આ સ્તંભ બરાબર છે 0 કારણ કે, આ પંક્તિ અને આ સ્તંભ ફક્ત આ રાણી દ્વારા જ હુમલો કરવામાં આવે છે. યાદ રાખો કે આપણે એક જ કર્ણ પર બે રાણીઓ કરી શકતા નથી કારણ કે, તેઓ એકબીજા પર હુમલો કરશે. તેથી, કોઈપણ બિંદુએ, આ પંક્તિઓમાંથી દરેક એક કોલમ અને કર્ણ એક જ રાણી દ્વારા હુમલો કરવામાં આવે છે અને તે રાણી દ્વારા આઈ કોમમા J દ્વારા હુમલો થવો જોઈએ. તેથી, હું ફક્ત અલ્પવિરામ J રાણી J આ બધા પર હુમલો કરી શકું છું, જો તે અન્ય રાણી દ્વારા હુમલો કરવામાં આવે તો અમે રાણીને આઈ કોમમા જેનમાં મૂકી શકતા નથી. હકીકત એ છે કે આ પહેલા તે સૂચવે છે કે આ બધા J વર્તમાન રાણી દ્વારા જ હુમલો કર્યો છે. તેથી, જ્યારે આપણે વર્તમાન રાણીને દૂર કરીએ છીએ ત્યારે આપણે તેમને ફરીથી 0 પર ફરીથી સેટ કરવું પડશે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 25:35)

પાયથોન માટે એક અમલીકરણ વિગત એ છે કે આ 5 અલગ અલગ ડેટા સ્ટ્રક્ચર્સ રાખવાની જગ્યાએ, અમારી પાસે એક બોર્ડ અને એક પંક્તિ છે અને સ્તંભ અને તે બધા જે અમે એક નેસ્ટેડ શબ્દકોશ તરીકે રાખીએ છીએ. તેથી, તે બોર્ડને બોલાવવા માટે અનુકૂળ છે અને અમે 5 ઉપ શબ્દકોશને સૂચવતી ટોચના 5 કી મૂલ્ય પર હશે રાણીની પદ આપણે કી રાણીને ફોન કરીશું. તેથી, બોર્ડ બોલાવવાના બદલે હું જ છું, અમે રાણી સાથે બોર્ડને સ્થિતિ તરીકે કી તરીકે કહીશું I પછી j છે તો આપણે કહીશું કે પંક્તિ I ની સંખ્યા 1 અથવા ઓછા 1 છે, આપણે કહીશું કે કીની હરોળમાં બોર્ડ સમાન બોર્ડ છે ઉત્તર પશ્ચિમથી દક્ષિણ પૂર્વમાં કી સ્તંભ બોર્ડ અને દક્ષિણ પશ્ચિમથી ઉત્તર પૂર્વમાં બોર્ડ. તેથી, અમે હમણાં જ તેને રૂપાંતરિત કર્યા છે. તેથી, આપણે દરેક વિધેયમાં લગભગ 5 જુદા જુદા ભાગો પસાર કરવાની જરૂર નથી. આપણે માત્ર એક જ બોર્ડ પાસ કરવી પડે છે જે એક શબ્દકોશ છે જેનો રસ છે. યાદ રાખો કે આ રીતે અમે અમારા ઉકેલ આપવાનો પ્રયાસ કરીએ છીએ. તેથી, અમે રાણીમાં હરોળમાં મૂકવા માગીએ છીએ અને દરેક સ્તંભ માટે ઉપલબ્ધ છે, અમે બોર્ડને અપડેટ કરવાનો પ્રયાસ કરીશું અને આ રીતે. હવે, આ બાબતો યોગ્ય રીતે કરવા માટે હવે આપણી પાસે વધુ સારી રીતો છે. તેથી, આપણે બતાવ્યું છે કે આ શબ્દકોશનો

ઉપયોગ કરીને અથવા આ 5 અલગ અલગ રજૂઆતો આપણે એક પંક્તિ અને કોલમ ઉપલબ્ધ છે કે નહીં તે ચકાસી શકીએ છીએ, જ્યારે આપણે રાણી મૂકીએ છીએ ત્યારે બોર્ડને કેવી રીતે અપડેટ કરવું અને અમે રાણીને પૂર્વવત્ કરીએ છીએ અહીં આપણે જે ચર્ચા કરી તે વિશે એક વાસ્તવિક પાયથોન અમલીકરણ છે. તેથી, આપણી પાસે અહીં આ કાર્ય છે જેને સ્થાન રાણી કહેવામાં આવે છે. સ્થાન રાણી અમે જણાવ્યું હતું કે પંક્તિ હું અને બોર્ડ લે છે અને તે કરે છે પ્રથમ વસ્તુ તે નક્કી કરવા માટે છે. N ની મૂલ્ય શું છે? તેથી, અમે ફક્ત લેવા યાદ રાખો કે બોર્ડ હવે એક શબ્દકોશ છે તેથી, રાણીના બોર્ડ અમને જણાવશે કે વસ્તુમાં કેટલા પંક્તિઓ છે. જો આપણે રાણીના બોર્ડની કીની લંબાઈ લઈએ છીએ તો આપણે આ તે પસાર કર્યા વગર N પુનઃ પ્રાપ્ત કરવાની માત્ર રીત છે હવે, આપણે દરેક શક્ય મૂલ્ય માટે 0 થી N ની 1 ઘાત 1 માટે છે, જે તમામ સ્તંભ કિંમતો માટે અમે તપાસીએ છીએ કે જો i, j વર્તમાન બોર્ડમાં મફત છે. જો તે મફત છે તો આપણે રાણી ઉમેરીએ છીએ તે આ બરાબર કોડ છે જે સ્યુડો કોડ હતું, જો હું એન બાદ 1 છે તો આપણે સાચી દિશામાં આગળ વધીશું તો હું રાણીને 1 થી વધુ પંક્તિ પર મૂકીને ઉકેલ વધારવાનો પ્રયાસ કરીશ. જો સોલ્યુશન વિસ્તરે તો અમે સાચું પામીએ છીએ તો આપણે રાણીને રદબાતલ કરીએ છીએ. તેથી, રાણીને રદ કરીને આ રાણીને દૂર કરવામાં આવશે અને તે બોર્ડને અપડેટ કરશે અને છેવટે, જો આ લૂપ દરેક શક્ય સ્તંભ માટે તમામ રીતે જાય અને તે સાચું ન આવે તો અમે તેનો અર્થ એ કે અમે રાણીને i th પંક્તિ પર મૂકી શકતા નથી. તેથી, આપણે ખોટા ફેરવીએ છીએ. હવે મુખ્ય કાર્ય છે કે આપણી પાસે મુખ્ય કોડ ખાલી શબ્દકોષ થવા માટે બોર્ડને પ્રારંભ કરીને શરૂ થશે, તે વપરાશકર્તાને પૂછશે કે કેટલી રાણીઓ અમે એન દ્વારા N દ્વારા કયા પ્રકારની બોર્ડ પર છીએ. તેથી, યાદ રાખો કે અમે ઇનપુટ લઈશું તે હશે એક શબ્દમાળા અમે તેને ઇન વાપરીને રૂપાંતરિત કરીએ છીએ અને આપણે તેને એન તરીકે રેકોર્ડ કરીએ છીએ. તેથી, તે આપણને પૂછે છે કે તે સંખ્યાને પૂર્ણાંકમાં રૂપાંતરિત કરે છે અને તેને N તરીકે પસાર કરે છે. પછી આપણે N નો નંબર સાથે બોર્ડને ઇનીશિયલાઈઝ કરીશું. અમને N ની જરૂર છે, કારણ કે, આપણને તે કેવી રીતે સેટ કરવું તે યાદ રાખો કે સૂચકાંકો 0 થી N ઓછા 1 અથવા એન બાદબાકી N થી ચાલે છે વત્તા N . N એ શબ્દકોશ પ્રારંભ કરવા માટે જરૂરી છે અને છેલ્લે, અમે રાણી મૂકવાનો પ્રયાસ કરીએ છીએ. તેથી, આરંભથી ખાલી બોર્ડ સેટ કરવામાં આવશે જ્યાં કશું હુમલો હેઠળ નથી, તો અમે આ બોર્ડ પર 0 મી પંક્તિમાં રાણી મૂકવાનો પ્રયાસ કરીએ છીએ, જો તે સફળ થાય, તો અમારી પાસે એક કાર્ય છે જે બોર્ડને છાપે છે. ચાલો જોઈએ કે આ અન્ય કાર્યો કેવી રીતે કામ કરે છે. ચાલો પ્રથમ કાર્ય કરીએ જે બોર્ડને પ્રારંભ કરે છે. બોર્ડની શરૂઆતમાં જણાવાયું છે કે દરેક પેટા ક્વિન પંક્તિ કોલમ ઉત્તર પશ્ચિમ દક્ષિણ પૂર્વીય દક્ષિણ પશ્ચિમ અથવા ઉત્તર પૂર્વમાં દરેક માટે દરેક કી માટે સૌ પ્રથમ તો આપણે તે કી સાથે એક શબ્દકોશ બનાવીએ છીએ. તેથી, આ કહે છે એક ખાલી શબ્દકોશ બનાવો; હવે રાણી, હાર અને સ્તંભ માટે આ ત્રણ બાબતો માટે, સૂચકાંકો 0 થી N ની 1 ઘાત છે. શ્રેણીમાં હું માત્ર કી મૂલ્ય નક્કી કરું છું, જે હું રાણીના કિસ્સામાં 1 બાદ કરું છું તે કહે છે કે રાણીની રાણી મને મૂકી દેવામાં આવ્યું નથી અને પંક્તિ અને સ્તંભ માટે આ હુમલાખોર છે, જે કહે છે કે તેઓ 0 છે જો તેઓ હુમલા હેઠળ નથી અને એક જો તેઓ હુમલો હેઠળ છે. પ્રારંભિક વસ્તુ 0 કહે છે. એ જ પ્રમાણે ઉત્તર-પશ્ચિમથી દક્ષિણ પૂર્વમાં રેન્જ નોન એન બાદ ઓછા એન બાદ 1 થી વત્તા એન બાદ -1 નીકળે છે. અમે આમાંની દરેક કી 0 થી 2 નો 2 થી n ની 1 ની જેમ જ સુયોજિત કરીએ છીએ. આપણે દક્ષિણ પશ્ચિમ દિશામાં ઉત્તર પૂર્વીય કણઓ 0 કરવા ઈચ્છીએ છીએ. આ એક કારણ એ છે કે આપણે શા માટે એક શબ્દનો ઉપયોગ કરી રહ્યા છીએ કારણ કે અન્ય વસ્તુઓ માટે અલબત્ત, આપણે સૂચિનો ઉપયોગ કરી શકીએ છીએ 0 થી N ઓછા એક કુદરતી સૂચિ અનુક્રમણિકા છે, પરંતુ અહીં આપણી પાસે વિચિત્ર સૂચકાંકો છે જે ઓછા એન બાદ એક થી વત્તા એન બાદ 1 અને તેથી પર. એટલે જ આપણે બે સ્તરે પુનરાવર્તિત શબ્દકોશનો ઉપયોગ કરીએ છીએ. આ

બોર્ડને પ્રારંભ કરે છે; આપણે કઈ પંક્તિઓને દરેક પંક્તિ માટે સારી રીતે છાપીએ છીએ તે પંક્તિઓ આપણે ગોઠવીએ છીએ. તેથી, આપણે બોર્ડને ડોટ રાણી ડોટ કીઝ આપીએ છીએ 0 નોટ 1 ઓછા સુધી આપણે તેને સોઈ કરીએ છીએ અને દરેક પ્રકારની પંક્તિ માટે આપણે તે પંક્તિ માટે પંક્તિ અને સ્તંભ નંબર છાપીશું. જ્યારે અમારી પાસે સફળ ઉકેલ હોય ત્યારે આવું થાય છે. જ્યારે સ્થિતિ મફત છે ત્યારે આપણે તપાસીએ છીએ કે પંક્તિની પંક્તિ બોર્ડ છે 0 સ્તંભ એન્ટ્રી 0 છે, ત્રાંસી પ્રવેશ જ ઓછા 1 0 છે અને કર્ણ એન્ટ્રી j વત્તા હું 0 બરાબર તે જ છે જેમ આપણે પહેલાં અને છેલ્લે જણાવ્યું હતું કે, ત્યારે શું થાય છે અમે રાણી ઉમેરવા જ્યારે અમે તેને મૂકવા માટે છે રાણી અધિકાર ઉમેરો. તેથી, અમે પંક્તિ i અને j માટે રાણીની એન્ટ્રી ગોઠવીએ છીએ અને તે પછી આપણે અનુરૂપ પંક્તિ કોલમને ચિહ્નિત કરીએ છીએ અને કર્ણ એક હોવું જોઈએ અને જ્યારે આપણે રાણીને રદ કરીએ છીએ ત્યારે આપણે રાણી પ્રવેશને બાદબાકી 1 અને પંક્તિ કોલમ અને વિકર્ણ એન્ટ્રીઓ 0 ; આ બધી જ છે જે આપણે સ્યુડો કોડમાં લખ્યું હતું જે પાયથોન કોડમાં ઔપચારિક છે. હવે અમે આ કોડ ચલાવીએ છીએ અને ચકાસી શકીએ છીએ કે તે કાર્ય કરે છે. તેથી, અહીં આપણી પાસે આ કોડ 8 ક્વિન્સ ડોટ પી છે જે કોડ છે જે આપણે એડિટરમાં જોયું છે. હવે જો હું આ કોડને પાયથોન 3.5 8 ક્વિન્સ ડોટ પીઈ તરીકે ચલાવીશ. આ રીતે જો તમારી પાસે પાયથોન પ્રોગ્રામ છે તો તમે તેને સીધું જ ચલાવી શકો છો અને પછી તે આયાત કરી શકો છો જો તમે આમ કરો તો તે અમને પૂછશે કે અમે કેટલી ક્લીન્સ જોઈએ છે. દાખલા તરીકે, જો આપણે તેને 4 નંબર આપીએ તો આપણે તે ઉકેલ મેળવીશું જે આપણે પહેલાનાં ઉદાહરણમાં જોયું છે તે ખૂબ સરસ રીતે છપાયેલ નથી. તેથી, જો આપણે નંબર 8 આપીએ તો આપણે આનો એક ઉકેલ મેળવી શકીએ છીએ કે તે દર્શાવે છે કે તમે ખરેખર તે પ્રિન્ટ બોર્ડ ફંક્શનને બદલી શકો છો, હું તમને કોડ બતાવતો નથી, પરંતુ તેને વધુ વપરાશકર્તા મૈત્રીપૂર્ણ રીતે છાપવા માટે. તેથી, મારી પાસે બીજું એક કાર્ય છે જેને સાડે કહેવામાં આવે છે જો હું આવું કરું તો તે મને વધુ વાંચનીય સ્વરૂપમાં 4 રાણીનો ઉકેલ બતાવે છે. તેથી, તમે ક્યારેક કર્ણની સ્થિતિ જુઓ છો અને જો હું 8 રાણીઓ માટે કરું તો પછી જુઓ કે ત્યાં વધારાની કોલમ છે. તેમાં કેટલીક ભૂલ છે, પરંતુ એક વધારાનું કોલમ છે, પરંતુ મૂળભૂત રીતે તમે જોઈ શકો છો કે જો તમે છેલ્લા સ્તંભને અવગણી શકો છો જે પ્રથમ 8 રાણી ઉકેલમાં રાણીઓની સ્થિતિ દર્શાવે છે. તેથી, એકવાર આપણી પાસે પ્રતિનિધિત્વ કર્યું છે અને કોડનું માળખું કામ કર્યું છે તે પછી તે એકદમ સીધા આગળ છે, તેને વાસ્તવિક પિથન કોડમાં પરિવર્તિત કરવું ખૂબ સરળ છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 32:54)

અંતિમ પગલું તરીકે આપણે ધારવું નથી કે એક ઉકેલ છે, પરંતુ બધા ઉકેલો યોગ્ય છે. તેથી, આપણે કોઈ ક્ષણમાં અગાઉની વસ્તુ ઇચ્છતા નથી કે તે ઉકેલ શોધે પછી તે સાચું પાછું આપે છે અને તે પછી પાછલું સ્તર સાચું વળે છે અને છેવટે તે બોર્ડને છાપે છે. ધારો કે અમે પ્રથમ ઉકેલ પર રોકવા નથી માંગતા, પરંતુ છાપવાનું ચાલુ રાખવું તે વાસ્તવમાં ખૂબ સરળ છે; તો આપણે જે કરીએ છીએ તે બધા શક્ય સ્થિતિમાંથી પસાર થવું જ છે અને જ્યારે પણ અંતિમ તબક્કા સુધી પહોંચીએ છીએ ત્યારે ખરેખર ઉકેલ અંતિમ તબક્કે પહોંચે છે તો આપણે તેને અમારા કેસમાં રેકોર્ડ કરીશું તો તે કદાચ તેને છાપી દેશે નહીં તો અમે તેને આગળ વધારીશું અને આગળ વધીએ છીએ. . વાસ્તવમાં તે એક સોલ્યુશન છાપો કરતાં બધા સોલ્યુશન્સ છાપવા માટે ખૂબ સરળ છે કારણ કે અમારે યાદ રાખવું નથી કે અમારું ઉકેલ વિસ્તરે છે કે નહીં તે ખરેખર દરેક શક્ય ઉકેલથી ચાલી રહ્યું છે. આ જ વસ્તુ એ છે કે તે દરેક ઉકેલથી ખૂબ જ અંત સુધી ચાલશે નહીં અને પછી તે કામ કરશે નહીં તે નક્કી કરશે. એવું નથી કે અમે બધી શક્ય રાણી હોદ્દાની મૂકે છે અને પછી તેને અજમાવીએ છીએ તો અમે તેને નાની વસ્તુઓ માટે અજમાવી રહ્યા છીએ, કારણ

કે એકવાર આપણે પદ 5 પર અટવાઈ જઈએ તો પછી તે લંબાવવાનો પ્રયાસ નહીં કરે તે પાછો આવશે અને આવો, પરંતુ આ ફક્ત ખૂબ સરળ લૂપ છે જે ફક્ત બધા ઉકેલો છાપે છે. તેથી, અહીં કોડ છે તે બરાબર એ જ કોડ છે અન્યથા માત્ર વસ્તુ એ છે કે રાણી કાર્ય ખૂબ જ સરળ છે, આપણે ફક્ત દરેક જ માટે પ્રયત્ન કરીએ છીએ અને એક થી 0 થી એન નીચું 1 ની જરૂર છે, જો તે મફત છે તો આપણે રાણી ઉમેરીએ છીએ, જો આપણે છેલ્લી પંક્તિ સુધી પહોંચ્યા છીએ તો અમે બોર્ડને છાપીએ છીએ, અમે ઉકેલને વિસ્તારિત કરીએ છીએ અને પછી અમે રાણીને પૂર્વવત્ કરીએ છીએ અને આગામી એકનો પ્રયાસ કરીએ છીએ. દરેક જ માટે આપણે પહેલા રાણી ઉમેરવા જઈ રહ્યા છીએ, જો તે સ્થાનાંતરિત થાય તો તે ઉકેલને વિસ્તારિત કરે છે અને છેવટે, આપણે તેને પૂર્વવત્ કરવા અને આગામી જ; અમે હમણાં જ અજાણપણે દરેક સંભવિત જ પ્રયાસ કરવાનો પ્રયાસ કરી રહ્યા છીએ અને અમે ક્યારેય ફરિયાદ કરી રહ્યા છીએ કે અમે સફળ થયા નથી, બાકીના ખૂબ સમાન છે, પ્રિન્ટ બોર્ડમાં પ્રિન્ટ બોર્ડ થોડો અને થોડો ફેરફાર બદલાઈ ગયો છે કે આપણે તેને બદલ્યું છે, તે એક જ પંક્તિ પર આખી વસ્તુ છાપશે તેથી, આપણે આ વસ્તુને ઉમેરે છે જે કહે છે કે જગ્યા બરાબર છે. તેથી, પંક્તિ દ્વારા પંક્તિને બદલે એક હરોળમાં અમે પોઝિશન્સને છાપીએ છીએ, જેથી આપણે તે બધાને જોઈ શકીએ. હવે જો આપણે આ કાર્યને હવે જોયું અને અમે તેને 4 રાણીઓ માટે છાપીએ તે છાપવાનો પ્રયત્ન કરીએ, તો તે બે ઉકેલો છાપે છે, આ એક જ વસ્તુના બે ફેરવાયેલા ઉકેલો છે. જો આપણે તે 8 રાણીઓ માટે કરીએ તો તે વાસ્તવમાં 92 ઉકેલો છે, પરંતુ ખરેખર આ 92 સોલ્યુશન્સ જો તમે પરિભ્રમણ અને પ્રતિબિંબે જોશો તો તે ઘણું ઓછું થાય છે, પરંતુ જો તમે ફક્ત ચોરસની સ્થિતિને જુઓ કે જે તમને આપવામાં આવે છે, તે પછી 92 વિવિધ ઉકેલો છે જે તે છાપે છે. આ 8 રાણીઓની સમસ્યાની સામે બેક્ટ્રેકિંગની અમારી ચર્ચાને સમાપ્ત કરે છે.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રો. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 06 લેક્ચર - 02 ગ્લોબલ સ્કોપ, નેસ્ટેડ ફંક્શન્સ

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 00:02)

અમે 8 રાણીઓની સમસ્યા, અને અમારા ઉકેલનો સમાવેશ બોર્ડનું પ્રતિનિધિત્વ કરે છે, જે ચોરસમાં હુમલો હેઠળ છે અને રાણીઓને એક પછી એક મૂકીને. આ સોલ્યુશનની એક વિશેષતા એ છે કે અમને કાર્યો દ્વારા બોર્ડને પાસ કરવા માટે તેમને સુધારવાનું અથવા તેનું કદ બદલવાનું કરવાનું હતું અને મારે તેમને પ્રારંભ કરવાનું હતું અને તેથી જ કારણ કે બોર્ડને દરેક કાર્ય દ્વારા સુધારાશે રાખવો પડ્યો હતો. હવે પ્રશ્ન એ છે કે બધા જ સ્થળે બોર્ડ પસાર થઈ શકે છે?

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 00:31)

તો, શું આપણે આ બોર્ડને સ્પષ્ટ રીતે પસાર કરવાનું ટાળી શકીએ અથવા અમારી પાસે એક જ વૈશ્લિક કોપિ હોઈ શકે કે જે તમામ કાર્યોને અપડેટ કરી શકે છે જે અમને આ બોર્ડને આગળ અને પાછળથી સાચવશે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 00:42)

તેથી, આ અમને સ્કોપની એક ખ્યાલ લાવે છે. પાયથોન માં નામની તક એ કોડનો ભાગ છે કે જ્યાં તે એક અપડેટ વાંચવા માટે ઉપલબ્ધ છે. હવે પિથન અવકાશમાં મૂળભૂત રીતે ફંક્શનમાં સ્થાનિક છે, આપણે જોયું કે જો આપણે કોઈ કાર્યમાં કોઈ નામનો ઉપયોગ કરીએ છીએ અને તે કાર્યની બહાર જ નામનો ઉપયોગ કરતા અલગ છે. પરંતુ વાસ્તવમાં આ જ થાય છે જ્યારે આપણે કાર્યની અંદર નામ અપડેટ કરીએ છીએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 01:08)

ચાલો આ ચોક્કસ કોડને જોઈએ. અહીં આપણી પાસે એક ફંક્શન છે જે x મૂલ્ય વાંચે છે અને તેને y નામમાં સંગ્રહ કરીને પ્રિન્ટ કરે છે, હવે પ્રશ્ન છે: આ x શું છે? વેલ અહીં એક એક્સ છે તો, શું આ કાર્યની અંદરની x બરાબર કાર્યની બહાર x દર્શાવે છે કે નહીં.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 01:33)

તો અહીં આપણે તે ફંક્શન જોયું છે, આપણે એક 1 ડોટ પીઆઇ ફાઇલ લખી છે જેમાં તે કોડ છે. તેથી, આપણી પાસે કાર્ય f છે જે બહારથી x વાંચે છે અને તેને છાપવાનો પ્રયાસ કરે છે. જો તમે આ રન કરો છો, તો ખરેખર તે મૂલ્ય 7 ને છાપે છે જે આપણે અપેક્ષા રાખીએ છીએ, તેથી y મૂલ્ય 7 મેળવે છે કારણ કે x પાસે મૂલ્ય 7 ની બહાર છે અને તે x ને અંદરથી એક ફંક્શનમાં વારસામાં મળેલ છે.

(સ્લાઇડ સમય જણાવો: 02:03)

તો આ કામ કરે છે હવે જો તમે આ કરો તો, અને આ બરાબર જ કાર્ય છે સિવાય કે y ની મૂલ્ય પ્રિન્ટ કર્યા પછી x બરાબર 22 ની અંદર એક સુયોજિત કરે છે. હવે શું થાય છે?

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 02:19)

તેથી અહીં સ્ક્રીનના મધ્ય ભાગમાં f2 ડોટ પી.આઇ.વાય છે, તેથી એક 1 ડોટ પી સાથેના સંબંધમાં માત્ર એટલો જ તફાવત છે, x ની બરાબર 22 બરાબર. હવે જો તમે એક 2 ડોટ પીઈ ચલાવવાનો પ્રયાસ કરો છો, તો તે આપણને એક ભૂલ કહે છે કે મૂળ એસેન્શન y બરાબર x આપણને અનબાઉન્ડ સ્થાનિક નામ આપે છે ત્યાં કોઈ એક્સ નથી જે આ બિંદુએ એક ની અંદર ઉપલબ્ધ છે. તેથી, કોઈક રીતે x ની બરાબર 22 ની અંદર એક ની સ્થિતિને બદલીને x ની સ્થિતિ બદલાય છે, તે બહારની x નજર કરવા તૈયાર નથી, તે આગ્રહ કરશે કે અંદરની x છે આ ભૂલ તરીકે આપે છે

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 02:59)

તેથી જો x એ f માં મળ્યું નથી, તો પાયથોન વૈશ્વિક x માટે બંધાયેલ કાર્યને જોવા માટે તૈયાર છે જો કે, જો x f માં સુધારવામાં આવે તો તે સ્થાનિક નામ બની જાય છે અને પછી તે ભૂલ આપે છે

(સ્લાઇડ સમયનો સંદર્ભ લો: 03:15)

તેથી આ સખત રીતે કહીએ તો માત્ર અનિવાર્ય મૂલ્યો પર જ લાગુ પડે છે. જો આપણે આ ફંક્શનને બદલી નાખીશું તો આપણે x પૂર્ણાંક નથી, પરંતુ ઉદાહરણ તરીકે એક સૂચિ અને અમે y ને સૂચિમાં 0 મી તત્વ પસંદ કરવા માટે કહેવામાં આવ્યા છીએ અને ત્યારબાદ આપણે 0 થી 0 ના તત્વને 22 થી બદલીએ. સમય: 03:38)

અહીં આપણી પાસે આ કાર્ય છે જેમાં આપણે હવે એક્સને પૂર્ણાંકથી સૂચિમાં બદલ્યા છીએ અને ત્યારબાદ આપણે તેને y માં મૂકવાનો પ્રયાસ કરીએ છીએ. પરંતુ આપણે તે સૂચિને ફંક્શનમાં અપડેટ કરીએ છીએ અને પછી જો તમે તેને ચલાવો છો તો તે 7 ની કિંમતની પ્રિન્ટ કરે છે જે આપણે અપેક્ષા રાખીએ છીએ.

(સ્લાઇડ સમય જણાવો: 03:55)

તો આ કામ કરે છે જો અમારી પાસે અવિશ્વસનીય મૂલ્ય છે, તો મારે અર્થવ્યવહારુ મૂલ્ય દિલગીર છે, તો પછી આપણે વાસ્તવમાં તેને એક અંદર બદલી શકીએ છીએ અને કંઈ થશે નહીં. તેથી, વૈશ્વિક નામો જે પરિવર્તનક્ષમ કિંમતો તરફ નિર્દેશ કરે છે તે ફંક્શનમાં અપડેટ થઈ શકે છે. હકીકતમાં, આનો અર્થ એ થાય છે કે જે સમસ્યા છે જે અમે શરૂ કરીએ છીએ. કેવી રીતે તેની અંદરની 8 ક્વીન્સ સાથે બોર્ડ પસાર થવું ટાળવા માટે ખરેખર વધુ સમજૂતીની જરૂર નથી. બોર્ડ એક શબ્દકોશ છે, તે પરિવર્તનક્ષમ મૂલ્ય છે અને હકીકતમાં આપણે 8 રાણીઓ એવી રીતે લખી શકીએ છીએ કે અમે ફક્ત બોર્ડને પસાર કરવાની અવગણના કરીએ છીએ, અમે બધી વ્યાખ્યાઓ બદલીએ છીએ જેથી બોર્ડ ન થાય અને દંડ કામ કરે છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 04:33)

અહીં આપણે પાછલા કોડને ફક્ત તમામ વિધેયોમાંથી બોર્ડને દૂર કરવા માટે ફરીથી લખ્યા છે. તેથી અગાઉ પ્રારંભ કર્યો બોર્ડ શરૂ કરો અને હવે તે ફક્ત પ્રિન્ટ બોર્ડને દલીલ નહીં લેતા અને તે બધા જ આ વૈશ્વિક મૂલ્ય બોર્ડનો ઉલ્લેખ કરે છે જે તમે દરેક જગ્યાએ જોઈ શકો છો. તેથી, આપણી પાસે આ વૈશ્વિક મૂલ્ય બોર્ડ છે જે ફંક્શનની અંદર ઓળખવામાં આવે છે અને તે કોઈ બાબત નથી કે પસાર થઈ રહ્યું નથી

કારણ કે આ પરિવર્તનક્ષમ મૂલ્ય છે, તેથી તે મૂલ્યની તપાસ કરવા જઈ રહ્યું છે જેનું નામ બહાર આ ખાલી છે શબ્દકોશ. અને પછી સ્થાન રાણી જેવી તમામ વિધેયો અથવા ક્વિઝને પૂર્વવત્ કરોⁿ અથવા રાણી ઉમેરો માત્ર તેમના સંબંધિત પરિમાણો લે છે અને સર્વવ્યાપક રીતે વૈશ્વિક મૂલ્યનો સંદર્ભ લો. તેથી, જો તમે હવે આ વૈશ્વિક વર્ઝન રન કરો છો, તો આપણે બરાબર એ જ આઉટપુટ મેળવીએ છીએ. તેથી, જેમ આપણે આપણા ઉદ્દેશ્ય માટે જણાવ્યું હતું કે તે 8 ક્વિન્સની સમસ્યાનો ઉકેલ લાવવાનો છે, તે બોર્ડને પસાર કર્યા વિના, હકીકત એ છે કે પાયથોન એક વિધેયની અંદર અદલાબદલી વૈશ્વિક નામોને અમલમાં મૂકે છે, જે આપણને જરૂર છે.

(સ્લાઇડ સમય જણાવો: 05:36)

પરંતુ, જો આપણે વાસ્તવમાં વૈશ્વિક પૂર્ણાંક માગીએ તો શું? શા માટે તમે એક વૈશ્વિક પૂર્ણાંક માંગો છો? ઠીક છે, દાખલા તરીકે, આપણે ફંક્શનને કેટલી વખત ગણવામાં આવે છે તેની ગણતરી કરવી છે. તેથી દરેક વખતે કાર્ય કહેવાય છે, આપણે આ ફંક્શનની અંદર પૂર્ણાંકને અપડેટ કરવાનું પસંદ કરીશું. પરંતુ પૂર્ણાંક એ ફંક્શનમાં સ્થાનિક નામ ન હોઈ શકે, કારણ કે કાર્યમાં સ્થાનિક મૂલ્યનો નાશ થશે, અમે તેને ચાલુ રાખવું જોઈએ, તેથી તે મૂલ્ય હોવું જોઈએ જે ફંક્શનની બહાર અસ્તિત્વમાં છે. પરંતુ પૂર્ણાંક હોવું તે એક અસંબદ્ધ મૂલ્ય છે અને તેથી અમે તેને કાર્યની અંદર અપડેટ કરવાનો પ્રયાસ કરીશું તો તે તેને સ્થાનિક મૂલ્ય તરીકે ગણશે. તો, આપણે આ કેવી રીતે મેળવી શકીએ? પાયથોન પાસે ઝોબલ નામની ઘોષણા છે, જે કહે છે કે વૈશ્વિક સ્તરે કોઈ નામનો ઉલ્લેખ કરવામાં આવે છે અને સ્થાનિક મૂલ્ય તરીકે લેવાય નહીં. જો આપણે એક ની અમારી અગાઉની વ્યાખ્યા બદલીએ, જેથી અમે આ ચોક્કસ ટેગ વૈશ્વિક x ઉમેરીએ તો તે કહે છે કે આ x અને આ x બન્ને એક જ x નો સંદર્ભ લે છે. આ પાયથોન ને કહેવાનો માર્ગ છે, નવું x નામના સ્થાનિક નામ x ની રચના સાથે આ x બરાબર 22 ને મૂંઝવતા નથી. એક માં ઉલ્લેખ બધા એક્સ ખરેખર બહાર x તરીકે જ છે અને વૈશ્વિક કિંમતો તરીકે ગણવામાં આવે છે. તેથી, આ એ એક એવું રીત છે કે જેમાં આપણે પાયથોન પ્રોગ્રામમાં વૈશ્વિક ધોરણે સુલભ અસમાન મૂલ્ય બનાવી શકીએ છીએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 06:51)

તો અહીં વૈશ્વિક કોડ છે. અમારી પાસે વૈશ્વિક x છે, અને ફક્ત ખાતરી કરો કે x બરાબર 22 ની અંદર છે તે વાસ્તવમાં તે x ને બહાર છે. અમારી પાસે પ્રિન્ટ એક્સ છે, જે હવે કોલ માટે એક છે. તેથી મુખ્ય પ્રોગ્રામની નીચે આપણી પાસે પ્રિન્ટ x છે, હવે x એ 7 પહેલા એક કહેવાતા હતા પરંતુ x એ 22 ની અંદર એક માં સુયોજિત થઈ હતી. તેથી, આપણે બીજા પ્રિન્ટ સ્ટેટમેન્ટ સિવાય બીજાને 22 આપીશું. આ નિવેદન પ્રથમ છાપી દેવું જોઈએ. તે છાપવા માટે 7 માંથી છાપો અને પછી આ પ્રિન્ટ x માંથી 22 છાપો. તેથી, જો તમે આ ચલાવો છો તો આપણે આ જોઈ શકીએ છીએ કે અમારી પાસે બે લીટીઓ છે, પ્રથમ 7 પ્રિન્ટ વાચથી આવે છે અને બીજા સ્તર 22 પ્રિન્ટ એક્સની બહાર આવે છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 07:33)

જ્યારે આપણે સ્થાનિક અવકાશના વિષય પર છીએ, ત્યારે પાયથોન આપણને વિધેયોમાં વિધેયોને વ્યાખ્યાયિત કરવા દે છે. તેથી, દાખલા તરીકે, અહીં ફંક્શન એક એક ફંક્શન જ અને એચ વ્યાખ્યાયિત કર્યું છે, વળતરની જી વત્તા 1, b એ b નો 2 ગુણ્યા b આપે છે. હવે આપણે y ને x વત્તા H નો x ને બદલે x ને સુયોજિત કરવાને બદલે y ને અપડેટ કરી શકીએ છીએ. હવે, આમાં નોંધવું છે કે આ કાર્યો જ અને એચ ફક્ત એક માટે જ દેખાય છે. તેઓ એક ની આ અવકાશમાં વ્યાખ્યાયિત કરવામાં આવે છે. તેથી, તેઓ એક ની અંદર છે, અને તેથી તેઓ બહાર દૃશ્યમાન નથી. તો, જો હું આ સમયે x ના ૬ ને પૂછું તો હું બહાર જઈશ તો આ એક ભૂલ હશે, કારણ કે તે કહેશે કે આવી કોઈ વ્યાખ્યાયિત વ્યાખ્યા નથી. આ ઉપયોગી છે કારણ કે હવે તમે સ્થાનિક કાર્યોને વ્યાખ્યાયિત કરી શકો છો કે જે અમે એક વિશિષ્ટ કાર્ય

કરવા માગીએ છીએ જે એક સાથે સુસંગત છે, પરંતુ તે કોઈ કાર્ય ન હોવું જોઈએ જે બીજા બધાને ખુલ્લા છે અને આ શક્ય છે. અલબત્ત, એ જ નિયમો લાગુ પડે છે જો આપણે જી અથવા એચ ની અંદર x જોશું તો તેથી, જો આપણે એક x અહીં જોશું તો તે પ્રથમ એક ની શોધ કરવાનો પ્રયત્ન કરશે, જો તે ત્યાં ન હોય તો તે બહાર જઈ શકે છે અને તેથી આગળ. તેથી, ક્યાં તો આપણે તે વૈશ્વિક જાહેર કરીશું કે જેમાં આપણે જી અથવા એચ ની અંદર તેને અપડેટ કરી શકીએ અથવા તે પહેલાં જ નિયમનો ઉપયોગ કરશે જો આપણે અમલપાત્ર મૂલ્યને અપડેટ કરતા નથી તો તે બહાર દેખાશે અને જો તે પરિવર્તનક્ષમ મૂલ્ય હશે તો તે તેને અંદરથી અપડેટ કરવા માટે. હવે કેટલાક વધુ રિફાઇનમેન્ટ્સ છે. પાયથોનમાં બિન સ્થાનિક તરીકે ઓળખાતું મધ્યવર્તી સ્કોપ છે જે જી અને એચ ની અંદર જણાવે છે તે એક ની અંદર મૂલ્યનો સંદર્ભ આપે છે, પરંતુ એક ની બહારના મૂલ્યને નહીં. આ એક તકનીકી છે અને તે ખૂબ જ સુસંગત નથી જો અમને તેની આવશ્યકતા હોય તો અમે તે પાછો આવશે, પરંતુ ક્ષણ માટે જો તમે બિન સ્થાનિક જાહેરાતો વિશે વધુ જાણવા માંગતા હો, તો કૃપા કરીને પાયથોન દસ્તાવેજીકરણ જુઓ. પરંતુ વૈશ્વિક એ મહત્વનું છે, વૈશ્વિક અમને કાર્યસ્થળમાં અવિશ્વસનીય મૂલ્યને સ્થાનાંતરિત કરવા અને એક કાર્યમાં સુધારાત્મક બનાવવા માટે પરવાનગી આપે છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 09:33)

સારાંશમાં, આપણે જે જોયું છે તે એ છે કે પાયથોન નામો કાર્યોની અંદરથી જોવામાં આવે છે, જો આપણે અવિચારી મૂલ્યને અપડેટ કરીએ છીએ તો તે સ્થાનિક કોપિ બનાવે છે તેથી અમારે અપડેટ કરવા માટે વૈશ્વિક વ્યાખ્યા હોવી જરૂરી છે નિર્વિકાર મૂલ્યો પ્રશ્નની બીજી બાજુએ, જો તમારી પાસે યાદીઓ અને શબ્દકોશો જેવા અમૂર્ત મૂલ્યો છે તો કોઈ સમસ્યા નથી. કોઈ વિધેયની અંદર અમે વૈશ્વિક રીતે તેનો સંદર્ભ લઈ શકીએ અને તેને અપડેટ કરી શકીએ છીએ. અને આ આપણે આપણા 8 ક્વીન સોલ્યુશનમાં જોયું છે, અમે બોર્ડને વૈશ્વિક મૂલ્યમાં બનાવી શકીએ છીએ અને માત્ર તેને દલીલ તરીકે સ્પષ્ટપણે પસાર કરતા તેને દરેક વિધેયની અંદર અપડેટ કરીએ છીએ. છેવટે, આપણે જે જોયું છે તે છે કે આપણે નેસ્ટ કાર્યો કરી શકીએ છીએ. અમે વિધેયોમાં કહેવાતા હેલ્પર વિધેયો બનાવી શકીએ છીએ જે બહારથી છુપાયેલ હોય છે જેનો ઉપયોગ ફંક્શનમાં ઉપયોગમાં લઈ શકાય છે જેથી તેની પ્રવૃત્તિઓને નાના એકમોમાં વિભાજિત કરી શકાય.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રોફેસર. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 06 લેકચર - 03 જનરેટિંગ કમયયો

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 00:03)

અમે બેકટ્રેકિંગમાં જોઈશું. બેકટ્રેકિંગમાં અમે એક સમયે એક પગલું ઉકેલવા માટે વ્યવસ્થિત રૂપે શોધ કરીએ છીએ અને જ્યારે અમે એક મૂત અંતને હટાવતા હોઈએ છીએ ત્યારે અમે છેલ્લો પગથિયાનો અનાદર કરીએ છીએ અને આગળના વિકલ્પનો પ્રયાસ કરીએ છીએ.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 00:14)

હવે, બેકટ્રેકિંગ કરવાની પ્રક્રિયામાં અમને બધી શક્યતાઓ પેદા કરવાની જરૂર છે. દાખલા તરીકે યાદ રાખો કે જ્યારે આપણે દરેક રાણી માટે દરેક રાણી માટે દરેક શક્ય સ્થિતિ દ્વારા ચાલી રહેલી તમામ રાણીઓને છાપવાનો પ્રયાસ કરીએ છીએ, અને જો તે ફી હતી, તો અમે તેને અજમાવી લીધું અને જો ઉકેલ અંતિમ કેસમાં વિસ્તારીએ તો અમે તેને છાપીએ છીએ. હવે જો આપણે 8 રાણીઓ માટે વિચારતા ઉકેલો જોયાં, હાર પરની દરેક રાણી દરેક અન્ય રાણીના એક અલગ સ્તંભમાં છે. સ્તંભની સંખ્યા જો આપણે પંક્તિ દ્વારા હરોળમાં વાંચીએ છીએ, તો સ્તંભની સંખ્યા 0 થી N ની 1 ની ક્રમયય રહે છે. તેથી, દરેક સંખ્યા 0 થી N ની 1 ની ઉત્પત્તિ બરાબર એકવાર રાણી માટે સ્તંભ નંબર તરીકે થાય છે. તેથી, 8 ક્વીન્સ અથવા સમાન સમસ્યાઓ જેવી સમસ્યાનો ઉકેલ લાવવાનો એક રસ્તો વાસ્તવમાં તે તમામ ક્રમયયો પેદા કરે છે અને એક સમયે તેમને એક પ્રયાસ કરવાનું ચાલુ રાખે છે. આ નીચેના પ્રશ્નનો ઉદય આપે છે; જો આપણે 0 થી N ની બાદબાકી કરીએ છીએ તો આપણે કેવી રીતે આગળ ક્રમયય બનાવવું જોઈએ. આ તે વિશે આગામી નંબર તરીકે વિચારી રહ્યા છે, પરંતુ આ સંકેતોની મનસ્વી સંખ્યામાં હોઈ શકે છે. ધારો કે, આપણી પાસે અક્ષરો છે જે મીટર છે. તેથી, આ મૂળાક્ષરોનાં પ્રથમ 13 અક્ષરો છે અને આપણે શબ્દોના ક્રમમાં શબ્દોને ક્રમાંક તરીકે ઓર્ડર આપીએ છીએ, જો આપણે વિચારીએ કે તે આધાર તેર છે દાખલા તરીકે, અહીં બેસ ત્રીસમાં સંખ્યા છે અથવા અમુક વૈકલ્પિક ક્રમમાં મીટરનું વૈકલ્પિક પુનર્ચયના છે. હવે આપણે શું કરવા માગીએ છીએ, આ પછી તરત જ આગામી ક્રમાંકમાં તમારે શું કરવું જોઈએ તે પછીના પુનર્ચયના શું છે?

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 01:54)

આ સમસ્યાનું નિરાકરણ કરવા માટે આપણે સૌ પ્રથમ નિરીક્ષણ કરી શકીએ તે એ છે કે, જો અમારી પાસે આ પ્રકારના અક્ષરો અથવા અંકોનો ક્રમ હોય તો નાના ક્રમયય ક્રમ છે જેમાં તત્વોને ચડતા ક્રમમાં ગોઠવવામાં આવે છે. તેથી આપણે એકથી શરૂ કરીએ છીએ જે પછી નાના અને પછી બી અને સી અને તેથી વધુ છે અને આ એક કરતા ઓછી કોઈ ક્રમયય નથી. તેવી જ રીતે, સૌથી ક્રમયય ક્રમબદ્ધ છે જેમાં તમામ ઘટકો ઉતરતા ક્રમમાં છે, તેથી અમે સૌથી મોટું તત્વ મીટર સાથે શરૂ કરીએ છીએ અને અમે પાછળની બાજુએ કામ કરીએ છીએ. જો આપણે આગળના ક્રમાંકને શોધવા માગીએ છીએ તો આપણે શક્ય તેટલા ટૂંકા પ્રત્યયને શોધી કાઢવાની જરૂર છે જે વધેલો હોઈ શકે છે, તે શક્ય છે કે તે નંબરોની દ્રષ્ટિએ કરવું, પરંતુ ચાલો આપણે તેને અક્ષરો સાથે કરીએ. સૌથી ટૂંકું પ્રત્યય કે જે વધેલો હોઈ શકે છે તેમાં સૌથી લાંબી પ્રત્યય દ્વારા અનુસરવામાં આવતી વસ્તુની સંખ્યા વધારી શકાતી નથી. તેથી જ્યારે આપણે ઉદાહરણ દ્વારા કામ કરીએ છીએ ત્યારે આ થોડું સ્પષ્ટ થશે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 02:56)

અમે સૌથી લાંબો પ્રત્યય શોધવા માગીએ છીએ જે વધારી શકાતો નથી. તેથી, એક પ્રત્યય જે વધારી શકાતું નથી તે એક છે જે તેટલું મોટી છે કારણ કે તે સંભવતઃ હોઈ શકે છે, જેનો અર્થ એ છે કે તે ઉતરતા ક્રમમાં પહેલેથી જ છે જો તમે ઉદાહરણ જુઓ કે જે પહેલાં આપણે આગળના ક્રમચયને વ્યાખ્યાયિત કરવા માંગીએ છીએ, તો આપણે આ પ્રત્યય ઓમ્મજી શોધીએ છીએ કે આ પાંચ અક્ષરો ઉતરતા ક્રમમાં છે તેથી હું આનો ઉપયોગ કરીને કોઈ પણ મોટી ક્રમચય કરી શકતો નથી. તો, જો હું અક્ષરને ડી થી કેરના ઠીક ઠીક કરું તો આ સૌથી ક્રમનું ક્રમાંકન છે જે હું ડી થી ક્યુ સાથે પેદા કરી શકું છું. જો હું તેને બદલવા માંગું છું અને તેને વધારવાની જરૂર છે અને તેનો અર્થ હું વધારી રહ્યો છું, તો હું તેને આ લાલ બોક્સમાં વધારી શકતો નથી જેથી હું તેને વિસ્તૃત કરું સૌથી ટૂંકું પ્રત્યયને શોધો; પ્રત્યય k સાથે શરૂ થાય છે જ્યાં કંઈક વધે છે. હવે આપણે આ કેવી રીતે વધારીએ? ઠીક છે, આપણે શું કરવાની જરૂર છે તે છે કે હવે આપણે કહીએ છીએ કે અમારી પાસે K સાથે કોઈ વધુ સારી કામગીરી કરી શકાતી નથી તેથી આપણે ક્યાએ વધુ મોટું કરી શકીએ છીએ અને જે કંઈક મોટું છે તે સૌથી નાનું વસ્તુ હોવું જોઈએ જે આપણે બદલી શકીએ છીએ, તેથી આપણે તેની જમણી બાજુના આગામી સૌથી મોટા અક્ષરમાં k નું નામ બદલીશું. આ અક્ષરો એમ.એન. અને ઓ એ છે જે k કરતા વધુ છે જો હું તેને j અથવા i દ્વારા બદલું છું, મને એક નાની ક્રમચય મળશે જે હું ઈચ્છતો નથી, તેથી હું તેને mn અથવા o ના સ્થાને બદલી શકું છું, પરંતુ આમાંનું આ મીટર એ સૌથી નાનું છે હવે ક્રમ શરૂ કરો જ્યાં લંબાઈ છનો પ્રત્યય અક્ષર મીટર સાથે શરૂ થાય છે. અને પ્રત્યય વચ્ચે મીટર સાથે શરૂ થાય છે જેમા મને સૌથી નાનો જરૂર છે, તેનો અર્થ એ છે કે હું બાકીના અક્ષરોમાં કોન્જીને ચડતા ક્રમમાં ગોઠવવા માટે મને મીટરની શરૂઆત કરવા માટેનું સૌથી ક્રમચય આપું છું અને તેના પછી અક્ષરો કોન્જી છે. આ મને આ ક્રમચય આપે છે તેથી, મેં આ મીટર અહીં ખસેડી દીધું છે અને મેં હવે આ પત્રો લીધી છે અને ijkn o મેળવવા માટે ચડતા ક્રમમાં તેમને ફરીથી ગોઠવ્યાં છે. તેથી, આનો અર્થ એ છે આ ક્રમચય માટે આગળ ક્રમચય એ આ એક છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 04:55)

તેથી, અમે એલ્ગોરિધમનીક પ્રમાણે નીચે પ્રમાણે કરી શકીએ છીએ, આપણે શું કરવાની જરૂર છે તે પ્રથમ પ્રત્યયની ઓળખ કરે છે જે વધારી શકાય છે. અમે પ્રત્યયની શોધ કરીને શરૂ કરીએ છીએ જેને વધારી શકાતું નથી, એટલે આપણે ઉતરતા ક્રમમાં હોય ત્યાં સુધી પાછળથી જાઓ. તેથી અમે મૂલ્યો શોધી રહ્યાં છીએ કારણ કે તે વધારો કરે છે. તેથી, હું j કરતા નાની છે, j મીટર કરતાં નાનું છે, m એ કરતા ઓછું છે, n એ ઓ કરતા ઓછું છે, પરંતુ o k કરતા મોટો છે તેથી તેનો અર્થ એ છે અહીં સુધી આપણે એક પ્રત્યય છે જે વધારી શકાતો નથી અને આ પ્રથમ સ્થાન જ્યાં અમે એક ઇન્ક્રીમેન્ટ કરી શકો છો. આને પૂર્ણ કર્યા પછી હવે આપણે તેના જમણા અક્ષર દ્વારા k ને બદલવાની જરૂર છે જે આગામી મોટી છે. હવે આ એક બીટ છે કે આપણે એક પછી એક જઈએ છીએ, આપણે કહીએ છીએ કે n એ કરતા નાની અને તેથી આપણે ચાલુ રાખીએ છીએ, અને આપણે કહીએ છીએ કે કીએ કરતાં જે મોટું છે તેથી અમે અહીં બંધ કરીએ છીએ. તેથી આ આપણને કહે છે કે પત્ર એમ એ છે જે આપણે ઈચ્છીએ છીએ. અમે તેને એક સ્કેનમાં ઓળખી શકીએ છીએ, કારણ કે તે યાદ રાખવું તે ઉતરતા ક્રમમાં છે, તે ક્રમમાં ગોઠવવામાં આવે છે તેથી આપણે પ્રથમ સ્થાન શોધી શકીએ છીએ જ્યાં આપણે K કરતા નાના કંઈક કરતા વધુ કંઈક પર ઓળંગીએ છીએ અને તે સ્થાન છે પત્ર બદલવાની જરૂર છે. તેથી, તે સોર્ટ કરેલ સૂચિમાં કંઈક શામેલ કરવા જેવું છે. હવે આ કરવાથી, આપણે આ એમ અને કેનનું વિનિમય કરીએ છીએ, હવે આપણે આને ચડતા ક્રમમાં મુકવાની જરૂર છે, પરંતુ યાદ રાખો કે તે ઉતરતા ક્રમમાં હતો અને અમે ઉતરતા ક્રમમાં શું કર્યું, અમે મીટર દ્વારા કેવું બદલીએ પરંતુ કુંપની મિલકત શું છે? K મીટર કરતાં નાનું હતું પરંતુ જે કરતાં મોટી છે, ઓક્જી ઉતરતા ક્રમમાં રહે છે જો આપણે તે ચઢતા ક્રમમાં કન્વર્ટ કરવા માંગીએ છીએ તો અમને તેને સોર્ટ કરવાની જરૂર નથી, આપણે તેને પાછળ ઉતારવાની જરૂર છે, આપણે ફક્ત પાછળની જરૂર છે, તેથી તે આનો ફક્ત રિવર્સલ છે. આ બધામાં એક કોફિટ રસ્તો છે, જે આગળના ક્રમચયને શોધી

કાઢે છે, અંતથી પાછળથી જ ચાલો અને જુઓ કે જ્યારે ક્રમ વધતા અટકે છે. તેથી, જ્યાં આપણે પ્રથમ આને ઘટાડીએ છીએ તે પ્રત્યય છે કે જે તમે વધારવા માંગો છો, અલબત્ત જો આપણે બધી રીતે જઈએ અને પ્રથમ અક્ષર પર પાછા જઈએ અને અમને આ સ્થાન મળ્યું ન હોય તો આપણે પહેલાથી જ અંતિમ ક્રમચયમાં પહોંચી ગયા છીએ. વસ્તુઓ એકંદર યોજના એકવાર આપણને આ સ્થાન મળ્યું પછી આપણે તે પત્ર શોધવાનું શીખી શકીએ છીએ કે શોધને બરાબર કરવાથી તે સ્વેપ કરી શકાય છે. તેથી, અમે આ પ્રકારની વસ્તુને શોધીએ છીએ કે આ કેસમાં પોઝિશન મીટરને સ્વેપ કર્યા પછી સ્વેપ કરીને આપણે તેને નવા અક્ષર પછી મીટર બનાવીશું અને આપણે તે અક્ષર મીટર સાથે શરૂ કરાયેલ સૌથી નાનું ક્રમચય મેળવવા માટે તેને રિવર્સ કરીએ છીએ.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રોફેસર. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 06 લેકચર - 04 સેટ્સ, સ્ટેક્સ, ક્યુઓસ

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 00:01)

નાક્લાઉસ વેર્થમાં, પ્રોગ્રામિંગ ભાષાના શોધક પાસ્કલએ એલ્ગોરિધમ્સ વત્તા ડેટા સ્ટ્રક્ચર્સ ઇક્વલ્સ પ્રોગ્રામ્સ નામના અત્યંત પ્રભાવશાળી પુસ્તક લખ્યું હતું. તેથી, શીર્ષક અસરકારક કાર્યક્રમોનાં ઘટકો તરીકે બંને ગાણિતિક નિયમો અને ડેટા માળખાનો મહત્વ પર ભાર મૂકે છે. અત્યાર સુધી અમે કેટલાક વિગતવાર એલ્ગોરિધમનો જોયું છે. તેથી, હવે ચાલો આપણે કેટલાક વિશિષ્ટ ડેટા સ્ટ્રક્ચર્સને નજીકથી જોવું. ડેટા સ્ટ્રક્ચર્સ કે જે આપણે જોયાં છે કે જે પાયથોન માં બનેલા છે તે એરે અને યાદીઓ સાથે શરૂઆત કરે છે, જે ફક્ત મૂલ્યોની શ્રેણીઓ છે. અમે શબ્દકોશ પણ જોયાં છે જે કી મૂલ્ય જોડીઓ છે અને જે વિવિધ પ્રકારની માહિતી જાળવવા માટે ખૂબ ઉપયોગી છે. અન્ય એક ડેટા પ્રકારમાં બનાવવામાં આવે છે જે પિથનમાં ઉપલબ્ધ છે સમૂહ છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 00:48)

સેટ એ સૂચિની જેમ છે કે તમારી પાસે ડુપ્લિકેટ્સ નથી. પાયથોન માં, સેટ લખવાનું એક રીત આની જેમ કૌંસમાં લિસ્ટ લખવાનું છે. તેથી, અહીં અમે નામ રંગો સાથે સંકળાયેલ છે લાલ, કાળો, લાલ અને લીલા મૂલ્યોની સૂચિ. નોંધ લો કે તે સેટ કરવાથી, અમે વારંવાર લાલ મુક્યો છે, પરંતુ કારણ કે આ સેટ છે, ડુપ્લિકેટ લાલ આપમેળે દૂર થઈ જશે તેથી, જો આપણે નામના રંગોને છાપીએ, તો આપણે ફક્ત કાળા, લાલ અને લીલા યાદી મેળવીએ છીએ. હવે, ખાલી બોસ નોટેશન પહેલેથી જ વપરાય છે, ખાલી શબ્દકોશ માટે જો આપણે ખાલી સેટ બનાવવા માંગીએ છીએ, તો આપણે સેટ ફંક્શનને નીચે પ્રમાણે કોલ કરવો પડશે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 01:33)

તેથી, આપણે કોઈ દલીલો સાથે સુયોજિત કરવા માટે સમાન રંગો કહીએ છીએ. સૂચિ અને અન્ય માહિતી માળખાંની જેમ, અમે સાઇન ઇનનો ઉપયોગ કરીને સભ્યપદની ચકાસણી કરી શકીએ છીએ. તેથી, જો પહેલાંની યાદીઓમાં લાલ, કાળો અને લીલા હોય તેવા રંગોનો સેટ હોય, તો અમે કહીએ છીએ કે કાળા શબ્દનો ઉપયોગ કરીને રંગોમાં છે, પછી, વળતર મૂલ્ય સાચું. સામાન્ય રીતે આપણે સેટ ફંક્શનનો ઉપયોગ કરીને સેટમાં કોઈ પણ સૂચિને કન્વર્ટ કરી શકીએ છીએ. અમે જોયું કે જો અમે કોઈ ખાલી સમૂહ મેળવવા માટે કોઈ દલીલો આપીએ નહીં, પરંતુ જો આપણે ડુપ્લિકેટ્સ સાથે આ 1, 3, 2, 1, 4 જેવી સૂચિ આપીએ છીએ અને તેને નામના નંબરોમાં સોંપી દો, તો પછી તે એક સેટ છે ડુપ્લિકેટ્સ દૂર કરવામાં આવશે અને અમને એક સૂચિ મળશે, અમે 0, 1, 2, 3, 4 નો સમૂહનો એક નંબર મેળવીશું. ફરીથી નોંધ કરો કે જે સેટનો પ્રિન્ટ કરવામાં આવ્યો છે તે ક્રમમાં તમારે તે ઓર્ડર નહીં આપવો જોઈએ તે આ ખૂબ જ શબ્દકોશ સમૂહો જેવું છે; કોઈ ડુપ્લિકેટ્સ વગેરે નથી તેની ખાતરી કરવા માટે આંતરિક સંગ્રહ માટે શ્રેષ્ઠ છે. તેથી, આપણે સમૂહમાંના તત્વોના ક્રમ વિશે કંઈક ધારે નહીં. એક રસપ્રદ લક્ષણ એ છે કે શબ્દમાળા એ અસંખ્ય અક્ષરોની સૂચિ છે. તેથી, જો આપણે સમૂહને સ્ટ્રિંગ આપીએ છીએ, તો તે સમૂહ ફંક્શનનું નિર્માણ કરે છે, પછી તે એક સેટ ઉત્પન્ન કરે છે જે આ સેટમાંથી વ્યક્તિગત અક્ષરો ધરાવે છે. તેથી, જો

આપણે સેટ ફંક્શનમાં આ સ્ટ્રિન બનાના આપીએ તો, આપણે સમૂહમાં ડુપ્લિકેટ્સ વગર ત્રણ, વ્યક્તિગત અક્ષરો a, n અને b મેળવીશું.

(સ્લાઇડ સમયનો સંદર્ભ લો: 02:58)

તેથી, જેમ તમે અપેક્ષા રાખી શકો છો કે ગણિતમાં તેમના સમકક્ષ જેવા સેટ્સ મૂળભૂત કામગીરીને આધાર આપે છે, તેથી ધારો કે આપણે 1 થી 11 ની વચ્ચે તમામ વિચિત્ર સંખ્યાઓના સમૂહ તરીકે અમાન્ય સંખ્યાઓ સુયોજિત કરીએ અને મુખ્ય સંખ્યાઓ 1 અને 11 ના બધા મુખ્ય સંખ્યાઓનો સમૂહ 2 અને 11 ની વચ્ચે આ સમૂહ વિધેયનો ઉપયોગ કરીને જે આપણે પહેલાં જોયું. જો આપણે આ ઊભી બાર લખીએ તો આપણે બે સમૂહોનું જોડાણ મેળવી શકીએ છીએ. તેથી, વિચિત્ર સંઘ વડા તે ઘટકો હશે જે ક્યાં તો વિચિત્ર અથવા મુખ્ય છે તેથી, અમે તળિયે 3, 5, 7, 9, 11 થી ટોચની બેમાંથી એક મેળવીએ છીએ. અમે બન્ને સેટમાં બધા તત્વો મેળવીએ છીએ, પરંતુ કોઈપણ ડુપ્લિકેટ્સ વિના જો આપણે બે સમૂહોના આંતરછેદ માટે પૂછતા હોવ, તો આપણે આને દર્શાવવા માટે એમ્પરસેન્ડનો ઉપયોગ કરીએ છીએ. અમે તે બન્ને સેટ્સ, તે સેટ્સ, તે નંબરો, જે બન્ને વિચિત્ર અને મુખ્ય છે અને આ કિસ્સામાં 3, 5, 7 અને 11 માં મળે છે તે નોંધો. નોંધ કરો કે આ ક્રમાંકો કયા ક્રમમાં મુદ્રિત થાય છે તે મનસ્વી હોઈ શકે છે. સેટમાં તફાવત એવા તત્વો માટે પૂછે છે જે વિચિત્ર છે, પરંતુ પ્રાથમમાં નહીં. બીજા શબ્દોમાં કહીએ તો, અસ્પષ્ટ સંખ્યાઓ જે મુખ્ય નથી, આ સંગ્રહમાં 1 અને 9 માં વિચિત્ર સંખ્યાઓના ઉદાહરણો છે જે મુખ્ય નથી. અને આખરે, યુનિયનથી વિપરીત, જે બંને સેટમાં છે તે તત્વો ભેગો કરે છે, અમે એક વિશિષ્ટ કરી શકીએ છીએ અથવા જે બે સેટમાંના એકમાં બરાબર છે જો આપણે આ ગાર્ટ પ્રતીકનો ઉપયોગ કરીએ છીએ, તો પ્રથમ સેટમાંથી, 1 નો પ્રથમ સેટમાંથી અને બીજા સેટમાંથી 2 મળશે, કારણ કે 3, 5, 7 અને 11 બંને સેટમાં થાય છે. તેથી, અમે સમૂહો વિશે વધુ વાત નહીં કરે, પરંતુ તમે આ બિલ્ટ ઇન ઓપરેશન્સનો ઉપયોગ કરીને ડુપ્લિકેટ્સ વગરના મૂલ્યોના સંગ્રહનો સાચો રાખવા માટે વિવિધ સંદર્ભોમાં તેનો ઉપયોગ કરી શકો છો.

(સ્લાઇડ સમયનો સંદર્ભ લો: 04:40)

ચાલો આપણે જુદી જુદી રીતો જોઈએ કે જેમાં આપણે સિક્વન્સને મેનિપ્યુલેટ કરી શકીએ છીએ. એક યાદી જે આપણે જોયેલી છે તે ક્રમ છે જેમાં અમે મુક્તપણે સ્થાનો પર મૂલ્યોને દાખલ કરી અને કાઢી નાખી શકીએ છીએ. હવે, જો આપણે આના પર કેટલાક શિસ્ત લાદીશું તો, આપણે વિશિષ્ટ માહિતી માળખાં મેળવીશું જેમાંથી એક સ્ટેક છે. સ્ટેક પ્રથમ આઉટ યાદીમાં છેલ્લા છે. તેથી, અમે ફક્ત તે સ્ટેકથી જ દૂર કરી શકીએ છીએ જે અમે છેલ્લે ઉમેર્યું હતું. સામાન્ય રીતે આને બે ઓપરેશન આપવામાં આવે છે. જ્યારે આપણે સ્ટેક પર એક તત્વને દબાણ કરીએ છીએ, ત્યારે અમે તેને સ્ટેકની અંતમાં ઉમેરીએ છીએ અને જ્યારે અમે એક સ્ટેક પોપ કરીએ છીએ, ત્યારે અમે ઉમેરેલા અંતિમ મૂલ્યને નિષ્ક્રિય રીતે મેળવીએ છીએ. હવે, પાચથોન સૂચિમાં બિલ્ટ ઇન ઉપયોગ કરીને અમલ કરવાનું સરળ છે. આપણે ધારણ કરી શકીએ છીએ કે સ્ટેક્સ જમણી તરફ વધે છે. તેથી, અમે અધિકાર પર દબાણ અને અમે જમણી પોપ તેથી, પુશ કરો કે એસએક્સ ફક્ત એક્સ માટે એસ જોડશે તેથી, તમે બિલ્ટ-ઇન એપેન્ડ ફંક્શનનો ઉપયોગ કરી શકો છો જે સૂચિ માટે ઉપલબ્ધ છે, જ્યારે આપણે તેને ધક્કો માગીએ છીએ અને તે તારણ આપે છે કે પાચથોન ની યાદીઓમાં વાસ્તવમાં પોપ નામના ફંક્શનમાં બિલ્ટ ઇન છે જે છેલ્લા ઘટકને દૂર કરે છે અને તે પાછું આપે છે. અમારા માટે. તેથી, આપણે ફક્ત ડોટ પોટ કહીએ છીએ, જ્યાં એ સૂચિ છે અને આપણે બરાબર વર્તણૂક મેળવીએ છીએ જે અમારા સ્ટેકની અપેક્ષા રાખે છે.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 05:57)

એક સ્ટેક સામાન્ય રીતે ફરી યાદ આવવું ફંક્શન કોલ્સનો ટ્રેક રાખવા માટે ઉપયોગમાં લેવાય છે, જ્યાં આપણે કાર્યોની શ્રેણીમાંથી પસાર થવું હોય અને તે પછી, તે પહેલાંના કાર્યમાં પાછો બોલાવવામાં આવે છે. ખાસ કરીને જ્યારે અમે પાછા ટ્રેકિંગ કરીએ છીએ, અમારી પાસે વર્તન જેવું સ્ટેક છે કારણ કે જેમ

આપણે રાણીઓ ઉમેરીએ છીએ અને તેમને દૂર કરીએ છીએ, તાજેતરની ક્વિનને સ્ટેક પર દબાણ કરવા માટે અમારે અસરકારક રીતે શું કરવાની જરૂર છે, જેથી જ્યારે આપણે બેકટ્રેક કરીએ, ત્યારે અમે તેને પોપ કરી શકીએ છીએ અને છેલ્લા ચાલને પૂર્વવત્ કરો

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 06:29)

સૂચિનો ઉપયોગ કરવાનો અન્ય શિસ્તબદ્ધ માર્ગ કતાર છે સ્ટેકની વિપરીત, જે પ્રથમ વખત છેલ્લો છે, કતાર પ્રથમ આઉટ ક્રમમાં પ્રથમ છે. બીજા શબ્દોમાં કહીએ તો, અમે એક ઓવરને અંતે ઉમેરો અને અમે અન્ય ઓવરને અંતે દૂર આ એક કતાર જેવું છે જે તમે વાસ્તવિક જીવનમાં જુઓ છો, જ્યાં તમે પાછળની કતારમાં જોડાઓ છો અને જ્યારે તમારો વારો આવે છે, ત્યારે તમે કતારના વડા છો અને પછી તમે પીરસવામાં આવે છે. તેથી, ક્યૂ એ કતારના પાછળના ભાગમાં એક્સ ઉમેરશે અને q દૂર થશે જે q ના માથા પર છે તે ઘટક દૂર કરશે. ફરી એકવાર આપણે પાયથોન સૂચિનો ઉપયોગ કરી શકીએ છીએ અને તે એવું સ્વીકારવાનું અનુકૂળ છે કે તે એક કતારને રજૂ કરે છે તે સૂચિને જમણા ખૂણે ડાબી તરફ અને જમણી બાજુના વડા કરતાં તેના વડાને બદલે છે. આ કારણ છે કે આપણે પહેલાની જેમ પોપનો ઉપયોગ કરી શકીએ છીએ, પરંતુ હવે જ્યારે આપણે કોઈ કતારમાં દાખલ કરવા ઈચ્છીએ છીએ, ત્યારે આપણે આ સાથે આપેલ શામેલ કાર્યનો ઉપયોગ કરી શકીએ છીએ. અમે આ સ્પષ્ટપણે જોયું નથી, પરંતુ જો તમે દસ્તાવેજોમાંથી પસાર થઈ ગયા છો, તો તમને તે મળશે. જો મારી પાસે સૂચિ i છે અને જો હું બે દલીલો j અને x સાથે શામેલ કરું છું, તો તેનો અર્થ એ છે કે, j એ j ની પહેલા j એ મૂકવા માટે છે, જો તમે પોઝિશન 0 માં દાખલ કરો તો ખાસ કરીને પોઝિશન j એ પહેલા x ને મૂકો. યાદીમાં દરેક ઘટક પહેલાં કંઈક મૂકવા. તેથી, qq અલ્પવિરામ એક્સ ઉમેરો માત્ર એ જ છે જેમ કે q ડોટ શામેલ કરો 0 અલ્પવિરામ એક્સ. બીજા શબ્દોમાં, શરૂઆતમાં x ને દબાણ કરો. જો મારી પાસે આ ફોર્મ પર કતાર છે જેમાં કેટલાક મૂલ્યો v_1, v_2 અને તેથી વધુ હોય તો, આ શામેલ કાર્ય ફક્ત શરૂઆતમાં x મુકી દેશે અને જેમ આપણે પહેલા કહ્યું હતું, આપણે આ સંકેતનો ઉપયોગ કરવાનું પસંદ કર્યું છે તે છે અમે પછી યાદીનો છેલ્લો ભાગ દૂર કરવા માટે પોપનો ઉપયોગ કરી શકીએ છીએ. ક્યૂઓ અને સ્ટેક્સ બંને આંતરિક લિંક્સનો ઉપયોગ કરીને સરળતાથી અમલ કરી શકે છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 08:22)

તેથી, કતારનો એક સામાન્ય ઉપયોગ શોધ જગ્યા દ્વારા વ્યવસ્થિત રીતે શોધવામાં આવે છે. કલ્પના કરો કે અમારી પાસે એક લંબચોરસ મીટર કોસ એન ગ્રીડ છે અને અમારી પાસે ઘોડો છે. નાઈટ શેઝના ભાગરૂપે પોઝિશન એસએક્સ અલ્પવિરામથી શરૂ થાય છે. આ કિસ્સામાં, ઘોડો આ લાલ પ્રતીક દ્વારા સૂચિત થયેલ છે. તેથી, આ અમારી ઘોડો છે હવે, ઘોડો ચાલ, જો તમે ચેસથી પરિચિત હોવ તો બે ચોરસ ઉપર અને એક ચોરસ ડાબા ખસેડો. આ એક ઘોડો ચાલ છે એ જ રીતે, આ એક ઘોડો ચાલ છે; તેવી જ રીતે આ એક ઘોડો ચાલ અને તેથી પર છે. તેથી, ઘોડો ચાલ બે ચોરસ એક દિશામાં ખસેડવાનો સમાવેશ થાય છે, પછી એક ઓસમગ્ર ક્વાર્ટર તેથી, આ બધી જગ્યાઓ છે કે જે આ પ્રારંભિક સ્થિતિથી પહોંચી શકાય છે, જ્યાં ઘોડો ખસેડવા તેમાંથી આઠ આઠ છે. તેથી, અમારું સવાલ એ છે કે આપણી પાસે આ લાલ શરૂઆતનું ચોરસ છે અને આપણી પાસે એક લીલું ડાયમન્ડ છે જે લક્ષ્ય સ્કવેરને સૂચવે છે. શું હું ધ્રુવડ ડાયમન્ડથી લાલ ચોરસથી લઈને ગ્રીન ડાયમન્ડ સુધીનો હુકમ ચલાવી શકું? તેથી, આ કરવા માટેનો એક રસ્તો એ છે કે ચોરસની સૂચિને વધારીને રાખી શકો છો. તેથી, પ્રથમ તબક્કે અમે આ 8 ચોરસનું પરીક્ષણ કરીએ છીએ જે આપણે શરૂઆતની સ્થિતિમાંથી એક ચાલનો ઉપયોગ કરીને કહી શકીએ છીએ અને અમે તેને ચોરસ તરીકે ચિહ્નિત કરીએ છીએ જે એક પગથિયાં સુધી પહોંચવા માટે અમને ઉપલબ્ધ છે. હમણાં, અમે ઉદાહરણ તરીકે, ઉપરના ડાબી બાજુમાંથી એક પસંદ કરી શકીએ છીએ અને અન્વેષણ કરી શકીએ છીએ કે આપણે ત્યાંથી કઈ રીતે પહોંચી શકીએ. તેથી, દાખલા તરીકે, જો આપણે આ ચોરસથી શરૂ કરીએ અને હવે આપણે તેના પડોશીઓને શોધીએ છીએ, તો તેના કેટલાક પડોશીઓ ગ્રીડની બહાર છે. તેથી, અમે તેમને ફેંકી દો અમે ગ્રીડની અંદર ફક્ત તે પડોશીઓને જ

રાખીએ છીએ અને તેમાંના એક નોટિસ અમને તે સ્થળ પર પાછા લાવે છે જ્યાંથી અમે શરૂ કર્યું છે. હવે, ઉદાહરણ તરીકે, અમે આ ચોરસને પસંદ કરી શકીએ છીએ અને જો આપણે શોધી કાઢીએ કે તે ફરીથી 8 પડોશીઓ પેદા કરશે અને આ પડોશીઓમાંના કેટલાક પીળા પડોશીઓને ઓવરલેપ કરશે. હું તે પીળા અને લીલો અને ખાસ કરીને સાંધાના ભાગથી સૂચિત કરું છું કારણ કે તે બંને પ્રારંભમાં પ્રારંભ બિંદુથી પહોંચી ગયા હતા. અલબત્ત, પ્રારંભિક બિંદુ બંનેમાંથી પહોંચે છે. પ્રારંભિક બિંદુ બંને રંગીન પીળો અને લીલા છે તેથી, જેમ તમે ચિહ્નિત કરવાની પ્રક્રિયામાં જોઈ શકો છો, આ ચોરસ, કેટલીક વખત આપણે બે વાર ચોરસને ચિહ્નિત કરીએ છીએ અને અમારે સુનિશ્ચિત કરવાની પદ્ધતિસરની રીત હોવી જોઈએ કે આપણે આ યોગ્ય રીતે કરીએ અને લૂપમાં ન આવો.

(સ્લાઇડ સમય જણાવો: 10:34)

તો, આપણે શું કરવાનો પ્રયાસ કરીએ છીએ તે નીચે મુજબ છે. તેથી, પ્રથમ પગલામાં આપણે તમામ ચોરસને એક તબક્કે પ્રાપ્ય બિંદુ એસએક્સ અલ્પવિરામથી ખસેડી શકીએ છીએ. પછી, અમે એક ચોરસમાં એક્સ 1 થી બધા ચોરસ સુધી પહોંચી શકાય તેવું ચિહ્નિત કરવાનો પ્રયત્ન કરીએ, આને 2 બોલાવીએ, અને પછી આપણે એક ચાલમાં એક્સ 2 થી બધા ચોરસ સુધી પહોંચી શકાય તેવું શોધ કરીશું, આને x 3 અને ટૂંક સમયમાં ફોન કરો. હવે, એક સમસ્યા એ છે કે આપણે જોયું છે કે આપણે એક દિશામાં x 1 થી x2 સુધી પહોંચી શકીએ, પછી ચોરસ જે x 2 થી પહોંચી શકે છે તેમાં એક્સ 1 માં ચોરસનો સમાવેશ થશે. તેથી, અમે કેવી રીતે ખાતરી કરીએ છીએ કે અમે અન્વેષણ કરતા નથી પહેલાથી ચિહ્નિત ચોરસ અને વર્તુળોમાં ગોળ અને ગોળાકાર છે અને આ પ્રશ્ન સાથે સંબંધિત છે કે કેવી રીતે બંધ કરવું તે અમે કેવી રીતે જાણીએ છીએ

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 11:40)

અલબત્ત આપણે જાણીએ છીએ કે આપણે લક્ષ્ય ચોરસ શોધી રહ્યા છીએ, જો આપણે ક્યારેય લક્ષ્ય ચોરસ ચિહ્નિત કર્યું હોય, તો અમે રોકી શકીએ છીએ. બીજી તરફ, શક્ય છે કે લક્ષ્ય સ્ક્વેર પહોંચવા યોગ્ય ન હોય. આ કિસ્સામાં, અમે વિનામૂલ્યે જઈને ચાલુ રાખી શકીએ છીએ કે અમે નિરંતર આગળ વધી રહ્યા છીએ અને અમે ક્યારેય લક્ષ્ય ચોરસ સુધી પહોંચી શકતા નથી. તેથી, અમે કેવી રીતે રોકવું જ્યારે ખબર નથી? તેથી, આ માટે કતાર ખૂબ જ ઉપયોગી છે. આપણે જે કરીએ છીએ તે કોઈ પણ સમયે કોષોની કતારમાં જાળવી રાખવામાં આવે છે જેનું સંશોધન કરવું જોઈએ. શરૂઆતમાં કતારમાં માત્ર પ્રારંભ નોડ છે જે એસએક્સ અલ્પવિરામ છે. દરેક સમયે અમે કતારના વડાને દૂર કરીએ છીએ અને અમે તેના પડોશીઓને શોધી કાઢીએ છીએ, પરંતુ જ્યારે આપણે તેના પડોશીઓને શોધીએ છીએ, ત્યારે અમે આ પડોશીઓને ચિહ્નિત કરીએ છીએ. તેમાંના કેટલાક પહેલેથી ચિહ્નિત થઈ શકે છે તેથી, અમે કુહાડી પર ધ્યાન આપીએ છીએ, એયૂ, અમે કતારના શિરમાંથી દૂર કરીએ છીએ અને અમે એક સ્ક્વેર પર તમામ ચોરસ પહોંચી શકાય તેવું જુઓ. તેથી, પહોંચી શકાય તેવો અર્થ છે કે હું એક ઘોડો ખસેડી શકું છું અને ત્યાં જઈ શકું છું અને આ નાઈટ ચાલના પરિણામથી મને બોર્ડમાંથી બહાર ના આવે છે તેથી, હું આ તમામ ચોરસને માર્ક કરું છું જે કુહાડી અને એઈથી પહોંચી શકાય છે, જેમાંથી કેટલાક પહેલેથી જ ચિહ્નિત થયા હતા, જેમાંથી કેટલાક હમણાં જ ચિહ્નિત થયા છે. તેથી, હું શું કરું છું, હું જે લોકોનું નવું ચિહ્નિત કરું છું તે લઈએ છીએ અને તેમને કતારમાં ઉમેરીને કહીએ છીએ કે આ નવા ચિહ્નિત છે. હવે હું આ ચોરસનું અન્વેષણ કરું છું કે જેનાથી હું ત્યાંથી પહોંચી શકું છું. તેથી, આ બાંધધરી આપે છે કે જે એકવાર પહોંચી ગયું છે તે કતારમાં ફરી ક્યારેય પાછું આવશે નહીં. છેલ્લે, અમે કતાર ખાલી હો ત્યાં સુધી ચાલુ રાખીએ છીએ. જ્યારે કતાર ખાલી હોય, ત્યાં કોઈ નવા વર્ગ ઉમેરાયા નથી કે જે ઉમેરવામાં આવ્યા તે પહેલાં તેને ચિહ્નિત કરવામાં આવ્યા છે. તેથી, અન્વેષણ કરવા માટે વધુ કંઈ નથી અને અમે દરેક સ્ક્વેરમાં જઈએ છીએ જે આપણે મુલાકાત લઈ શકીએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 13:04)

અહીં આ માટે કેટલાક પિથન સ્યુડો કોડ છે. આપણે એસએક્સ, એસઆઈએનથી ટીએક્સ, ટી વાયથી શોધવું છે. અમે ધારીએ છીએ કે આપણે આપણી કિંમતો m અને n આપેલ છે જે અમારા ગ્રીડમાં પંક્તિઓ અને કોલમ્સની સંખ્યા દર્શાવે છે. તેથી, આપણે શું કરીએ શરૂઆતમાં આપણે 0 ની ચિહ્નિત એરે સુયોજિત કરી છે. આ સૂચિ ગ્રેસ નોટેશન વાદ રાખો. તે કહે છે કે શ્રેણી n માં i માટે n એ અમને z શુન્યોની સૂચિ આપે છે અને અમે શ્રેણી મીટરમાં j માટે આ m વખત કરીએ છીએ. તેથી, હું M બ્લોક ધરાવતી યાદી અને દરેક બ્લોક ધરાવતી એન ઝરોસ મળશે. આ કહે છે કે શરૂઆતમાં કશું ચિહ્નિત નથી. હવે, આપણે કહીએ છીએ કે આપણે પ્રારંભ નોડને માર્ક કરીએ છીએ અને અમે કતારમાંથી પ્રારંભ નોડ દાખલ કરીએ છીએ. હવે, લાંબી કતાર ખાલી નથી, અમે કતારમાંથી એક તત્વ પોપ કરીએ છીએ. આ કિસ્સામાં એસએક્સ, એસઆઈ બહાર આવશે. હવે, એક એવું કાર્ય છે જે અમે લખ્યું નથી, પરંતુ જે તમામ પડોશીઓની તપાસ કરશે કે હું કુહાડીથી પહોંચી શકું છું અને મને આવા નોડોના જોડાની યાદી આપી શકું છું. દરેક પાડોશી $nx \ ny$ માટે જો તે ચિહ્નિત નથી, તો હું તેને માર્ક કરીશ અને હું તેને કતારમાં દાખલ કરીશ, જમણે. તેથી, હું કતારમાંથી એક ઘટકને બહાર કાઢવા માટે શોધ કરું છું, તેના તમામ પડોશીઓને જુઓ કે જે ચિહ્નિત નથી, હું માર્ક કરું છું અને તેમને કતારમાં પાછું મૂકીશ અને છેવટે આ કિસ્સામાં હું તપાસ કરું છું કે હું માર્ક કરીશ કે નહીં મધ્યમાં tx અથવા ty . હું જાણું છું કે જો મારી પાસે અમુક સમયે ચોરસનો મર્યાદિત સેટ છે, તો આ પ્રક્રિયાને બંધ કરવી પડશે. અંતે હું પાછો પરત કરીશ કે હું લક્ષ્ય નોડ ટીએક્સ પર ચિહ્નિત કરેલા મૂલ્યને પાછો આપીશ, ટી વાય. તેથી, જો હું તેને પહોંચીશ, તો તે પાછું 1 આપશે જે સાચું છે. જો તે પહોંચી ન જાય, તો તે 0 આપશે જે ખોટા છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 14:53)

ચાલો આ કેવી રીતે કામ કરે છે તેનું એક ઉદાહરણ જોઈએ. તેથી, અહીં આપણી પાસે ત્રણ દ્વારા ત્રણ ગ્રીડ છે. યાદ રાખો કે કોશિકાઓ 1 0 2 2 અને 0 1 2 છે અમારા નામકરણ સંમેલન દ્વારા આપણે ટોચનું કેન્દ્ર ચોરસથી શરૂ કરવા માગીએ છીએ. આ અમારું સ્રોત છે અને અહીં કેન્દ્રમાં અમારું લક્ષ્ય છે તો, ચાલો આ બધા ગુણને ભૂંસી નાખીએ અને આ વસ્તુને આપણે અપેક્ષા રાખીએ. તેથી, અમે કહીએ છીએ કે શરૂઆતમાં કતાર કે જે આપણે સ્રોત નોડ તરીકે કરવા માંગીએ છીએ અને અમે ગ્રીડમાં સ્રોત નોડને ચિહ્નિત કરીએ છીએ. માર્કિંગ એ લાલ ચિહ્ન દ્વારા દર્શાવેલ છે. તેથી, આ રીતે આપણે પ્રારંભ કરીએ છીએ તેથી, અમારું પ્રથમ પગલું ક્યુમાંથી તેને દૂર કરવું અને તેના પડોશીઓને શોધવું. હવે, તેના પડોશીઓ 2, 0 અને 2 છે. આનો અર્થ એ કે હવેથી આપણે આ કોંસને દૂર કરીશું કારણ કે તે વધુ હેરાન છે. તેથી, તમે તેને આની જેમ વૃદ્ધિ કરો છો. તો, આપણે કહીએ છીએ કે મારી કતારમાં 2 અલ્પવિરામ 0 અને 2 અલ્પવિરામ 2 છે. આ મારી સમાંતર શોધવાની દિશામાં કતાર છે. દરેક પગલે હું હવે કતારના પ્રથમ ઘટકને દૂર કરીશ અને તેના પાડોશીઓને શોધું છું. જ્યારે હું 2 0 ના પડોશીઓને શોધું છું, ત્યારે મને તેમાંથી એક મળશે તે અલબત્ત છે જ્યાંથી હું તેને શરૂ કરું છું હું ફક્ત અનમાર્ક પડોશીઓને જ જોઉં છું તેથી, એક અશક્ત પાડોશી 1 છે. હું કતારના અંતે તે પાછી ઉમેરીશ. હવે, આગળ વધવાથી હું કતારના આગળનો ભાગ લઈશ જે 2 2 છે અને તેના પાડોશીઓને જુઓ. તેથી, 2 2 ફરીથી મૂળ વસ્તુમાં પાછા જઈ શકે છે અને તેની પાસે અહીં એક નવી વસ્તુ છે જે 1 0 છે. તેથી, આ રીતે ચાલુ રાખવાથી હું 1 2 દૂર કરું છું જે તે એક છે અને પછી તેના પડોશીઓને જુઓ. તેથી, તેના પડોશીઓમાંથી એક 2 0 છે, પરંતુ તેમાંથી એક 0 0 છે. તેથી, મને એક નવી પાડોશી 0 0 મળે છે અને પછી, હું કતારમાં 0 0 લઈને ચાલુ રાખું છું. તેથી, 1 0 આ એક છે. તેથી, તેની પાસે એક નવો પડોશી નીરક્ષણ છે જે તે એક છે. તેથી, મારી કતારમાં હવે 0 0 છે ત્યારબાદ 0 2. પછી, જ્યારે હું 0 0 ની શોધ કરું છું, ત્યારે મને આ પાડોશીને 2 અંશ મળે છે. હવે, જ્યારે હું 0 થી દૂર કરું છું જે આ એક છે, મને લાગે છે કે આ બંને પડોશીઓ મેં શોધ્યું તેથી, હું કંઈ ઉમેરી નથી હું 2 સાથે ચાલુ રાખું છું 1. ફરીથી, હું તેના બંને પડોશીઓને શોધી કાઢું છું અને કંઈ પણ કરતો નથી. હવે આ બિંદુએ, કતાર ખાલી છે અને કતાર ખાલી હોવાને કારણે, હું બંધ કરું છું અને મને લાગે છે કે મારું વ્યાજ એટલે કે 1 1 ચિહ્નિત નથી. તેથી, આ કિસ્સામાં લક્ષ્ય સ્રોત નોડથી પહોંચી શકાય તેવું નથી.

(સ્લાઈડ ટાઇમ નો ઉલ્લેખ કરો: 17:35)

આ વાસ્તવમાં બૃહદ પ્રથમ શોધનો એક ઉદાહરણ છે જે તમે આલેખને જોશો તો તમે અભ્યાસ કરશો, પરંતુ તે ફક્ત સમજાવે છે કે એક કતાર વ્યવસ્થિત રીતે તમે કેવી રીતે શોધ દ્વારા શોધખોરી છો તેનો સાચવી રાખવાનો સરસ ઉપાય છે જગ્યા

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 17:47)

ડેટા સ્ટ્રક્ચર્સનો સારાંશ આપવા માટે માહિતીને ગોઠવવાનાં રીતો છે જે ચોક્કસ સંદર્ભોમાં કાર્યક્ષમ પ્રોસેસિંગને મંજૂરી આપે છે. તેથી, આપણે જોયું છે કે પાયથોન માં સેટ્સની સૂચિની બિલ્ટ-ઇન અમલીકરણ છે, પણ આપણે જોયું છે કે આપણે સિક્વન્સ લઈ શકીએ છીએ અને તેમને બે માળખાગત રીતે વાપરી શકીએ છીએ. તેથી, સ્ટેક એ પ્રથમ-આઉટ સૂચિની છેલ્લી યાદી છે અને અમે તેનો ઉપયોગ રિકર્સિવ કોમ્પ્યુટેશન્સનો ટ્રેક રાખવા માટે કરી શકીએ છીએ અને ક્યુઝ પ્રથમ-પ્રથમ સૂચિ છે કે જે બૃહદ પ્રથમ સંશોધન માટે ઉપયોગી છે.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રો. માધવન મુકુંદ,
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 06 લેક્ચર - 05 પ્રાધાન્યતા કતાર અને ઢગલાઓ

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 00:02)

ચાલો ડેટા માળખું જોઈએ કાર્ય સમયપત્રકને સંડોવતા સમસ્યા જોબ શિડ્યુલર અગ્રતા સાથે બાકી નોકરીઓની યાદી જાળવે છે હવે, જોબ શિડ્યુલરને કોઈપણ બિંદુએ ચલાવવા માટે આગામી નોકરી પસંદ કરવી પડશે. તેથી, જ્યારે પણ પ્રોસેસર મફત હોય ત્યારે તે નોકરીને પસંદ કરે છે, નહીં કે નોકરી જે સૌથી વહેલી પહોંચે છે, પરંતુ યાદીમાં મહત્તમ અગ્રતા ધરાવતી વ્યક્તિ અને તે પછી તે સુનિશ્ચિત કરે છે. નવી નોકરીઓ યાદીમાં જોડાઈ રહી છે, પ્રત્યેક પોતાની પ્રાથમિકતાની સાથે અને તેમની પ્રાધાન્યતા પ્રમાણે તેઓ અગાઉની સાથે જોડાયેલા હોઈ શકે તેવી અન્ય નોકરીઓ આગળ પ્રોત્સાહન મેળવશે. તેથી, અમારું પ્રશ્ન છે કે કેવી રીતે સુનિશ્ચિત બાકી રહેલી નોકરીઓની સૂચિ અને તેની પ્રાથમિકતાઓને જાળવવી જોઈએ? જેથી તે હંમેશા આગામી સુનિશ્ચિત કરવા માટે સૌથી વધુ પ્રાધાન્યવાળી વ્યક્તિને ખૂબ ઝડપથી ખેંચી શકે છે.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 00:47)

આ એક કતારની જેમ છે, પરંતુ એક કતારમાં જેમાં અન્ય કોઈ લાક્ષણિકતાને આધારે વસ્તુઓની પ્રાધાન્ય નથી કે જ્યારે તેઓ આવ્યા તેથી, અમે જોયું કે એક સામાન્ય કતાર એ પ્રથમ-આઉટ ઓબ્જેક્ટ છે, જે પહેલા પ્રથમ રજા પર આવે છે. અગ્રતા કતારમાં, તેઓ તેમની અગ્રતા અનુસાર છોડી દે છે અગ્રતા કતાર સાથે સંકળાયેલ બે કામગીરીઓ છે, એક મહત્તમ કાઢી નાંખો છે કતારમાં કાઢી નાખવા માટે અમે હમણાં જ કહ્યું હતું કે અમે કતારના માથા પર તત્વ લઈએ છીએ. મહત્તમ કાઢી નાખવા માટે આપણે કતારમાં જોવાની જરૂર છે અને નોકરીને સૌથી વધુ અગ્રતા સાથે ઓળખી કાઢીને દૂર કરવી પડશે. અલબત્ત, બે જુદી જુદી નોકરીઓની અગ્રતા તે જ હોઈ શકે છે કે જેમાં આપણે કોઈ પણ એક પસંદ કરી શકીએ અને અન્ય ઓપરેશન છે જે આપણે સામાન્ય રીતે કહીએ છીએ કતારમાં ઉમેરીએ છીએ અને દાખલ કરીએ છીએ ફક્ત યાદીમાં નવી નોકરી ઉમેરે છે અને દરેક નોકરી જ્યારે તે ઉમેરવામાં આવે ત્યારે તેની સાથે આવે છે તેની પોતાની પ્રાથમિકતા છે

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 01:39)

આપણે પહેલેથી જ અભ્યાસ કરી લીધેલ રેખીય માળખાં પર આધારિત છે, આપણે આ કાર્યને યાદી તરીકે જાળવવા વિશે વિચારી શકીએ છીએ. હમણાં, જો તે એક કમમાંગોઠવાયેલનથી સૂચિ છે, જ્યારે અમે કંઈક કતારમાં ઉમેરીએ તો અમે તેને ફક્ત કોઈ પણ સ્થાનમાં ઉમેરી શકીએ છીએ. આ સતત સમય લે છે; તેમ છતાં, કાઢી નાખવું મહત્તમ કરવા માટે આપણે સૂચિમાં સ્કેન કરવું પડશે અને વધુમાં વધુ ઘટકની શોધ કરવી પડશે અને જેમ આપણે એક કમમાંગોળાં યાદીમાં જોયું છે, તે યાદીમાં મહત્તમ ઘટક શોધવા માટે અમને ઓર્ડર લેશે, કારણ કે આપણે બધાને સ્કેન કરવું પડશે આઇટમ્સ અન્ય વિકલ્પ સૂચિને સોર્ટ રાખવાનો છે. આ મહત્તમ કાઢી નાખવામાં મદદ કરે છે, દાખલા તરીકે, જો આપણે તેને ઉતરતા કમમાં ગોઠવીએ છીએ તો સૂચિનું પહેલું તત્વ હંમેશા સૌથી મોટું ઘટક છે. જો કે, અમે જે કિંમત ચૂકવીએ છીએ તે શામેલ કરવા માટે છે કારણ કે સોર્ટ ઓર્ડરને જાળવી રાખવા માટે જ્યારે આપણે એલિમેન્ટ શામેલ કરીએ છીએ તો આપણે તેને યોગ્ય સ્થાને મુકવું પડશે અને જેમ આપણે સોર્ટિંગ

સોઈમાં જોયું તેમ, શામેલ કરો, રેખીય સમય લેશે. તેથી, ટ્રેડ-ઓફ તરીકે અમે ક્યાં તો ડિલિટ માટે મહત્તમ અથવા રેખીય સમય કાઢી નાખવા માટે રેખીય સમય લે છે. જો આપણે કોઈ નોકરીઓ અથવા કતારમાંથી એક રસ્તો દાખલ કરવા અને છોડીને જતા હોઈએ છીએ તો આપણે આ n નોકરીઓ પર પ્રક્રિયા કરવા માટે ખર્ચ કરતા સ્કેલેડ સમયને સમાપ્ત કરીએ છીએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 02:55)

ડેટાને એક પરિમાણીય માળખામાં રાખવાની આ મૂળભૂત મર્યાદા છે. ચાલો આપણે બે પરિમાણીય માળખાં જોઈએ; સૌથી મૂળભૂત બે પરિમાણીય માળખું જે આપણે વિચારી શકીએ તે દ્વિસંગી વૃક્ષ છે. દ્વિસંગી વૃક્ષમાં ગાંઠોનો સમાવેશ થાય છે અને દરેક નોડમાં તેની સંગ્રહિત કિંમત હોય છે અને તે સંભવતઃ ડાબી અને યોગ્ય બાળક હોય છે. અમે રુટથી શરૂ કરીએ છીએ જે વૃક્ષની ટોચ છે અને ત્યારબાદ દરેક નોડમાં 1 કે 2 બાળકો હશે. એક નોડ જેને કોઈ બાળકો ન હોય તે પાંદડા કહેવાય છે અને પછી તે બાળકના સંદર્ભમાં આપણે માતાપિતા નોડને તેના ઉપરના નોડને બોલાવીએ છીએ અને અમે બાળકોને ડાબી બાળક અને યોગ્ય બાળક તરીકે વર્ણવતા છીએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 03:40)

તેથી, અમારું ધ્યેય એક અગ્રણી કતારને બાઈનરીના એક ખાસ પ્રકાર તરીકે જાળવવાનું છે, જેને આપણે ઢગલો કહીશું. આ વૃક્ષ સંતુલિત થશે. એક સંતુલિત વૃક્ષ એ છે કે જેમાં દરેક બિંદુએ ડાબી અને જમણી બાજુઓ આશરે લગભગ સમાન માપ છે. આને કારણે તે તારણ આપે છે કે સંતુલિત વૃક્ષમાં, જો અમારી પાસે n નોડ હોય તો વૃક્ષની ઊંચાઈ લઘુગણક હશે કારણ કે યાદ રાખો કે ઊંચાઈ દરેક સ્તર સાથે ડબલ્સ છે અને જેના પરિણામે આપણે લોગ એ સ્તરોમાં n ગાંઠો ફિટ કરી શકીએ છીએ. પૂરી પાડવામાં તે સંતુલિત છે અને કારણ કે તે ઊંચાઈ લોગ n છે, અમે બંને દાખલ કરો અને ક્રમમાં લોગ n સમય મહત્તમ કાઢી નાખશે. આનો અર્થ એ છે કે જો આપણે કતારમાંથી એન એલિમેન્ટોને એડ કરવા અને કાઢી નાખવા પડશે, તો એકંદરે આપણે એન લોગ એન ને સ્કેલેડમાંથી જઈશું અને ઢગલો વિશે અન્ય સરસ સુવિધા એ છે કે આપણે અગાઉથી n ને ઠીક કરવાની જરૂર નથી કારણ કે આ કામ કરશે. ઢગલો વધે છે અને ઘટતો જાય છે તેથી અમને ખબર નથી કે n શું છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 04:44)

ઢગલો શું છે? હીપ બે ગુણધર્મ સાથે દ્વિસંગી વૃક્ષ છે, પ્રથમ ગુણધર્મ માળખાકીય છે: જે મૂલ્યો જે વૃક્ષમાં સંગ્રહિત છે, યાદ રાખો કે પાંદડા, દ્વિસંગી વૃક્ષના નોડમાં 0 બાળકો, 1 બાળકો અથવા 2 બાળકો હોઈ શકે છે. તેથી, આપણે સામાન્ય રીતે ખૂબ અસમાન માળખામાં હોઈ શકીએ છીએ. ઢગલો પાસે ખૂબ જ નિયમિત માળખું હોય છે જ્યારે અમારી પાસે ઢગલો હોય છે ત્યારે આપણી પાસે દ્વિસંગી વૃક્ષ હોય છે જેમાં અમે દરેક સ્તર ઉપરથી નીચે, ડાબેથી જમણે ભરીએ છીએ બીજા શબ્દોમાં કહીએ તો, કોઈ પણ તબક્કે ઢગલોમાં ભરેલા સ્તરોનો સમાવેશ થાય છે અને ત્યારબાદ તળિયે આપણે ગાંઠો ડાબેથી જમણા ભરેલાં હોય છે અને પછી કદાચ કેટલાક અલ્પ ગાંઠો. ઢગલો સાથેની બીજી મિલકત, પ્રથમ મિલકત માળખાકીય છે, તે ફક્ત આપણને કહે છે કે કેવી રીતે કિંમતો ઢગલામાં દેખાય છે. આ નોડમાં, ઉદાહરણ તરીકે, આ ચિત્રમાં વાદળી ગાંઠો તે છે જે મૂલ્ય ધરાવે છે અને ખાલી ગાંઠો નીચે ખુલ્લા વર્તુળો સાથે દર્શાવવામાં આવે છે. ઢગલા વિશેની બીજી મિલકત એ પોતે મૂલ્યો છે. તેથી, આ કેસમાં ઢગલો મિલકત આપણે મહત્તમ ઢગલો મિલકતને કહીએ છીએ કારણ કે અમે મહત્તમ મૂલ્યોમાં રસ ધરાવીએ છીએ તે કહે છે કે દરેક મૂલ્ય તેના 2 બાળકોની કિંમતો કરતાં મોટી છે. તેથી, દરેક નોડ પર જો તમે મૂલ્ય જુઓ છો અને અમે ડાબા બાળકની કિંમત અને યોગ્ય બાળકને જોતા હોવ તો માતાપિતા બંને બાળકો કરતાં વધુ મૂલ્યવાન હશે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 06:06)

ચાલો આપણે કેટલાક ઉદાહરણો જોઈએ. અહીં ચાર નોડ ઢગલો છે, કારણ કે તેમાં ચાર ગાંઠો છે જે આપણે પ્રથમ સ્તરમાં રૂટને ભરીએ છીએ અને છેવટે, બીજા સ્તર પર અમારી પાસે ફક્ત એક નોડ છે જે ડાબી બાજુએ છે અને નોંધવું છે કે મૂલ્યો યોગ્ય રીતે ક્રમાંકિત છે, 24 એ 11 અને 7, 11 તેના એકમાત્ર બાળક કરતાં મોટી છે, 7 ના બાળકો નથી. તેથી, ત્યાં કોઈ અવરોધ નથી. અહીં બીજું એક ઉદાહરણ છે જ્યાં નીચેનું સ્તર વાસ્તવમાં પૂર્ણ થયું છે, અહીં આપણી પાસે 7 ગાંઠો છે અને એક વાર ફરી દરેક બિંદુએ આપણે જોઈ શકીએ છીએ કે 11 10 કરતા વધારે અને 5, 7 એ 6 અને 5 કરતા મોટો છે. તેથી, અમારી પાસે વેલ્યુ પ્રોપર્ટી છે - માળખાકીય મિલકત સાથે મહત્તમ ઢગલો મિલકત.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 06:42)

અહીં એવી વસ્તુનું ઉદાહરણ છે કે જે માળખાકીય રીતે ઢગલો નથી કારણ કે તે એક ઢગલો છે, આપણે તેને ઉપરથી નીચે સુધી, ડાબેથી જમણે ભરવું જોઈએ. તેથી, આપણી પાસે 7 ની ડાબી બાજુ નોડ હોવું જોઈએ તે પહેલાં આપણે એક યોગ્ય બાળક ઉમેરવું જોઈએ, આ ઢગલો નથી.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 06:59)

એ જ રીતે, અહીં આપણે 7 ના જમણા બાળકને ભરી તે પહેલાં નોડ 8 અહીં ઉમેરાઈ શક્યું ન હતું. તેથી, ફરી એકવાર આથી જમણી, ઉપરથી નીચે, , આ ઢગલો નથી.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 07:14)

આ ચોક્કસ વૃક્ષ એક ઢગલાના માળખાકીય મિલકતને સંતુષ્ટ કરે છે કે તે ઉપરથી નીચે સુધી ભરેલો છે, પરંતુ અહીં આપણે ઢગલોની મિલકતનું ઉલ્લંઘન કર્યું છે કારણ કે 7 ના બાળકનું મોટું કદ છે મૂલ્ય તે કરતાં વધુ છે 8.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 07:32)

અમારી પ્રથમ નોકરી ઢગલોની જાળવણી કરતી વખતે એક ઢગલામાં મૂલ્ય દાખલ કરવાનું છે. તેથી, નોંધ લેવાની પ્રથમ વાત એ છે કે નવા નોડને ક્યાં મૂકી શકાય તે અંગે અમારી પાસે કોઈ પસંદગી નથી, યાદ રાખો કે ઢગલો ગાંઠો ઉપરથી નીચે સુધી, ડાબેથી જમણે બનેલા છે જો આપણે 12 દાખલ કરવા ઇચ્છીએ છીએ તો તે 10 ની નીચે નીચે આવવું જોઈએ કારણ કે આપણે એક નવું સ્તર શરૂ કરવું પડશે, કારણ કે પાછલા સ્તર પૂર્ણ છે. સમસ્યા એ છે કે આ ઢગલો મિલકતને સંતુષ્ટ કરી શકશે નહીં; આ કિસ્સામાં 12 તેના પિતૃ કરતાં મોટી છે 10. જોકે આ હવે માળખાકીય રીતે યોગ્ય છે, તેની પાસે યોગ્ય મૂલ્ય વિતરણ નથી. તેથી, અમે અમુક રીતે ઢગલો મિલકત પુનઃસ્થાપિત કરવા માટે હોય છે. આ આપણે સૌ પ્રથમ નોડ બનાવીએ છીએ, અમે તે નોડમાં નવું મૂલ્ય મૂકે છે અને ત્યારબાદ આપણે તેના સંબંધમાં ઉલ્લંઘન જોવાનું શરૂ કરીએ છીએ તે પિતૃ છે. અમે નોંધ્યું છે કે 12 અને 10 યોગ્ય ક્રમાંકન નથી. તેથી, અમે તેમને અદલાબદલ કરીએ છીએ, હમણાં આ એક નવા નોડ છે અમારે તપાસ કરવી પડશે કે તે તેના વર્તમાન પિતૃના સંબંધમાં યોગ્ય રીતે આદેશ આપ્યો છે કે નહીં. તેથી, અમે જુઓ અને અમને લાગે છે કે તે નથી. તેથી, ફરીથી અમે આનું પરિવહન કરીએ છીએ. હવે, નોંધ લો કે 11 પહેલાથી 5 કરતા વધારે મોટી છે, 12 5 કરતાં મોટી રહેશે. જ્યાંથી અમને મળ્યું ત્યાંથી કંઈપણ તપાસ કરવાની જરૂર નથી, અમને ફક્ત જોવાની જરૂર છે હવે, આપણે તપાસવું પડશે કે ઉપરની સમસ્યા હજુ પણ છે. આ કિસ્સામાં, કોઈ સમસ્યા 12 કરતાં ઓછી છે 24. તેથી, અમે બંધ.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 08:59)

ચાલો બીજું નોડ ઉમેરીએ. ધારી રહ્યા છીએ, આપણે હમણાં 33 બનાવ્યું છે જે આપણે બનાવેલું છે. તેથી, 33 ફરીથી આ બિંદુએ એક નવો નોડ બનાવે છે. હવે, 33 11 કરતાં મોટી છે, આપણને ચાલવા અને તે સ્વેપ કરવી પડશે પછી અમે 33 અને તેનાં માતાપિતા 12 ની સરખામણી કરીએ છીએ અને આપણે નોંધ્યું છે કે 33 એ 12 કરતા મોટો છે. તેથી, આપણે તેને ફરીથી સ્વેપ કરીએ તો આપણે રૂટને જુઓ, આ કિસ્સામાં 24 અને આપણે શોધી કાઢીએ છીએ કે 33 24 કરતા મોટી છે. તેથી, આપણે તેને ફરીથી સ્વેપ કરીએ છીએ અને હવે 33 ના માતાપિતા છે અને ચોક્કસપણે તે 2 બાળકો કરતાં મોટી છે તેથી, અમે બંધ કરી શકો છો

(સ્લાઇડ સમયનો સંદર્ભ લો: 09:35)

દાખલ કરવામાં કેટલો સમય લે છે? દરેક વખતે અમે નોડ શામેલ કરીએ છીએ, તેના માતાપિતા, સ્વેપ, તેના માતાપિતા, સ્વેપ અને તેથી પર તપાસ કરાવવું પડે છે, પરંતુ સારી વસ્તુ એ છે કે આપણે કોઈ પાથ સુધી જ ચાલીએ છીએ કે અમે કોઈ પાથ નીચે ન ચાલીએ. તેથી, તમે ચાલતાં પગલાઓની સંખ્યા વૃક્ષની ઊંચાઈથી ઘેરાયેલી હશે. હવે, અમે પહેલાં દલીલો કરી હતી કે આપણે પહેલાં ઉલ્લેખ કર્યો છે કે સંતુલિત વૃક્ષ ઊંચાઈ લોગ n હશે. તેથી, આપણે વાસ્તવમાં તે કહીને યોગ્ય રીતે માપી શકીએ છીએ કે સ્તર 1 પરના ગાંઠોની સંખ્યા i થી 2 છે. શરૂઆતમાં, અમારી પાસે 0 થી 1 નોડ 2 છે, પછી પ્રથમ સ્તર પર આપણી પાસે 2 ગાંઠો 2 થી 1 અને બીજા સ્તરે અમારી પાસે 4 ગાંઠો 2 અને 2 છે. જો આપણે આમ આ રીતે કરીએ છીએ તો આપણે શોધી કાઢીએ છીએ કે જ્યારે કક્ષાનું સ્તર ભરવામાં આવે છે, ત્યારે આપણે 2 થી k ઓછા 1 ગાંઠો હશે અને તેથી, આને ફેરવવાથી આપણે શોધી શકીએ કે જો અમારી પાસે નોડ હશે તો સ્તરોની સંખ્યા લોગ n હોવી જોઈએ. તેથી, દાખલ કરો એક પાથ અપ લઈ જશે, પાથ વૃક્ષની ઊંચાઈ બરાબર છે, અને વૃક્ષની ઊંચાઈ લોગ n એ છે. તેથી, insert સમયની લોગ n એ લે છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 10:42)

આપણે ઢગલામાં અમલમાં મૂકવાની જરૂર છે તે અન્ય ઓપરેશન ડિલક્સ મહત્તમ છે. હવે, એક ઢગલો વિશે એક વાત એ છે કે મહત્તમ મૂલ્ય રૂટ પર હંમેશા હોય છે કારણ કે ઢગલોની મિલકતને કારણે તમે તેને જોઈ શકો છો કારણ કે દરેક નોડ તેના બાળકો કરતા મોટા છે, સમગ્ર વૃક્ષનું મહત્તમ મૂલ્ય રૂટ પર હોવું જોઈએ. તેથી, આપણે જાણીએ છીએ કે રૂટ ક્યાં છે; હવે પ્રશ્ન એ છે કે આપણે કેવી રીતે તેને અસરકારક રીતે દૂર કરીએ?

(સ્લાઇડ સમયનો સંદર્ભ લો: 11:10)

જો આપણે આ નોડને દૂર કરીએ તો, સૌ પ્રથમ આપણે નોડને દૂર કરી શકતા નથી કારણ કે તે મૂળ છે. જો તમે આ મૂલ્ય દૂર કરો તો આપણે ત્યાં અમુક મૂલ્ય મુકવી પડશે. બીજી તરફ, ઢગલામાં નોડના મૂલ્યોની સંખ્યા હવે સંકોચાઈ છે. તેથી, તળિયે જમણે આ નોડ કાઢી નાંખવો જોઈએ કારણ કે ઢગલાના માળખાકીય મિલકત કહે છે કે આપણે વૃક્ષને ડાબેથી જમણે, ઉપરથી નીચે સુધી ભરીએ. અમે ઉપરથી નીચે જઈ રહ્યા છીએ અને અમારી પાસે મૂલ્યની બહાર છે. અમે ઉમેર્યું છે કે છેલ્લા નોડ નીચે પંક્તિની જમણી સૌથી ઓવરને અંતે એક હતું અને તે જવું આવશ્યક છે. તેથી, અમારી પાસે એક મૂલ્ય છે જે ટોચ પર ખૂટે છે અને અમારી પાસે તળિયે મૂલ્ય છે, જે 11 નું નોડ કાઢી નાખવામાં આવશે. તેથી, આ મૂલ્ય હવે આ મૂલ્યને 11 થી ખસેડવાનું છે અને પછી વસ્તુઓને ઠીક કરવા, અધિકાર. તેથી, આપણે પહેલા રૂટમાંથી 33 દૂર કરીએ, આપણે નોડને 11 થી દૂર કરીએ છીએ અને આપણે 11 ને રૂટની પદ પર ખસેડીએ છીએ. હવે, આની સમસ્યા એ છે કે આપણે એક મૂલ્યવાન મૂલ્ય ખસેડ્યું છે જે ટોચ પર મહત્તમ

મૂલ્ય નથી. દેખીતી રીતે, તેના બાળકોના સંદર્ભમાં કોઈ સમસ્યા હશે. તેથી, અહીં તે તારણ આપે છે કે 11 7 કરતા મોટો છે, જે સાચું છે, પરંતુ દુર્ભાગ્યે તે 24 કરતા ઓછું છે.

(સ્લાઈડ ટાઇ નો ઉલ્લેખ કરો: 12:19)

આપણે શું કરવું તે સંપત્તિ પુનઃસ્થાપિત કરવા માટે છે, અમે બંને દિશાઓ જમણી બાજુએ જુઓ અને અમે તે સૌથી મોટું બાળક સાથેનું વિનિમય કરો ધારીએ કે, અહીં 17 હતા, તો આપણે અહીં 17 સાથે 11 અથવા 11 સાથે 24 ફેરબદલ કરી શકીએ છીએ, બન્ને ઉલ્લંઘન છે, પરંતુ જો તમે આ નાનું બાળક ખસેડો તો 17 17 કરતાં મોટી નહીં, તેથી અમે સૌથી મોટું એક ખસેડીએ છીએ. આ કિસ્સામાં અમે 24 ઉપર જમણી બાજુએ ખસેડીએ છીએ અને હવે આપણી પાસે 11 ફરી છે અને હવે ફરી તપાસ કરવી પડશે કે શું તે તેના 2 બાળકોને સંબંધિત છે કે નહીં, તે ફરીથી નથી. તેથી, આપણે સૌથી મોટા 12 અપ ખસેડીએ છીએ અને તે પછી હવે આપણે જોઈ શકીએ છીએ કે તે તેના બાળકોના સંદર્ભમાં સાચું છે કે કેમ. આ બિંદુએ 11 10 કરતા મોટો છે. તેથી, અમે બંધ કરીએ છીએ.

(સ્લાઈડ ટાઇમ નો ઉલ્લેખ કરો: 13:03)

જેમ જેમ નળને રુટ સુધી પહોં પર નવો નોડમાંથી એક રસ્તાની દિશામાં આવે છે, એટલું જ નહીં, રુટમાંથી નીચે એક પાંદડાની તરફ એક પાથને અનુસરવાની મહત્તમ ડીલ કરો. એકવાર ફરીથી કાઢી નાખોનો ખર્ચ વૃક્ષની ઊંચાઈને પ્રમાણસર હશે જે આપણે અગાઉ જણાવ્યું હતું કે લોગ એન છે. ચાલો આપણે બીજા કાઢી નાખો; આપણે કાઢી નાખીએ, આ કિસ્સામાં 24, હવે આપણે 10 માટે નોડ દૂર કરીએ, 10 રુટ સુધી જાય છે. અમે તેના 2 બાળકો, 12 અને 7 ની સરખામણીએ 10 ની તુલના કરીએ છીએ અને શોધી કાઢીએ છીએ કે તે ઢગલો મિલકતને સંતોષજનક નથી. તેથી, આપણે મોટાભાગના મોટાભાગના 12 ને આગળ વધારીએ છીએ. હવે, આપણે તેનાં બાળકો, નવા બાળકો અહીં 11 અને 5 જુઓ અને ફરીથી અમે જોઈ શકીએ છીએ કે તે મિલકતને સંતોષજનક નથી. તેથી, મોટા એક ઉપર જાય છે અને એકવાર તે પાંદડા સુધી પહોંચે છે અને ત્યાં હવે સંતોષવા માટે કોઈ ગુણધર્મો નથી. તેથી, અમે બંધ કરીએ છીએ.

(સ્લાઈડ ટાઇમ નો ઉલ્લેખ કરો: 13:56)

ઢગલોની એક ખૂબ જ આકર્ષક લાક્ષણિકતા એ છે કે આપણે આ વૃક્ષને સીધી યાદીમાં અથવા એરેમાં અમલમાં મૂકી શકીએ છીએ. તેથી, આપણી પાસે n નોડ ઢગલો છે, આપણે તેને 0 થી n ની સ્થિતિ સાથે સૂચિ અથવા એરે તરીકે રજૂ કરી શકીએ છીએ. સ્થિતિ 0 રુટનું પ્રતિનિધિત્વ કરે છે, પછી ક્રમમાં 1 અને 2 બાળકોને રજૂ કરે છે, પછી 3, 4, 5, 6, 7 ગાંઠો આગામી સ્તર છે અને તેથી. તેથી, જેમ આપણે કહ્યું હતું કે અમે આ ઢગલોને ડાબેથી જમણે, ઉપરથી નીચે જમણે ભરી દીધું છે. એ જ રીતે, આપણે નોડ્સને ઉપરથી નીચે સુધી, ડાબેથી જમણા ક્રમાંકિત કરીએ છીએ. તેથી, આપણે રુટ પર 0 થી શરૂ કરીએ, પછી ડાબી બાજુએ 1, જમણી બાજુએ 3, 4, 5, 6, 7, 8, 9, 10 અને તેથી વધુ. આમાંથી તમે જોઈ શકો છો કે, જો મારી પાસે લેબલ હોવાની સ્થિતિમાં હોય તો પછી બે બાળકો વાંચે છે 2 આઈ પ્લસ 1 અને 2 આઈ પ્લસ 2 જમણી. તેથી, 1 ના બાળકો 2 વત્તા 1 વડે 1, જે 3 અને 2 માં 1 વત્તા 2 છે, જે 4 છે. એ જ રીતે, 5 ના બાળકો 2 થી 5 વત્તા 1, જે 11 અને 2 માં 5 વત્તા 2 છે. 12. તેથી, ઢગલામાં પદ માટે ઇન્ડેક્સની ગણતરી કરીને આપણે તે શોધી શકીએ છીએ કે તે બાળકો ક્યાં છે અને આ ગણતરીને પાછો ખેંચી લઈને આપણે માતાપિતાના ઇન્ડેક્સ શોધી શકીએ છીએ, જેનાં માતાપિતા એ છે કે j ઓછા 1 બાય 2. હવે, જો -1 બાદ 2 બરાબર પૂર્ણાંક ન હોઈ શકે. તેથી, અમે ફ્લોર લઈએ છીએ. ઉદાહરણ તરીકે, જો આપણે 11 લે છે, 11 ઓછા 1 10, 10 ની 2 ઘાત 5 છે. જો આપણે 14 લે, ઉદાહરણ તરીકે, 14 ઓછા 1 13, 13 બરાબર 6.5, તો આપણે ફ્લોર લઈએ છીએ અને 6 મળે છે. સ્લાઈડ ટાઇમ: 15:32)

આ આપણને ઇન્ડેક્સ અંકગણિત કરવાથી માતાપિતા અને બાળકોના ગાંઠોને ચાલાકી આપવાની મંજૂરી આપે છે, હું 2 થી વત્તા 1, 2 આઇ પ્લસ 2 ને બાળકોમાં જવા માટે જઇએ છીએ અને આપણે j પિતૃ પર જવા માટે 2 ફ્લોર અમે એક ઢગલો કેવી રીતે બનાવી શકું ઢગલો બનાવવાનો એક નિષ્ક્રમ માર્ગ એ ફક્ત મૂલ્યોની સૂચિ લેવાનું છે અને ઢગલાબંધ ઢગલાને ઢગલામાં ઉપયોગ કરીને તેને એક પછી એક દાખલ કરો. તેથી, અમે ખાલી ઢગલાથી શરૂ કરીએ છીએ, અમે x 1 દાખલ કરીએ છીએ, એક્સ 1 ધરાવતી એક નવી ઢગલો બનાવીએ છીએ, અમે એક્સ 2 દાખલ કરીએ છીએ, એક્સ 1, એક્સ 2 અને તેથી વધુનું ઢગલો બનાવીએ છીએ. દરેક ઓપરેશન અલબત્ત લોગ એન ટાઇમ લે છે, n વધતી જશે, પરંતુ જો આપણે અંતિમ એન ને ઉપલા માધ્યમ તરીકે લેતા હોઈએ તો તે માત્ર ત્યારે જ લોગ n કરે અને અમે આ હપને ક્રમમાં n લોગ એન ટાઇમમાં બનાવી શકીએ.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 16:23)

જો આ એરેને x 1 થી x_n તરીકે હોય તો આ ઢગલો બિલ્ડીંગ કરવા માટે વધુ સારી રીત છે, નોડના છેલ્લા અડધા વૃક્ષના પાંદડા સાથે સંબંધિત છે. હવે, એક પાંદડાની નોડને સંતોષવા માટે કોઈ ગુણધર્મો નથી કારણ કે તેની પાસે કોઈ બાળકો નથી. આપણે જે કાંઈ પણ પાંદડા છોડી દઈએ છીએ તે કાંઈ પણ કરવાની જરૂર નથી. અમે ઉપર એક સ્તર ઉપર જઈએ છીએ અને ત્યારબાદ આપણે એક જ સ્તર ઉપર તમામ હીપ ભૂલોને ઠીક કરી શકીએ છીએ અને પછી ફરી ઉપર એક સ્તર ખસેડીએ છીએ. તેથી, અમે ઢગલાબંધ ફિક્સિંગની ટોચની પ્રકારની છે કે જે અમે કાઢી નાંખો સાથે કર્યું છે, જ્યારે અમે ઢગલો બનાવી રહ્યા છીએ. તેથી, જેમ આપણે પગલાંની સંખ્યા વધારીએ છીએ જે આપણને આ પ્રચાર કરવાની જરૂર છે તે વધારે અને ઊંચી જાય છે કારણ કે બીજી તરફ ઉચ્ચ બિંદુથી શરૂ કરવાની જરૂર છે, જેના માટે નોડ્સની સંખ્યા ઓછી થાય છે, જેના માટે આ બને છે તે નાની છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 17:19)

ચાલો આને અહીં જોઈએ. આપણે જે કહીએ છીએ તે એ છે કે જો આપણે 0 થી 14 તત્વોની મૂળ સૂચિ સાથે શરૂઆત કરીએ, તો 7 થી 14 નંબરો પહેલાથી જ ઢગલોની મિલકતને સંતોષશે. ગમે તે મૂલ્યો છે, આપણે ચિંતા કરવાની જરૂર નથી, પછી આપણે આગળ વધીએ છીએ અને તેના બાળકો સાથે આને સ્વેપ કરવું પડી શકે છે, આપણે તેના બાળકો સાથે આને સ્વેપ કરવું પડશે અને આ રીતે. 4 ગાંઠો માટે આપણે હીપ મિલકતને સુધારવા માટે કદાચ એક સ્તરનું સ્થળાંતર કરવું પડશે, તો આપણે હવે 1 અને 2 ની મૂળ કિંમત છે, તે ખોટી હોઈ શકે છે. તેથી, ફરી આપણે તેને એક મૂલ્ય નીચે પાળી શકીએ છીએ અને પછી બીજા મૂલ્ય. હવે, આપણને બે સ્તરનું સ્થળાંતર કરવાની જરૂર છે, પરંતુ અમારી પાસે આ માટે માત્ર બે નોડ છે. ગાંઠોની સંખ્યા કે જેના માટે આ જરૂરી છે તે સંકોચાઈ રહ્યું છે, તે વાસ્તવમાં અડધી છે અને જેના માટે આપણે તે કરવું છે તે સંખ્યાઓ 1 વધે છે. અમે અહીં સાવચેત ગણતરી નહીં કરીએ, પરંતુ તે તારણ આપે છે કે તેના પરિણામે આ, ઢગલોના તળિયેથી શરૂ કરીને અને ઉપરની જગ્યાએ ખાલી ઢગલામાં એકને દાખલ કરવાને બદલે હાથપાઈપ કરવાની આ ચોક્કસ રીત ખરેખર અમને ફક્ત રેખીય સમયનો ક્રમ n કરે છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 18:30)

હંપનો અંતિમ ઉપયોગ ખરેખર સોર્ટ કરવા માટે છે, અમે મહત્તમ એકથી શરૂ થતાં એક સમયે એક ઘટક લઈ રહ્યા છીએ. તે સ્વાભાવિક છે કે જો આપણે સૂચિથી શરૂ કરીએ, ઢગલો બનાવીએ અને પછી મહત્તમ વખત ડિલિટ કરીએ તો આપણે ઉતરતા ક્રમમાં મૂલ્યોની સૂચિ મેળવીશું. અમે ક્રમમાં n સમય એક ઢગલો બિલ્ડ, મહત્તમ n વખત કાઢી કોલ અને ઉતરતા ક્રમમાં તત્વો કાઢવા. તેથી, આપણને હીપ માટે ઓર્ડર એન લોગ એન એલ્ગોરિધમ મળે છે. હવે, પ્રશ્ન એ છે કે આપણે આ મૂલ્યો ક્યાં રાખીએ. વેલ, યાદ રાખો કે ઢગલો એરે છે. શરૂઆતમાં, અમારી પાસે એક હીપ છે જે n તત્વો ધરાવે છે. તેથી, આપણે આ

ઢગલાને બનાવીએ હવે આપણે કહ્યું છે કે ડિલ મેક્સ એ ટોચ પર તત્વને દૂર કરશે, કારણ કે તે મૂળ છે, પણ અહીં ખાલી ખાલી જગ્યા પણ બનાવશે, તે આ મૂલ્ય છે કે જે ટોચ પર જશે, આ છેલ્લું છે પણ કે જે ટોચ પર જશે જ્યારે અમે કાઢી નાંખો મહત્તમ સુધારવા. અહીં એક ખાલી જગ્યા હોવાથી અમે ખરેખર આ સ્થાને ખસેડી શકીએ છીએ, મહત્તમ મૂલ્ય હવે ઢગલોના અંતમાં જશે, પરંતુ જ્યારે આપણે આગળ ઢગલો પર પ્રક્રિયા કરીશું ત્યારે માત્ર n બાદ એક જ મૂલ્યો હશે. તેથી, આપણે તે મૂલ્યને જોશું નહીં, આપણે ફક્ત 0 થી n ની 2 ઘાત 2 માં જ કરીશું. ફરીથી આ મૂલ્ય અહીં આવે છે અને આ મૂલ્ય ત્યાં જશે અને હવે આપણી પાસે બે ઘટકો નિશ્ચિત હશે. તેથી, એક પછી એક મહત્તમ મૂલ્ય, બીજો મહત્તમ મૂલ્ય અને તે જ સૂચિ અથવા એરેમાં આવશે જેમાં અમે ઢગલાને સંગ્રહિત કરી રહ્યા છીએ અને આખરે ઢગલો યડતા ક્રમમાં બહાર આવશે. આ વાસ્તવમાં એન લોગ એન સોર્ટ છે તે અસંખ્ય જટિલતા જેવું જ છે જે મર્જ સોર્ટ પહેલાં અમે જોયું અને મર્જ સોર્ટથી વિપરીત જે અમને બે યાદીઓને મર્જ કરતી વખતે નવી એરે બનાવવા માટે ફરજ પાડતી હતી, આ વાસ્તવમાં સ્થાને એક સૂચિ બનાવી રહી છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 20:20)

હેપનો સારાંશ આપવા માટે અગ્રતા ક્યુને એક વૃક્ષ આધારીત અમલીકરણ છે જેમાં પ્રવેશ અને કાઢી નાંખવાનું બંને લોગ એન સમયમાં કરી શકાય છે. અમે ક્રમમાં nap સમય માટે ઢગલો બિલ્ડ કરવા માટે નીચે અપ heapify કરી શકો છો અને આ વૃક્ષો છે, પરંતુ તેઓ એક એરે મદદથી ખૂબ જ સરળતાથી ચાલાકીથી કરી શકાય છે. હવે, આ કિસ્સામાં અમે મહત્તમ ઢગલાઓ જોઈ રહ્યા હતા; અમે ડુઅલ બાંધકામ પણ કરી શકીએ છીએ જ્યાં આપણે ઢગલોની સ્થિતિને બદલીએ છીએ તે કહેવું છે કે દરેક તત્વ તેના બાળકો કરતાં નાના હોવું જોઈએ, આ કિસ્સામાં આપણે શું કરવું જોઈએ તે પછી મિનિ-હેપ કહેવામાં આવે છે અને પછી કાઢી નાંખોને બદલે, આપણી કામગીરી અમે કરીએ છીએ મિનિટ અન્ય કિસ્સામાં બધું બરાબર સપ્રમાણ છે. જ્યારે આપણે કોઈ વસ્તુને ખસેડીએ છીએ ત્યારે આપણે તેને મિની શરત પ્રમાણે ખસેડવાનું હોય છે અને જ્યારે આપણે તળિયે કંઈક દાખલ કરીએ છીએ ત્યારે આપણે તેને ઉપર ખસેડવાનું છે તેથી, દાખલ કરો અને કાઢી નાંખો બરાબર કામ કરે છે અને મહત્તમ કાઢી નાંખવું બરાબર છે, સિવાય કે સરખામણીઓ ઉલટાવી શકાય છે કારણ કે હવે અમે સ્થાનિક સ્તરે મહત્તમ સ્થિતિને બદલે મિની સ્થિતિ ધરાવીએ છીએ અને છેલ્લે, આપણે જોયું કે આપણે ઢગલો સાથે ક્રમમાં ગોઠવણી કરી શકીએ છીએ. જગ્યાએ લોગ સમય

પ્રોગ્રામિંગ ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રો. માધવન મુકુંદ,
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટીટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 07 લેકચર - 01 એબ્સ્ટ્રેક્ટ ડેટેટાઇપ્સ, વર્ગો અને ઓબ્જેક્ટ્સ

(સ્લાઇડ ટાઇ નો ઉલ્લેખ કરો: 00:02)

ડેટા સ્ટ્રક્ચરો જેમ કે, સ્ટેક્સ, ક્યુઓ અને હેપ્સ જેનો ઉપયોગ પાયથોનની યાદીના પ્રકારથી થાય છે. તે તારણ આપે છે કે કોઈ પાયથોન માં બિલ્ટ ઇન પ્રકારોથી આગળ જઈ શકે છે અને આપણા પોતાના ડેટા પ્રકારો બનાવી શકે છે. તેથી, અમે આ અઠવાડિયાના વ્યાખ્યાનમાં વધુ વિગતવાર જોઈશું. ચાલો જોઈએ કે આપણે ડેટા માળખું દ્વારા શું દુર્બળ છીએ. ડેટા માળખું મૂળભૂત રીતે માહિતીનું સંગ્રહ છે જેની વર્તણૂક ઇન્ટરફેસ દ્વારા વ્યાખ્યાયિત થયેલ છે. તેથી એક ઇન્ટરફેસ ઓપરેશન્સની મંજૂર સેટ સિવાય કશું જ નથી, પરંતુ સ્ટેક માટેની સંજોગોમાં મંજૂરીના સેટને પુશ અને પોપ છે. અને અલબત્ત, આપણે પણ જાણી શકીએ છીએ કે સ્ટેક ખાલી છે કે નહીં. તેવી જ રીતે, એક કતાર માટે આપણે કતારમાં ફેરફાર કરી શકીએ તે એકમાત્ર રસ્તો છે કતારની પૂંછડીને કંઈક એડ કરવાથી u ઉમેરવું અને ક્યુનું માથું તત્વને દૂર કરવાથી કાર્યને v દૂર કરો. અને ઉદાહરણ તરીકે મહત્તમ ઢગલો માટે, એક ઘટક ઉમેરવા અને ઢગલામાંથી સૌથી મોટું તત્વને દૂર કરે છે તેમાંથી એક ભાગ ઉમેરવા માટે કાર્યો શામેલ છે. હમણાં, માત્ર કારણ કે અમે એક યાદી તરીકે ઢગલો અમલ તે અર્થ એ નથી કે કાર્યો યાદી માટે વ્યાખ્યાયિત કરવામાં આવે છે ખરેખર ઢગલો માટે કાનૂની છે. તેથી જો અમારી પાસે એક ઢગલો છે, જે પાયથોન ની પધ્ધતિ તરીકે અમલમાં મુકવામાં આવે છે, જોકે આ સૂચિ એપેન્ડ ફંક્શનને મંજૂરી આપશે. તેના પર એન્ડડેન ફંક્શન પોતાનામાં મૂલ્ય શામેલ કરતું નથી અને હીપની મિલકતને જાળવી રાખે છે. તેથી, સામાન્ય રીતે કોલ, જેમ કે એચ ડોટ એડે 7, કાનૂની નહીં હોય.

(સ્લાઇડ સમયનો સંદર્ભ લો: 01:35)

તેથી, અમે મંજૂરી આપેલ કામગીરીની દ્રષ્ટિએ નવા અમૂર્ત ડેટા પ્રકારોને વ્યાખ્યાયિત કરવા માંગીએ છીએ. અમે અમલીકરણને જોવા નથી માગતા કે તે એક સૂચિ છે કે નહીં, કારણ કે અમે અમલીકરણને તે નક્કી કરવા માગતા નથી કે કઈ મંજૂરી છે, અમે ફક્ત વાસ્તવિક ઓપરેશન્સને જ જોઈએ કે જે અમે અમૂર્ત ઇન્ટરફેસને પરવાનગી આપીએ છીએ. દાખલા તરીકે, જો અમારી પાસે સ્ટેક હોય અને અમે મૂલ્ય વી કરીએ તો સ્ટેકની મિલકત ગેરંટી આપે છે કે જો અમે તરત જ પોપ લાગુ પાડીએ છીએ તો અમે જે પાછી મેળવીએ છીએ તે અમારો છેલ્લો ભાવ દબાણ છે અને તેથી આપણે પાછા મેળવીએ. વી. અન્ય શબ્દોમાં, જો આપણે આ ક્રમાંકને અમલમાં મુકીએ છીએ, આપણે સૌ પ્રથમ પોટ કરીએ છીએ અને ત્યારબાદ આપણે પોપ કરીએ છીએ અને તે મૂલ્ય કે જે આપણે દબાણ કર્યું તે મૂલ્ય હોવું જોઈએ જે આપણે પાછા મેળવીએ. આ સ્ટેકની મિલકતને અમૂર્ત રીતે વ્યાખ્યાયિત કરવાની રીત છે અને કેવી રીતે આંતરિક મૂલ્યોનું પ્રતિનિધિત્વ કરે છે તે વિશે અમને કંઈક કહેવા વગર દબાણ અને પોપ કેવી રીતે ક્રિયાપ્રતિક્રિયા કરે છે. એ જ રીતે જો અમારી પાસે ખાલી કતાર છે અને અમે તેને બે ઘટકો u અને v માં ઉમેરીએ છીએ અને પછી અમે કતારના વડાને દૂર કરીએ છીએ, પછી અમે અપેક્ષા રાખીએ છીએ કે અમે ખાલી કતારથી શરૂઆત કરી અને પછી અમે આ અંતથી મૂકીએ અને તો પછી આપણે AV માં મુકીએ છીએ, પછી જે તત્વ બહાર આવે છે તે પ્રથમ ઘટક એટલે કે u હોવું જોઈએ. બીજા શબ્દોમાં કહીએ તો આ ખાલી છે, જો આપણે યુ ઉમેર્યા અને v ઉમેર્યા અને પછી માથાને દૂર કરીએ તો આપણે પાછા પ્રથમ તત્વ મેળવવું જોઈએ જે અમે એટલે કે u માં મુકતા હતા. અગત્યની બાબત એ છે કે અમે અમલીકરણના સંદર્ભ વગર ડેટા પ્રકારનાં વર્તનને વ્યાખ્યાયિત કરવા માંગીએ છીએ. હવે આ ખૂબ જ કંટાળાજનક હોઈ

શકે છે કારણ કે તમારે ખાતરી કરવી પડશે કે તે વિધેયો વચ્ચેની બધી સંપત્તિઓ કેપ્ચર કરે છે, પરંતુ આ કરી શકાય છે અને તે ટેકનિકલી છે કે કેવી રીતે એક અમૂર્ત ડેટા પ્રકાર વ્યાખ્યાયિત થયેલ છે. હવે મોટા હેતુઓ અમે સામાન્ય રીતે તે વધુ અનૌપચારિક રીતે વ્યાખ્યાયિત કરીશું અને અમે અમલીકરણનો સંદર્ભ આપશું, પરંતુ અમે ચોક્કસપણે આ કાર્યને કેવી રીતે કાર્ય કરવું તે નક્કી કરવા અમલીકરણને નથી માગતો. બીજા શબ્દોમાં કહીએ તો, આપણે એક અમલીકરણને બીજા કોઈમાં બદલી શકીએ છીએ જેમ કે વિધેયો એ જ રીતે વર્તે છે અને બહારના યુઝર્સને કોઈ ખ્યાલ નથી કે જે અમલીકરણનો ઉપયોગ કરવામાં આવે છે. હવે આ મોટેભાગે જ્યારે આપણે અમલીકરણની ઓપ્ટિમાઇઝ કરવાની જરૂર છે, ત્યારે અમે બિનકાર્યક્ષમ અમલીકરણ સાથે આવી શકીએ છીએ અને પછી તેને ઓપ્ટિમાઇઝ કરી શકીએ છીએ. દાખલા તરીકે, આપણે જોયું કે પ્રાધાન્ય કતાર માટે આપણે તેને એક સોર્ટ કરેલી સૂચિ તરીકે અમલમાં મૂકી શકીએ છીએ અને પછી અમે એક સોર્ટ કરેલ સૂચિમાં દાખલ કરેલ ઓપરેશન તરીકે ઇન્સ્ટોલ કરી શકીએ છીએ, જે તમે ઓર્ડર n નો સમય લે છે, પરંતુ મહત્તમ કાઢી નાંખો, ફક્ત સૂચિનાં માથાને દૂર કરશે. આ શ્રેષ્ઠ નથી કારણ કે n ઇન્વેર્સર્સના ક્રમ પર અને કાઢી નાંખે છે, આને ટાઇમ ક્રમમાં એન ચોરસ લે છે. તેથી જો આપણે સોર્ટ કરેલી સૂચિમાંથી આંતરિક અમલીકરણને એક ઢગલોથી બદલીએ તો અમે વધુ સારી રીતે વર્તન કરીએ છીએ, પરંતુ વાસ્તવિક મૂલ્યોની દ્રષ્ટિએ કે જે વપરાશકર્તાને દાખલ કરેલા ક્રમ તરીકે જુએ છે અને વધુમાં વધુ વપરાશકર્તાને કાઢી નાંખે છે તે સોર્ટ કરેલ સૂચિ અમલીકરણ વચ્ચે કોઈ તફાવત નથી. અને ઢગલો અમલીકરણ. કદાચ, એવી ધારણા છે કે એક અન્ય કરતા વધુ ઝડપી છે, પરંતુ અમલીકરણની વાસ્તવિક શુદ્ધતાને તમે કેવી રીતે માહિતીનું પ્રતિનિધિત્વ કરવાનું પસંદ કરશો તેનાથી અસર થવી જોઈએ નહીં. તેથી, આ એબ્સ્ટ્રેક્ટ ડેટાટાઇપને વ્યાખ્યાયિત કરવાનું સાર છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 04:40)

તેથી, અમૂર્ત ડેટાટાઇપ વિશે વિચારવાનો એક સારો માર્ગ બ્લોક બોક્સ છે જે મર્યાદિત ક્રિયાપ્રતિક્રિયાને મંજૂરી આપે છે. એટીએમ મશીનની જેમ કંઈક કલ્પના કરો. તેથી, આપણી પાસે કાળા બોક્સ તરીકે ડેટા સ્ટ્રક્ચર છે અને આપણી પાસે કેટલાક બટન્સ છે જે જાહેર ઇન્ટરફેસ છે, આ એ કાર્ય છે જેનો ઉપયોગ અમને કરવાની મંજૂરી છે. આ ચિત્રમાં કલ્પના કરો કે આ એક સ્ટેક છે અને બટનોને પોપ થવાની મંજૂરી આપવામાં આવી હતી અને તેમને સ્ટેકમાંથી ટોચની ઘટકોને દૂર કરવાની મંજૂરી આપવામાં આવી હોવાનું દબાણ કરવામાં આવ્યું હતું; તેઓ સ્ટેકમાં એક ઘટક મૂકવાની મંજૂરી આપે છે. હવે આને સ્ટેકથી વસ્તુઓ ઉમેરવાની અને જોવાની જરૂર છે, તેથી આપણી પાસે પણ ઇનપુટ માટે સ્લોટ છે જે નીચે એક પ્રકારની વસ્તુ તરીકે બતાવવામાં આવે છે અહીં આપણી પાસે ઇનપુટ માટે સ્લોટ છે. અને સ્ટેકની સ્થિતિ વિશેની માહિતી પ્રાપ્ત કરવાની રીત છે. તેથી અમે કલ્પના કરી શકીએ છીએ કે અમારી પાસે કોઈ પ્રકારની પ્રદર્શન છે. આ સામાન્ય રીતે આપણે કેવી રીતે ડેટા માળખું વિશે વિચારવું ગમશે, આપણે કાળા બોક્સની અંદર શું છે તે જાણવા માંગતા નથી, આપણે ફક્ત તે સ્પષ્ટ કરવા માંગીએ છીએ કે જો આપણે બટનને અનુસરવું હોય અને અમે ઇનપુટ બોક્સ દ્વારા ઇનપુટ પૂરો પાડવાનું શરૂ કરીએ તો શું થશે શું આપણે એ જોવાની આશા રાખીએ છીએ કે પ્રદર્શન શું છે. આ સિવાય, અન્ય કોઈ હેરફેરને મંજૂરી આપવી જોઈએ નહીં. સ્ટેકના મધ્યમાં અથવા કતારમાં મધ્યમ સુધી પહોંચવા માટે અમને ઝડપથી એક્સેસ મેળવવા માટે બોક્સની અંદર શું છે તેનો ઉપયોગ કરવાની મંજૂરી નથી. તેથી અમે આ પ્રકારના ઓપરેશનોને ઈચ્છતા નથી, અમે ફક્ત તે ઓપરેશન્સ જોઈએ કે જે બ્લોક બોક્સ પિકચરના આ કિસ્સામાં બાહ્ય દૃશ્યમાન ઇન્ટરફેસ અથવા બટનો અમને ઉપયોગ કરવા દે છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 06:21)

એક અર્થમાં આ પહેલેથી જ લાગુ પાડવામાં આવે છે જ્યારે આપણે પાયથોનના ડેટા પ્રકારોનો ઉપયોગ કરીએ છીએ, જો આપણે જાહેરાત કરીએ છીએ કે name | ખાલી સૂચિમાં | ને સુયોજિત કરીને પ્રકાર સૂચિ છે તો તરત જ પાયથોન અમને કાર્યસૂચિનો ઉપયોગ કરવા અને આ સૂચિ પર વિસ્તૃત કરવા માટે, પરંતુ તે સૂચિના પ્રકાર અને શબ્દકોશ પ્રકારનું નહીં હોવાથી અમે કોઈ ક્રિયાઓ ચલાવવા માટે સમર્થ

નથી, જેમ કે કીઓ કે જે શબ્દકોશો માટે વ્યાખ્યાયિત છે તે સૂચિ નથી. તેવી જ રીતે જો આપણે ખાલી શબ્દ શબ્દકોશ બનવા માટે ડી વ્યાખ્યાયિત કરીએ તો આપણે વર્તમાનમાં સંગ્રહિત કિંમતોની સૂચિ મેળવવા માટે ફંક્શનનો ઉપયોગ કરી શકીએ છીએ, પરંતુ આપણે ડી તરીકે યાદીમાં ફેરફાર કરી શકતા નથી. તેથી, આપણે ડી ડોટ ઉમેરી શકીએ નહીં તે આપણને ભૂલ આપશે પાચથોન એ પ્રકારની માહિતીનો ઉપયોગ કરે છે કે જેની પાસે કિંમત છે તે નક્કી કરવા માટે કોઈ નામ આપીએ કે કયા ફંક્શન કાયદેસર છે, જે આ અમૂર્ત ડેટા પ્રકારો સાથે અમે જે કરવાના પ્રયાસ કરી રહ્યા છીએ તે બરાબર છે. અમે એમ કહીએ છીએ કે તેના પરનો ડેટા પ્રકાર પોતાના છે, ફક્ત અમુક મર્યાદિત પ્રકારનાં એક્સેસની મંજૂરી આપવી જોઈએ, જેની વર્તણૂક અમને આંતરિક અમલીકરણ વિશે કંઈ જણાવ્યા વિના સ્પષ્ટ કરે છે. ઉદાહરણ તરીકે યાદ રાખો કે આપણે જોયું છે કે જો કોઈ ચોક્કસ ક્રમાંકમાં ક્રમ અથવા મૂલ્યો ઉમેરતા હોય તો પણ આપણે તે મૂલ્યો પૂછી શકીએ છીએ, તે પછી કોઈ જ સમયે તે લખી શકશે નહીં, કારણ કે આંતરિક રીતે કેટલાક ઓપ્ટિમાઇઝેશન છે તે માટે કિંમત જોવા માટે તેને ઝડપી બનાવો. અમે અંદર કેવી રીતે અમલમાં મુકાયેલ છે તે ખરેખર કોઈ વિચાર નથી, પરંતુ આપણે શું જાણીએ છીએ તે છે કે જો આપણે કી પૂરી પાડીએ છીએ અને તે કી એક માન્ય કી છે, તો અમે તે કી સાથે સંકળાયેલા છીએ, અમે એ પૂછતા નથી કે આ કેવી રીતે થાય છે અને આપણે પછીથી પાચથોન ના એક સંસ્કરણથી તે કેવી રીતે ફેરફારો અમલમાં મુકાય છે તે આગળ વધો. અમારા પ્રશ્ન એ છે કે, સ્ટેક્સ, ક્યુને અને હેપ્સ અને અન્ય ડેટા માળખાં માટે બિલ્ટ ઇન સૂચિનો ઉપયોગ કરવાને બદલે, અમે એક પ્રકારનો પ્રકાર વ્યાખ્યાયિત કરી શકીએ છીએ જેમાં ચોક્કસ ઓપરેશન્સને અમે જે પ્રકારથી શરૂ કરીએ છીએ તેના આધારે પરવાનગી છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 08:17)

આ મુખ્ય વસ્તુઓ છે જે ઓબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગ નામના પ્રોગ્રામિંગની શૈલી સાથે સંકળાયેલા છે. ઓબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામમાં, અમે નવા ડેટા પ્રકારો માટે ડેટા પ્રકાર વ્યાખ્યાઓ પૂરી પાડી શકીએ છીએ, જેમાં અમે સાર્વજનિક રીતે વાત કરીએ છીએ કે અમે જાહેર ઇન્ટરફેસનું વર્ણન કરીએ છીએ, તે ઓપરેશન્સ છે જે ડેટા પર માન્ય છે અને અલગથી અમે અમલીકરણ પૂરું પાડીએ છીએ ખાનગી હોવ, પછી અમે ચર્ચા કરીશું કે પાચથોન માં ભાષાની પ્રકૃતિને કારણે અમારી પાસે ગોપનીયતા સંપૂર્ણ કલ્પના નથી. પરંતુ આદર્શ રીતે અમલીકરણ ફક્ત ઇન્ટરફેસથી જ દેખાતું હોવું જોઈએ નહીં કે વપરાશકર્તા અમલ ડેટા સાથે ક્રિયાપ્રતિક્રિયા કરે. અલબત્ત, અમલીકરણ તે જ હોવું જોઈએ કે જે વપરાશકર્તા માટે દૃશ્યમાન છે તે કાર્ય યોગ્ય રીતે વર્તે. તેથી અહીં ઉદાહરણ માટે જો અમારી પાસે એક ઢગલો છે તો જાહેર ઇન્ટરફેસ સામેલ અને મહત્તમ કાઢી નાખશે, ખાનગી અમલીકરણ એક સોર્ટ કરેલ સૂચિ હોઈ શકે છે અથવા તે ઢગલો હોઈ શકે છે અને તે પછી અમે તે સુનિશ્ચિત કરવું પડશે કે જો આપણે સોર્ટ કરેલ સૂચિનો ઉપયોગ કરીએ છીએ તો અમે મહત્તમ કાઢી નાંખો અને યોગ્ય રીતે શામેલ કરો અને જો આપણે તેમાંથી એક ઢગલો પર ખસેડો તો પ્રાધાન્યતા કતારની કામગીરી એ જ રહેશે.

(સ્લાઇડ સમય જણાવો: 09:33)

ઓબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગની પરિભાષામાં બે મહત્વપૂર્ણ ખ્યાલો છે; વર્ગો અને ઓબ્જેક્ટ્સ. એક ક્લાસ એ ફંક્શન ડિક્લેરેશનની જેમ નમૂનો છે, જ્યારે આપણે ડિક્લેરેટ કહીએ છીએ અને ફંક્શન વ્યાખ્યાયિત કરીએ તો ફંક્શનને એક્ઝેક્યુટ કરતું નથી, તે અમને ફક્ત એક વાદળી છાપ આપે છે જે કહે છે કે આ શું થાય છે જો આ ફંક્શનને ખાસ રીતે કહેવામાં આવે તો શું થશે? દલીલ અને વિધેયમાં ઔપચારિક પરિમાણો માટે અવેજી કરવા માટે તે દલીલ અને વિધેયનો કોડ અનુરૂપ મૂલ્ય પર ચલાવવામાં આવશે. તે જ રીતે એક વર્ગ એક ડેટા પ્રકાર માટે વાદળી પ્રિન્ટ અથવા નમૂના સુચોજિત કરે છે. તે અમને જણાવે છે તે બે વસ્તુઓ કહે છે; આંતરિક અમલ શું છે? ડેટા કેવી રીતે સંગ્રહિત થાય છે? અને તે આપણને વિધેયો આપે છે જે વાસ્તવિક જાહેર ઇન્ટરફેસને અમલમાં મૂકવા માટે વપરાય છે. તેથી, જાહેર ઇન્ટરફેસની મંજૂરી આપતી ઓપરેશન્સને પ્રભાવિત કરવા માટે તમે આંતરિક ડેટાને કેવી રીતે હેરફેર કરો છો હવે એક વાર અમારી પાસે આ નમૂનો છે, આપણે તેના ઘણા ઉદાહરણો બનાવી શકીએ છીએ.

તેથી, તમારી પાસે સ્ટેક માટે વાદળી છાપ છે જે તમે ઘણા સ્વતંત્ર સ્ટેક્સ બનાવી શકો છો, દરેક સ્વતંત્ર સ્ટેક પાસે તે પોતાના ડેટા છે જે સ્ટેક્સ એકબીજા સાથે દખલ કરતી નથી. તેમાંના દરેક પાસે કાર્યની નકલ છે કે જેને અમે તેની સાથે સંકળાયેલ વ્યાખ્યાયિત કરી છે. શાસ્ત્રીય પ્રોગ્રામિંગમાંથી મુખ્ય તફાવતના પ્રકારને બદલે, શાસ્ત્રીય પ્રોગ્રામિંગમાં તમારી પાસે એક કાર્ય હશે જે કહે છે કે પુશને વ્યાખ્યાયિત કરે છે અને તેની પાસે બે માપદંડો હશે, ખાસ કરીને સ્ટેક અને મૂલ્ય. તેથી, તમારી પાસે એક ફંક્શન છે અને પછી તમે તેને સ્ટેક પૂરું પાડશો જે તમે ચાલાકી કરવા માંગો છો. પ્રશ્નની બીજી બાજુએ, હવે અમારી પાસે કેટલાક સ્ટેક્સ S1, S2, S3, વગેરે છે, જે ક્લાસના ઉદાહરણ તરીકે બનાવવામાં આવે છે, અને તાર્કિક રીતે તે દરેકમાં પોતાનું પોશન કાર્ય છે. તેથી S1 સાથે સંકળાયેલ દબાણ છે, કે જે સંકળાયેલ દબાણ S2 છે, S3 સાથે સંકળાયેલ દબાણ અને તેથી. તેમાંના દરેક આ નમૂનામાંથી ઉદભવેલા સમાન વિધેયની એક કૉપિ છે, પરંતુ તે એક ઓબ્જેક્ટ સાથે જોડાય છે. તેથી, આ દ્રશ્યમાં થોડો તફાવત છે, જેના માટે તમે જે ઓબ્જેક્ટને ઓળંગવા માંગો છો તેને ઓબ્જેક્ટ આપવાની જરૂર છે અને તમે તેને કહો છો કે ફંક્શન પોતાને કેવી રીતે લાગુ પાડી શકે છે. તેથી, ચાલો આપણે એક દાખલો જોઈએ કે આ એક વિગતવાર ઉદાહરણ નથી, તે તમને ફક્ત અમે જે વિશે વાત કરી રહ્યા છીએ તેના સ્વાદ આપશે. અહીં ક્લાસ ઢગલાની વ્યાખ્યાની હાડપિંજર છે. તેથી હવે, બિલ્ટ ઇન લીસ્ટનો ઉપયોગ કરવાને બદલે આપણે આપણી પોતાની ડેટા ટાઇપ હીપને વ્યાખ્યાયિત કરવા માંગીએ છીએ. તેથી કેટલાક કાર્ય વ્યાખ્યાઓ છે. આ DEF નિવેદનો અને આ વિધેયોમાંની વ્યાખ્યાને અનુરૂપ છે અને આપણે શું જોશું તે આ વ્યાખ્યાઓની અંદર આપણે તે ઢગલાઓની દરેક નકલમાં સંગ્રહિત મૂલ્યો હશે. તેથી, આ કેવી રીતે કામ કરશે તે વિશે થોડુંક વિચાર. જ્યારે આપણે પ્રકારના ઢગલાના ઓબ્જેક્ટ પર બનાવીએ છીએ ત્યારે આપણે તેને એક કાર્ય જેવું કહીએ છીએ. તેથી, આપણે કહીએ છીએ h બરાબર હીપ । છે, તેથી આ અવ્યવસ્થિતપણે કહે છે કે મને શરૂઆતમાં મૂલ્ય । સાથે ક્લાસ ઢગલોનો એક દાખલો આપે છે. હવે આ ફંક્શન init કહે છે, એટલે કે તેને init કહેવામાં આવે છે. તેથી, init એ કન્સ્ટ્રક્ટર તરીકે ઓળખાતું છે. કન્સ્ટ્રક્ટર એક એવું કાર્ય છે જેને ઓબ્જેક્ટ બનાવવામાં આવે છે ત્યારે તે કહેવામાં આવે છે અને શરૂઆતમાં આ ચોક્કસ કિસ્સામાં તે સેટ કરે છે. અમારા કન્સ્ટ્રક્ટરને 14, 32, 15 ની કિંમતોની એક મનસ્વી યાદી લેવાની ધારણા કરવામાં આવે છે. તેથી, ક્યાંક init કોડ અંદર ત્યાં hAPિફિકેશન ઓપરેશન હશે જે અમે વાસ્તવમાં આ સ્લાઇડમાં બતાવેલ નથી. આ રીતે તમે વસ્તુઓ બનાવી શકો છો તમે પ્રથમ વર્ગ વ્યાખ્યાયિત કરો છો, અમે ટૂંક સમયમાં એક સંપૂર્ણ ઉદાહરણ જોશું, અમે એક વર્ગ વ્યાખ્યાયિત કરીએ છીએ અને પછી તમે વર્ગ સોર્ટને ફંક્શન જેવા કોલ કરો છો અને નામ કે જે આ ફંક્શન કોલ સાથે જોડાય છે અથવા આ વર્ગ નવી ઓબ્જેક્ટ બની જાય છે, જેમ આપણે જણાવ્યું હતું કે અમારી પાસે શામેલ છે અને હેક્સ માટે મહત્તમ વ્યાખ્યાયિત કરે છે જેવા કાર્ય કરે છે, પરંતુ તે જેમ અમારી પાસે દરેક વ્યક્તિગત ઢગલા માટે આ કાર્યની એક અલગ નકલ છે. આ કિસ્સામાં આપણે એક ઢગલો બનાવીએ છીએ, તેથી અમે h લખીએ કે તમારી જાતે 70 મૂલ્ય દાખલ કરો. તેથી, આપણે લખીએ છીએ કે h નો સંદર્ભમાં શામેલ કરો. તેથી, એચ ડોટ 17 શામેલ કરો, જેમ કે એચ 17 દાખલ કરો, જે લખાણોની સામાન્ય કાર્યાત્મક શૈલી હશે. અમે સામાન્ય રીતે તે ઢગલો અને મૂલ્યને પસાર કરીશું, અહીં તેના બદલે અમે કહીએ છીએ કે ઢગલો એચ ને લાગુ પડે છે જે હેપ એચ પર કાર્ય કરે છે. 70 ની દલીલ સાથે કાર્ય શામેલ કરો. આગળની લીટી કહે છે કે હેપ એચમાં ફંક્શન લાગુ પડે છે અને કિંમત 28 માં દાખલ થાય છે. દાખલા તરીકે, આપણે હવે મહત્તમ કિંમત પરત કરવા માટે h ને કહી શકીએ છીએ તે h ડોટ કરો મેક્સ છે અને મુખ્ય વીમાં રીટર્ન મૂલ્ય સંગ્રહિત કરે છે.

(સ્લાઇડ ટાઇ નો ઉલ્લેખ કરો: 14:28)

તો, આપણે શું ભાર મૂકે છે કે અમૂર્ત ડેટા પ્રકાર કાળા બોક્સ તરીકે જોવો જોઈએ. બ્લેક બોક્સની જેમ એક જાહેર ઇન્ટરફેસ બટન્સ છે જે તમે વસ્તુઓને ઉમેરવા, વસ્તુઓ કાઢી નાખવા અને ડેટા પ્રકાર ખાલી છે કે નહીં તે શોધવા માટે ડેટા પ્રકારને અપડેટ કરવા અને ક્વેરી કરવા માટે દબાણ કરી શકો છો. અંદર અમે એક ખાનગી અમલીકરણ છે આ વાસ્તવમાં માહિતીને સાર્વજનિક કાર્યોને યોગ્ય રીતે બનાવવા માટે અમુક ચોક્કસ રીતે ડેટા સંગ્રહિત કરે છે. પરંતુ મહત્વની બાબત એ છે કે, ખાનગી અમલીકરણને

બદલવું ઇનપુટ અને આઉટપુટના સંદર્ભમાં કાર્યો કેવી રીતે વર્તે તે અસર નહીં કરે. કાર્યક્ષમતાના સંદર્ભમાં તેઓ જુદી રીતે વર્તન કરી શકે છે, તમે જોશો કે એક સંસ્કરણ અન્ય કરતાં વધુ ઝડપી છે અથવા તો એક સંસ્કરણ બીજા કરતાં ધીમું છે, પરંતુ આ એવું કહેતા નથી કે કાર્યો બદલાય છે. તેથી, અમે કિંમતોને બદલવા માંગતા નથી, જો અમારી પાસે પ્રાધાન્યતા કતાર હોય અને અમે મૂલ્યોનો સમૂહ દાખલ કરીએ અને પછી મહત્તમ કાઢી નાંખો, અગ્રતા કતાર ખરેખર આંતરિક રીતે કેવી રીતે લાગુ કરવામાં આવે છે, કાઢી નાખવું મહત્તમ અમને અંતે સમાન મૂલ્ય આપવું જોઈએ. તેથી, આપણે જોયું કે પાયથોન ઓબ્જેક્ટ ઓરિએન્ટલ પ્રોગ્રામિંગને ટેકો આપે છે, અમે આ અઠવાડિયાના અભ્યાસક્રમમાં આગામી બે વ્યાખ્યાનો વધુ વિગતવાર જોઈશું, પરંતુ આ નિષ્ક્રિય લક્ષી પ્રોગ્રામિંગ સાથે સંકળાયેલ મુખ્ય ખ્યાલ વર્ગ અને ઓબ્જેક્ટસ છે. વર્ગો ડેટા પ્રકારો માટેના ટેમ્પલેટો છે અને ઓબ્જેક્ટસ આ ક્લાસનાં ઉદાહરણો છે અને તેમની પાસે કોઈક ડેટા પ્રકારો છે જેનો આપણે અમારા પ્રોગ્રામમાં ઉપયોગ કરીએ છીએ.

પ્રોગ્રામિંગ ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રો. માધવન મુકુંદ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ
ઇન્ડિયન ઇન્સ્ટીટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 07 લેક્ચર - 02 ક્લાસ અને ઓબ્જેક્ટ્સ ઇન પાયથોન

(સ્લાઇડ ટાઇ નો ઉલ્લેખ કરો: 00:02)

વ્યાખ્યાનમાં, આપણે જોયું કે ઓબ્જેક્ટ ઓરિઅન્ટ પ્રોગ્રામિંગમાં આપણે ડેટા પ્રકારને એક ક્લાસ દ્વારા વ્યાખ્યાયિત કરીએ છીએ, જે ક્લાસ કહેવાય છે, જે આંતરિક ડેટા અમલીકરણને વ્યાખ્યાયિત કરે છે, અને ડેટા પ્રકારનો ઉપયોગ કરવા માટે અમે જે વિધેયોનો ઉપયોગ કરીએ છીએ. અને ત્યારબાદ આપણે આ ડેટા પ્રકારોના ઘટકોને ઓબ્જેક્ટો તરીકે બનાવીએ છીએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 00:22)

અમે એક ઢગલા એક હાડપિંજર અમલીકરણ જોયું. આમાં એક વિશિષ્ટ કાર્ય હતું જે `init` તરીકે ઓળખાતું હતું, જેનો ઢગલો બનાવવા માટે ઉપયોગ થતો હતો, અને પછી અમારી પાસે કાર્યો શામેલ અને કાઢી નાખવાનાં હતાં. હવે એક વસ્તુ જે અમે સમજાવી નથી તે આ દલીલ સ્વ છે જે આ દ્વારા ચાલે છે. આ એક અજાયબી છે જે આપણને પાયથોન માં છે કે વર્ગમાં વ્યાખ્યાયિત કરેલ દરેક કાર્ય તેના પ્રથમ પેરામીટરનું નામ સ્વ હોવું જોઈએ, હવે તેને સ્વ કહેવામાં આવવાની જરૂર નથી, પરંતુ તે હંમેશાં તેને કોલ કરવા માટે ગૂંચવણમાં મૂકે છે, સ્વ. ચાલો આપણે ધારીએ કે આ પરિમાણ હંમેશાં છે અને તેને સ્વ કહેવાય છે. હવે, સ્વ શું છે, સ્વ એક નામ છે જે વર્ગની અંદર ઉપયોગમાં લેવાયેલા ઓબ્જેક્ટનો સંદર્ભ આપે છે. દાખલા તરીકે, જો આપણે આ ઢગલો સાથે વ્યવહાર કરીએ છીએ, તો જ્યારે આપણે `'h'` દાખલ કરીએ છીએ ત્યારે આ શામેલ 17 ની કિંમતનો ઉપયોગ કરે છે. તેથી, 17 એ મૂલ્ય `x` છે જે દાખલ કરવા માટે પસાર કરવામાં આવી રહ્યું છે, અને `h` એ મૂલ્ય છે જેના પર 17 ઉમેરાવી જોઈએ અને તે સ્વયં માટે એક નામ છે તેથી, બીજા શબ્દોમાં કહીએ તો, તે ક્રિયાને કહે છે જે તેને પોતાના પર કાર્ય કરે છે. તે પોતાને માટે એક નામ છે કારણ કે તમે કાર્યને કહી રહ્યા છો એક ઓબ્જેક્ટ હંપ એચ શામેલ કરો 17 તમારા `'સ્વ'` માં તે અર્થમાં, મારા મૂલ્યો સ્વ દ્વારા અને ઢગલામાં છે, તે અન્ય ઢગલાઓનો સંદર્ભ આપી શકે છે. અમે થોડા સમય પછી જોશું કે અમે એક મૂલ્ય લઈ શકીએ છીએ અને અન્ય મૂલ્યનો સંદર્ભ લઈ શકીએ છીએ. મારા મૂલ્યો હશે અને અન્ય મૂલ્યો હશે. તેથી, મારા મૂલ્યો હંમેશા સ્વાભાવિકરૂપે સ્વ તરીકે ઓળખાય છે, કારણ કે તે એક છે જે સામાન્ય રીતે કાર્ય દ્વારા ચાલાકીથી આવે છે. આને થોડું સ્પષ્ટ કરવા માટે, ચાલો આપણે બધી નોંધો મેળવવા માટે ઢગલાઓ કરતાં સહેજ વધુ સરળ ઉદાહરણ જોઈએ અને પરિભાષા આપણા માટે યોગ્ય છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 02:14)

અમારું પ્રથમ ઉદાહરણ માત્ર એક બિંદુ `x` વાચનું પ્રતિનિધિત્વ છે. તેથી, આપણે ફક્ત એક સામાન્ય સંકલન પદ્ધતિ વિશે વિચારીએ છીએ, જ્યાં અમારી પાસે એક્સ-અક્ષ, `y`- અક્ષ છે. તેથી, આપેલ બિંદુ અલ્પવિરામ બી જેવા કેટલાક સંકલન આપવામાં આવે છે. આ `x`- સંકલન અને `y`- સંકલન `b` સાથે એક બિંદુ છે, આ એક પરિચિત ખ્યાલ છે કે તમે બધાએ ગણિતમાં કોઈક જોયું હોવું જ જોઈએ. તેથી, અમે આવા પદાર્થને બે જથ્થા સાથે સંલગ્ન કરવા માંગીએ છીએ - `x`- સંકલન અને `y`- સંકલન અને આ મૂલ્યો `a` અને `b` પસાર કરીને `init` કાર્ય દ્વારા સુયોજિત છે જે તમે ઇચ્છતા હોવું જોઈએ અને હવે આપણી પાસે આ બિંદુની અંદર છે, આપણી પાસે આ બે વિશેષતા `x` અને `y` છે, એટલે દરેક બિંદુ આની જેમ દેખાય છે. તેમાં `x` મૂલ્ય અને કિંમત છે, અને આ `x` મૂલ્ય કંઈક છે અને `y` મૂલ્ય એ કંઈક છે. અને જો આપણે આ `x`

અને y મૂલ્યને બદલીએ, તો પછી બિંદુ એક સ્થાને બીજા સ્થળે ફેરબદલ કરે છે. હવે, તે નિર્દેશન કરવા માટે કે x અને y આ બિંદુથી છે અને અન્ય કોઈ નહીં, અમે તેને સ્વ દ્વારા પ્રીફિક્સ કરીએ છીએ. તેથી, સેલ્ફ ડોટ એક્સ, આ બિંદુની અંદર x મૂલ્ય ને જાતે સંદર્ભિત કરે છે, સ્વ ડોટ y એ y ની અંદર મારી મૂલ્ય છે. જો તમારી પાસે સામાન્ય બિંદુ p હોય તો આપણી પાસે p ડોટ x છે, p dot y , આ x અને y , x અને y ના બિંદુ p સાથે સંકળાયેલ નામોને સંદર્ભિત કરશે. વર્ગની વ્યાખ્યામાં, સ્વયં આ ચોક્કસ ઓબ્જેક્ટની અંદર એટ્રિબ્યુટ અથવા નામના મૂલ્યનો ઉલ્લેખ કરે છે. હવે આ ચોક્કસ ઓબ્જેક્ટ બદલાય છે કારણ કે આપણે એક ઓબ્જેક્ટમાંથી બીજા પર ખસેડીએ છીએ, પરંતુ પ્રત્યેક ઓબ્જેક્ટ માટે પોતે જ છે. તેથી, પૃષ્ઠ 1 માટે જો હું પી 1 ના સ્વરૂપે પી 1 ના સ્વરૂપે કંઈક બોલીશ તો પૃષ્ઠ 1 છે. જો મારી પાસે પૃષ્ઠ 2 ના સંદર્ભમાં અલગ અલગ બિંદુ છે 2, સ્વ છે પૃષ્ઠ 2. આ એક મહત્વપૂર્ણ વસ્તુ છે યાદ રાખો કે વર્ગની વ્યાખ્યામાં દરેક કાર્ય હંમેશા સ્વયં તરીકે પ્રથમ દલીલ હોવું જોઈએ અને પછી વાસ્તવિક દલીલ. તેથી, આ કિસ્સામાં `init` બે દલીલો લે છે, પરંતુ અમે હંમેશા ત્રીજા દલીલ દાખલ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 04:27)

ચાલો જોઈએ કે આ કેવી રીતે કાર્ય કરે છે. દાખલા તરીકે, જો આપણે કહીએ કે p બરાબર બિંદુ 3, 2; પછી 3 એક તરીકે પસાર થશે, અને 2 બી તરીકે પસાર થશે. અને આ એક બિંદુ બનાવશે જે આપણે અહીં દોર્યું છે, કે જે આંતરિક રીતે સ્વ ડોટ x 3 તરીકે રજૂ થાય છે અને સ્વ ડોટ y છે 2. હવે અહીં થોડો અલગ કાર્ય છે, તે એક બિંદુ લે છે અને તેને બદલ્યું છે. તેથી, તમે આને 3 વત્તા ડેલ્ટા x અને 4 વત્તા ડેલ્ટા વાય માં પાળી શકો છો. આ કાર્ય અમે અનુવાદ કહેવાય છે. તેથી, તે પાળીને દલીલ, ડેલ્ટા એક્સ અને ડેલ્ટા વાય તરીકે લે છે. અને હંમેશની જેમ આપણે હંમેશા સ્વયંને ડિફોલ્ટ પ્રથમ દલીલ તરીકે આપીએ છીએ. તે ફક્ત દરેક પાયથોન કાર્ય, દરેક પાયથોન વર્ગને ધ્યાનમાં રાખીને રાખો જો તમે ઓબ્જેક્ટ ઓરિએન્ટલ શૈલીમાં ફક્શનનો ઉપયોગ કરવા માંગતા હો, તો પ્રથમ દલીલ જરૂરી હોવી જોઈએ અને પછી વાસ્તવિક દલીલો. તેથી, જ્યારે આપણે કોઈ બિંદુને અનુવાદિત કરવા માંગીએ છીએ ત્યારે શું કરવું છે, આપણે સ્વયં ડોટ કરવા માંગીએ છીએ અને તેને સ્વ ડોટ x વત્તા વેલ્યુ ડેલ્ટા એક્સ પર ખસેડો. તેથી, તમે સ્વ ડોટ એક્સ વત્તા ડેલ્ટા એક્સ ઇચ્છો છો. હવે, આ પાયથોન અને અન્ય પ્રોગ્રામિંગ લેંગ્વેજ્સમાં એક બહુ સામાન્ય નમૂના છે જ્યાં તમે એક નામ `"z"` લેવા ઇચ્છો છો, અને પછી તમે તેને અમુક રકમ દ્વારા પાળી શકો છો, z વત્તા 6 અથવા z બરાબર z ઓછા 6. જ્યારે આપણે આ પ્રકારની એવી વસ્તુ છે કે જ્યાં તે જ નામ અપડેટ કરવામાં આવી રહ્યું છે, ત્યાં એક ટૂંકા સ્વરૂપ છે જ્યાં આપણે અસાઇનમેન્ટ સાથે ઓપરેશનને જોડીએ. તેથી, સેલ્ફ ડોટ એક્સ વત્તા અર્ધ ડેલ્ટા એક્સ સ્વયં ડોટ એક્સ માટે સ્વયં ડોટ એક્સ વત્તા ડેલ્ટા એક્સ માટે પાયથોન માટે ટૂંકો કટ છે; તેનો મતલબ એવો થાય છે કે અસામાન્ય રીતે ડાબી બાજુનું નામ સોંપણી પ્રક્રિયા સાથે ઉલ્લેખિત ઓપરેશનમાં પ્રથમ દલીલ છે. આ એક ખૂબ અનુકૂળ શોર્ટકટ છે જે આપણને અમુક ટાઇપિંગ બચાવવા માટે પરવાનગી આપે છે. સેલ્ફ ડોટ એક્સ સમાન સેલ્ફ ડોટ એક્સ વત્તા ડેલ્ટા એક્સ લખવાને બદલે આપણે ફક્ત સેલ્ફ ડોટ એક્સ વત્તા ડેલ્ટા એક્સની બરાબર કહીએ છીએ. આ અનુવાદને પ્રદાન કરેલા દલીલ દ્વારા વર્તમાન બિંદુના એક્સ સંકલનને બદલે છે. તેવી જ રીતે, સેલ્ફ ડોટ વાય વત્તા ડેલ્ટા વાયની બરાબર દલીલને ડેલ્ટા વાય દ્વારા પ્રદાન કરેલી રકમ દ્વારા બદલશે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 06:37)

દાખલા તરીકે, હવે જો આપણે કહીએ તો ડોટ ટીપ્પેટ 2 1 તો આપણને નવું બિંદુ મળે છે જે x કોઓર્ડિનેટ માટે 2 વત્તા 2 વત્તા 2 વત્તા 1 3 છે, તેથી આ 3 વત્તા 2 આપણને 5 આપે છે, અને 2 વત્તા 1 આપણને 3 આપે છે. આ બિંદુને 3, 2 થી 5, 3 માંથી બદલ્યો છે. આ રીતે આપણે આ આંતરિકને વ્યાખ્યાયિત કરીએ છીએ. આંતરિક અમલીકરણ `init` કાર્યમાં વ્યાખ્યાયિત થયેલ છે; આ એ કાર્ય છે જે કહેવામાં આવે છે કે બિંદુ ક્યારે સેટ કરેલું છે અને આ વસ્તુઓને સાથે આ આંતરિક નામો x અને y જોડે છે. આ તે છે જ્યાં અમલીકરણ ખરેખર સેટ અપ થાય છે, અને પછી વિધેયો જે અમે બાહ્ય રીતે

વ્યાખ્યાયિત કરીએ છીએ જેમ કે ભાષાંતર, આ આંતરિક પ્રતિનિધિત્વને યોગ્ય રીતે ચાલાવો, જેથી તે કાર્યોની અપેક્ષા રાખે તે પ્રમાણે સતત બદલાય.

(સ્લાઇડ સમયનો સંદર્ભ લો: 07:24)

ચાલો આપણે વિવિધ કાર્યો જોઈએ. તેથી ધારવું કે આપણે મૂળથી બિંદુની અંતર ગણતરી કરવા માગીએ છીએ. તેથી, આપણે જાણીએ છીએ કે આ અંતર શું છે. પાયથાગોરસ પ્રમેય દ્વારા આ અંતર એક્સ ચોરસ વત્તા વાય ચોરસનું વર્ગમૂળ છે. તેથી, યાદ રાખો કે આ જમણી બાજુ ત્રિકોણના હાયપોટેનેઝ જેવા છે. તેથી, તમે ax વર્ગ વત્તા y વર્ગમૂળ લો અને તમને ડી મળે. જો તમે કોઈ બિંદુની અંતર ઇચ્છતા હો, તો અમે તેને કોઈપણ દલીલો આપતા નથી, પરંતુ અમે હંમેશા આ મૂળભૂત દલીલ સ્વ છે. તેથી, આપણે જાણવા માગીએ છીએ કે $0, 0$ થી વર્તમાન બિંદુથી અંતર શું છે. તેથી, આપણે કહીશું કે આપણા પહેલાંના કેસમાં કંઈક p બરાબર બિંદુ કહે છે 3 અલ્પવિરામ 4 અને તે પછી આપણે કહીશું p ડોટ ઓ અંતર એ અંતર મેળવવા માટે. કદાચ આપણે આને નામ આપીએ, ચાલો આપણે તેને કોલ ન કરીએ. તો, આપણે આને n ના જેમ નામ આપીએ છીએ. તેથી, જ્યારે આપણે આ કરીએ છીએ, તે સ્વ ડોટની વર્તમાન કિંમત, સ્વયં ડોટની વર્તમાન કિંમત, તેને ચોરસ કરશે, તેને ઉમેરો અને વર્ગમૂળ લેશે. હવે યાદ રાખવું એક વસ્તુ એ છે કે વાસ્તવમાં ચોરસ રૂટ એ ફંક્શનમાં મૂળભૂત રીતે ઉપલબ્ધ ફંક્શન નથી. તેથી, તમારે ખરેખર ગણિત પુસ્તકાલય આયાત કરવું પડશે. તમારી વર્ગની વ્યાખ્યાની ટોચ પર, તમારે ગણિત આયાત તારોમાંથી લખવાનું યાદ રાખવું જોઈએ. એમ ધારી રહ્યા છીએ કે આપણે તે પૂર્ણ કર્યા પછી વર્ગમૂળ વ્યાખ્યાયિત થયેલ છે. આ એક વિશિષ્ટ કાર્ય છે જે બિંદુ વિશે કેટલીક માહિતી આપે છે; અગાઉના કાર્ય ફક્ત બિંદુ ભાષાંતર, અમને કંઈપણ ન કહી નહોતી; આ એ કાર્ય છે જે બિંદુ વિશેની માહિતી આપે છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 09:01)

હવે જો ઓ અંતર એ કંઈક છે જે આપણને વારંવાર કરવાની જરૂર છે, તો પછી તે ફક્ત એક અલગ રજૂઆતમાં રાખવા માટે ઉપયોગી છે. તેથી, તમે યાદ રાખી શકો અથવા તમે હાઈ સ્કૂલના ગણિતમાં ક્યાંય જોયું હશે જો તમે બિંદુ x, y પસંદ કરો છો, તો આ બિંદુ વાસ્તવમાં ધ્રુવીય કોઓર્ડિનેટ્સનો ઉપયોગ કરે છે તે રજૂ કરવાની એક વૈકલ્પિક રીત છે. તેથી, તમે આ અંતર રાખી શકો છો અને તમે આ કોણ રાખી શકો છો. તેથી, જો મારી પાસે r અને થીટા હોય તો તે x અને y રાખતી સમાન માહિતી છે. બે વચ્ચેના સંબંધ એ છે કે x એ આર કોસ થીટા છે જ્યાં કોન ટ્રિગોમિટરિક કોઝાઇન કાર્ય છે; y બરાબર આર પાપ થીટા છે. અને બીજી બાજુ, જો મારી પાસે x અને y હોય, તો હું રિક પુનઃ મેળવી શકું છું કારણ કે આપણે ફક્ત 0 અંતર માટે કર્યું છે તે x ચોરસ વત્તા વાય ચોરસનું વર્ગમૂળ છે, અને થિટા તેથી વાય દ્વારા x વાસ્તવમાં જો તમે જો y ને વિભાજીત કરો તો x તમે થિટા ની તન મેળવી શકો છો કારણ કે તે કોસ દ્વારા વિભાજીત થયેલ પાપ છે અને આર રદ કરે છે. તેથી, x દ્વારા y એ ટેન થીટા છે, તેથી થિટા એ x ની ટન ઇનવર્સ છે. હવે અમલીકરણ બદલવાની વાત કરીએ તો, અમે અમારા અમલીકરણને બદલી શકીએ છીએ, જેથી અમે x અને y ની દ્રષ્ટિએ આંતરિક પ્રતિનિધિત્વ ન રાખીએ, અમે ખરેખર તે આર અને થિટા દ્રષ્ટિએ ઇપી, પરંતુ કાર્યો જ રહે છે.

(સ્લાઇડ સમય જણાવો: 10:22)

દાખલા તરીકે, આપણે અગાઉની વ્યાખ્યા લઈ શકીએ અને તેને બદલી શકીએ. તેથી, આપણે ફરીથી તેને x અને y પાસ કરીએ. તેથી, વપરાશકર્તાની સંભવિત થી, વપરાશકર્તા માને છે કે બિંદુને x અને y સંકલનની દ્રષ્ટિએ વ્યાખ્યાયિત કરવામાં આવે છે, પરંતુ બિંદુ તરીકે સી અને બી એ સીધી દલીલ કરવા માટે ઉપયોગ કરવાને બદલે, આપણે પહેલા r - ત્રિજ્યા ચોરસ વત્તા ચોરસનું ચોરસ રૂટ લઈને. અને પછી આધાર રાખીને, તેથી અમે એ દ્વારા b ને વિભાજીત કરવા માંગીએ છીએ, પરંતુ જો 0 હોય, તો

આપણી પાસે એક વિશિષ્ટ કેસ છે જે આપણને એક ભૂલ આપશે. તેથી, જો 0 હોય, તો આપણે 0 ને કોણ સુયોજિત કરીએ છીએ; નહિંતર, આ ટેન વ્યસ્ત, ચાપ ટેન માટે ગણિત લાઇબ્રેરીમાં પાયથોન કાર્ય છે, અમે એટીએ ગોઠવીએ છીએ કે તે આર્ક ટન બી બાદબાજોને એક ભાગમાં વિભાજિત કરે. તેથી, આપણે આંતરિક રીતે xy સંસ્કરણને આર થિટામાં તે જ સૂત્રનો ઉપયોગ કરીને ચાલાકી કરીએ છીએ જે પહેલાં આપણે બતાવી દીધું છે કે જે r એ x ચોરસ વ યુ વાય ચોરસનું વર્ગમૂળ છે અને થીટા x દ્વારા y ની રાતામાં વ્યસ્ત છે. આપણે જે વસ્તુની કાળજી લેવી પડશે તે જ છે જ્યારે x બરાબર 0 હોય, તો આપણે જાતે 0 થીટાને સુયોજિત કરવું પડશે. હવે આંતરિક રીતે આપણે હવે સ્વ ડોટ આર અને સેલ્ફ ડોટ થીટા રાખીએ છીએ. અમે સેલ્ફ ડોટ એક્સ અને સેલ્ફ ડોટ યો રાખતા નથી. આ ઉપયોગી છે કારણ કે જો તમે 0 અંતર ઇચ્છતા હોવ - મૂળ અંતરને આપણે ફક્ત આર મૂલ્ય પરત કરવું પડશે, અમે કોઈ ગણતરી નથી કરત. તેથી, બીજા શબ્દોમાં જો આપણે ઓ અંતરનો ઉપયોગ વારંવાર કરવા જઈ રહ્યા છીએ તો તે ગણતરી માટે ચોરસ રુટ એક ચોરસ વત્તા b ચોરસનો ઉપયોગ કરવા માટે વધુ સારું છે જ્યારે આપણે બિંદુ સુયોજિત કરીએ છીએ, અને જ્યારે કોઈપણ ગણતરી વગર r મૂલ્ય પાછી મળે છે તમે મૂળથી અંતર ઇચ્છો છો આ તમે કેવી રીતે ઉપયોગ કરી રહ્યા છો તેના આધારે આ એક આવશ્યકતા હોઈ શકે છે અને એક અમલીકરણ અન્ય કરતાં વધુ સારી હોઈ શકે છે, પરંતુ વપરાશકર્તાની પરિપ્રેક્ષ્યમાં સમાન કાર્ય એ છે કે સ્વયં ત્યાં છે અંતર. તેથી, જો હું કોઈ બિંદુ લઈશ અને હું ઓ અંતર માટે પૂછતો હોઉં તો મને મૂળથી અંતર મળે છે કે નહીં તે બિંદુનો ઉપયોગ xy અથવા r થીટા દ્વારા થાય છે. હવે, અલબત્ત, ઓ અંતરનો ઉપયોગ કરીને આર થિટા અનુવાદ માટે સારું છે 0 અંતર માટે સારું છે. જો હું ડેલ્ટા એક્સ ડેલ્ટા વાય દ્વારા બિંદુ અનુવાદ કરવા માગું છું, તો મને બિંદુને આર થીટીસથી x, y ; એક્સ સમાન દુ આર કોસ થીટા અને y બરાબર આર સીન થીટાનો ઉપયોગ કરીને x વત્તા ડેલ્ટા એક્સ, વાય વત્તા ડેલ્ટા વાય અને તેને આર થીટા જમણામાં ફેરવો. તેથી, તમે એક ફંક્શન અથવા અન્યમાં કિંમત ચૂકવો છો; સાથે xy રજૂઆત અનુવાદ સારું છે; આર થિટા પ્રતિનિધિત્વ સાથે, અંતર સારું છે. અને આ પ્રકારની સમાધાનના એક વિશિષ્ટ કેસ છે જેનો તમારે વ્યવહાર કરવો પડે છે અને તમારે તે નક્કી કરવું પડશે કે આ કામગીરી કઈ સીધી સીધી અમલ કરવા માટે વધુ સામાન્ય અને વધુ ઉપયોગી છે. જો તમને લાગે કે ભાષાંતર વધુ વખત થાય છે તો તે કદાચ x અને y નો ઉપયોગ કરવા માટે વધુ સારું છે; જો તમને લાગ્યું કે અંતરનું મૂળ વધુ અગત્યનું છે, તેથી કદાચ r અને θ નો ઉપયોગ કરવો વધુ સારું છે. તેથી, ઘણી વાર કોઈ સારા જવાબ નથી. એવું કહીને એવું નથી કે અગ્રતા કતાર માટે ક્રમમાં ગોઠવેલી સૂચિ અમલીકરણ કરતાં હીપ અમલીકરણ હંમેશાં સારું છે. તેમાં એવા ટ્રેડઓફ્સ હોઈ શકે છે જે ઉપયોગના પ્રકાર પર આધાર રાખે છે જે તેઓ ડેટા માળખું મૂકવા જતા હોય છે જેમ કે આંતરિક અમલીકરણ સૌથી ખરાબમાં ખરાબ કામ કરે છે, પરંતુ તમારે હંમેશા ધ્યાનમાં રાખવું જોઈએ કે અમલીકરણ એ કાર્યને અનુકૂળ ન થવું જોઈએ. બાહ્ય વપરાશકર્તાને, કાર્યને તે જ રીતે વર્તવું જ જોઈએ.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 13:31)

આ ખાસ ઉદાહરણમાં આપણે જે જોયું છે તે ફરીથી સમજાવવા માટે, ફરીથી. અમે ખાનગી અમલીકરણ બદલ્યું છે, એટલે કે આપણે એક્સ, વાય થી આર થીટામાં ખસેડી દીધું છે, પરંતુ પબ્લિક ઇન્ટરફેસની કાર્યક્ષમતા, કાર્યો અથવા અંતરનું અનુવાદ વગેરે વગેરે બરાબર જ છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 13:48)

હવે આપણે અગાઉ જોયું છે કે પાયથોન વિધેયોમાં, આપણે ડિક્ઝેલ્ટ આર્ગ્યુમેન્ટ્સ આપી શકીએ છીએ જે કેટલીક વખત દલીલને વૈકલ્પિક બનાવે છે. તેથી, દાખલા તરીકે, જો તમે કહેવા માગો છો કે જો આપણે કોઈ બિંદુ પર એક્સ અને વાય કોઓર્ડિનેટ્સ સ્પષ્ટ ન કરીએ તો ડિક્ઝેલ્ટ દ્વારા બિંદુ મૂળ પર બનાવવામાં આવશે. પછી આપણે 0 ની સમાન, અને b ને 0 ની મૂળભૂત દલીલો તરીકે વાપરી શકીએ છીએ, જેથી જો વપરાશકર્તા એ અને b માટે મુલ્ય ન આપે તો, એક્સ 0 માં સુયોજિત થશે અને y 0 થી સુયોજિત થશે. દાખલા તરીકે, જો આપણે કોઈ ચોક્કસ સ્થળ 3 અલ્પવિરામ 4 પર નિર્દેશ કરવા ઇચ્છીએ છીએ, તો

આપણે આ વર્ગ આ વિધેયને બોલાવીશું, અમે દલીલ 3 અલ્પવિરામ 4 પસાર કરીને એક ઓબ્જેક્ટ બનાવીશું, પરંતુ જો આપણે કોઈપણ દલીલને p 2 ન આપીશું તો પછી આપણને એક બિંદુ મળશે મૂળ.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 14:35)

આ ફંક્શન init એક વિશેષ કાર્ય જેવો દેખાય છે કારણ કે આ બંને બાજુના અંડરસ્કોર અન્ડરસ્કોરને કારણે આપણે સામાન્ય રીતે ન જોઈ શકીએ કે જ્યારે આપણે સામાન્ય રીતે પાચથોન ફંક્શન લખવાનો ઉપયોગ કરતા નથી. જેમ આપણે કહ્યું હતું કે પિથન એક init ને કન્સ્ટ્રક્ટર તરીકે અર્થઘટન કરે છે, તેથી જ્યારે આપણે પબ્લિક ઓબ્જેક્ટને 0.54 સમાન કહીએ છીએ, ત્યારે આ ઇન્સ્ટોલિબલ રીતે init અને init ને સેલ્ફ ડોટ એક્સ સ્વ ડોટ યુ સુયોજિત કરવા માટે વપરાય છે. બિંદુની આંતરિક પ્રતિનિધિત્વ init દ્વારા યોગ્ય રીતે સુયોજિત થયેલ છે. તેથી, init એક વિશેષ કાર્ય છે. હવે પાચથોન ને અન્ય વિશિષ્ટ કાર્યો છે. દાખલા તરીકે, એક વસ્તુ જે આપણે કરવા માગીએ છીએ તેમાંથી કોઈ એક ઓબ્જેક્ટની કિંમત છાપવાનો છે, ઓબ્જેક્ટ શું કરે છે. અને આ માટે સૌથી સાનુકૂળ રીત એ છે કે ઓબ્જેક્ટને સ્ટ્રિંગમાં કન્વર્ટ કરવું. સામાન્ય રીતે કાર્ય strings શબ્દમાળામાં ફેરવે છે, પરંતુ આપણે કેવી રીતે વર્ણવવું જોઈએ કે કેવી રીતે str એક વસ્તુ માટે વર્તે છે, તો ત્યાં વિશિષ્ટ કાર્ય છે કે જેને આપણે અંડરસ્કોર અંડરસ્કોર str લખી શકો છો. તેથી, અંડરસ્કોર અંડરસ્કોર str એટલે જ જ્યારે અમે str નું લખીએ ત્યારે સર્વસામાન્ય રીતે જ વિનંતી કરે છે. તેથી, ઓબ્જેક્ટનો str એ કંઈપણ નથી પરંતુ ઓડ અંડરસ્કોર અંડરસ્કોર str છે. અને દાખલા તરીકે, છાપો - વિધેય પ્રિન્ટને છાપેલી કોઈપણ નામ તમે પ્રિન્ટ કરવા માટે પસાર કરે છે અને તેને સ્ટ્રિંગમાં રૂપાંતરિત કરે છે, જ્યારે હું કહું છું કે પ્રિન્ટ x અને x પૂર્ણાંક છે, પૂર્ણાંક x ની સ્ટ્રિંગ છે જે પ્રદર્શિત થાય છે. તેથી, સ્ટ્રોકનો ઉપયોગ કરવામાં આવે છે અને આંતરિક રૂપે આંતરિક રીતે આ વિશિષ્ટ કાર્યને અંડરસ્કોર અંડરસ્કોર str તરીકે બોલાવે છે. ચાલો જોઈએ કે કેવી રીતે આ આપણા બિંદુની વસ્તુ માટે ઉદાહરણ તરીકે કામ કરશે. તેથી, જો આપણે બિંદુઓનું પ્રતિનિધિત્વ કરવા માગીએ છીએ તો આપણે સ્વયં ડોટ એક્સ અને સેલ્ફ ડોટ વાય છીએ, આપણે આ ફોર્મ મુલ્ય x અને વેલ્યૂ વાયમાં આને છાપીએ છીએ. તો, આપણે શું કરીએ? અમે str સુયોજિત, તેથી યાદ રાખો કે સ્વ હંમેશા પરિમાણ છે. તેથી, તે શું કરે છે, તે પ્રથમ ખુલ્લું કૌંસ અને બંધ કૌંસ ક્યાં અંત અને મધ્યમાં એક અલ્પવિરામ સાથે એક સ્ટ્રિંગ બનાવે છે; અને ખુલ્લા કૌંસ અને અલ્પવિરામ વચ્ચે તે સ્વયં ડોટની સ્ટ્રિંગ રજૂઆત કરે છે અને અલ્પવિરામ અને બંધ કૌંસ વચ્ચે તે સ્વયં ડોટની સ્ટ્રિંગ પ્રતિનિધિત્વ કરે છે. આ મૂલ્યમાંથી એક સ્ટ્રિંગ બનાવે છે, તે આંતરિક રીતે પોતાને સ્વ ડોટ એક્સ અને સેલ્ફ ડોટ વાય પર સ્ટ્ર કરે છે અને પછી આ વધારાની વસ્તુઓને ઓપન ક્લોઝ કૌંસ અને અલ્પવિરામ બનાવે છે જેથી તે એક બિંદુ જેવો દેખાય તેમ અમે અપેક્ષા રાખીએ.

(સ્લાઈડ સમયનો સંદર્ભ લો: 16:55)

એક અન્ય રસપ્રદ કાર્ય કે જે પાચથોન ને એક ખાસ કાર્ય તરીકે ઉમેરવામાં આવે છે. તેથી, જ્યારે આપણે લખીએ વત્તા ફંક્શન એડ ઉમેરવામાં આવે છે. બીજા શબ્દોમાં કહીએ તો પી 1 વત્તા પી 2, જો આ બે પોઇન્ટ હશે તો p 1 અંડરસ્કોર અન્ડરસ્કોર એડ ઉમેરશે 2. તેથી, આપણે શું અપેક્ષા રાખીએ છીએ કે જો મારી પાસે પૃષ્ઠ 1 હશે અને જો મને 2 p હશે તો મને કંઈક મળશે જે મને આપે છે એક બિંદુ જ્યાં હું આ બે ભેગા. તેથી, હું x સંકલનને x 1 p 1 વત્તા પી 2 અને વાય કોઓર્ડિનેટ p 1 વત્તા પૃષ્ઠ 2 તરીકે મેળવી શકું છું. તે અમારા પર છે હું તેનો અર્થ એ નથી કે તેનો આ રીતે હોવો જોઈએ, પરંતુ જો હું કહું કે p 1 p 2 તે ફંક્શન જેનો ઉપયોગ કરવામાં આવે છે તે એડ કરે છે. અને તે આપણી ઉપર નિર્ધારિત છે કે શું ઉમેરવું અર્થ છે. ચાલો ધારીએ કે આપણે એક નવું બિંદુ બનાવવા માંગીએ છીએ જેની x સંકલન અને વાય સંકલન અમને આપેલ બે બિંદુઓનો સરવાળો છે.

(સ્લાઈડ સમયનો સંદર્ભ લો: 17:42)

અહીં તે રીત છે કે અમે તે કરીશું; આપણે એક નવું બિંદુ બનાવશું જેનું x સંકલન સ્વ ડોટ x વત્તા પી ડોટ x છે. હવે, નોંધ લો કે, સ્વયં એ આ કિસ્સામાં પૃષ્ઠ 1 સાથે સંકળાયેલું કાર્ય છે, ઉમેરો; તેથી સ્વ પેજનો ઉલ્લેખ કરે છે 1. જ્યારે હું કહું છું $p1$ વત્તા પૃષ્ઠ 2 અને અન્ય દલીલ $p2$ એ બિંદુ p છે. તેથી, હું p ની મુલ્ય જોઈ અને p ડોટ x ડોટ y લખી શકું, હું મારી મુલ્ય જોઈ શકું છું અને સેલ્ફ ડોટ x સેલ્ફ ડોટ કહી શકું છું, અને હવે હું આને નવી બિંદુ સ્વ ડોટ એક્સ વત્તા પી બિંદુ બનાવીને ભેગા કરી શકું છું. x અને સેલ્ફ ડોટ એક્સ વત્તા પી ડોટ વાચ દાખલા તરીકે, જો આપણી પાસે 1, 2 અને 2, 5 માં બે બિંદુઓ હોય તો, તે 3, 7 માં એક બિંદુ પાછો આપશે અને હું તેને નવી બિંદુમાં સંગ્રહિત કરું છું, તેથી $p3$ હવે એક નવું બિંદુ બની જાય છે જેની x સંકલન 3 અને y છે. સંકલન 7 છે.

(સ્લાઈડ ટાઇમ નો ઉલ્લેખ કરો: 18:30)

તે જ રીતે, આપણે ગુણાકારને વ્યાખ્યાયિત કરવા માટે એક વિશેષ રીત હોઈ શકે છે, અને પાયથોન માં અંડરસ્કોર અંડરસ્કોર મલ્ટ ફંક્શન એ તે છે જે સર્વસામાન્ય રીતે ગુણાકાર દ્વારા કહેવામાં આવે છે. એ જ રીતે, આપણે તુલના વ્યાખ્યાયિત કરી શકીએ છીએ; આપણે કહી શકીએ કે પીએ 1 પી કરતાં ઓછી છે કે કેમ તેની તુલના કરવી છે, શું અમે તપાસ કરીએ છીએ કે બંને કોઓર્ડિનેટ્સ ઓછી છે, આપણે તપાસ્યું છે કે ઉત્પત્તિની અંતર ઓછી છે; આને કેવી રીતે વ્યાખ્યાયિત કરવું તે સંપૂર્ણ સ્વતંત્રતા છે અમે ફક્ત અમારા પ્રોગ્રામમાં પૂરતી વાંચવા યોગ્યતા માટે $p1$ ઓછા પછી $p2$ લખીએ છીએ, અને આંતરિક રીતે તે લેટી કરતાં ઓછા કાર્યને કોલ કરશે, આ જ રીતે ફંક્શન જીટીને કોલ કરશે, અને ત્યાં બરાબર પછી બરાબર બરાબર અને કંઈક બરાબર છે તેથી પર આ બધા ખૂબ અનુકૂળ વિધેયો છે કે જે પાયથોન અમને સર્વથા કોલ કરવા માટે પરવાનગી આપે છે, અને અમને અમારા કોડમાં પરંપરાગત ઓપરેટર્સ અને પરંપરાગત કાર્યોનો ઉપયોગ કરવાની મંજૂરી આપે છે. તેથી, આપણે વાસ્તવમાં આ વસ્તુઓને અલગ રીતે વિચારવું પડશે નહીં જ્યારે આપણે અમારા કાર્યક્રમોને કોર્ડિંગ કરીશું. આવા કેટલાક અન્ય કાર્યો છે; તે બધાને સૂચિબદ્ધ કરવું અશક્ય છે અને જ્યારે તમે પાયથોન શીખતા હો ત્યારે આ તમામ પ્રારંભિક તબક્કામાં સૂચિબદ્ધ કરવા માટે તે ઉપયોગી નથી, પરંતુ તમે હંમેશા પાયથોન દસ્તાવેજો શોધી શકો છો અને ઓબ્જેક્ટ્સ માટે વ્યાખ્યાયિત તમામ વિશેષ કાર્યો અને જુઓ વર્ગો

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રોફેસર. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 07 લેક્ચર - 02 યુઝર નિર્ધારિત સૂચિ

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 00:03)

હવે અમે બેઝિક્સ વિશે જોયું છે. વર્ગો અને ઓબ્જેક્ટોને કેવી રીતે વ્યાખ્યાયિત કરવું, ચાલો આપણે રસપ્રદ ડેટા માળખું વ્યાખ્યાયિત કરીએ. ધારો કે અમે સૂચિનાં આપણા પોતાના સંસ્કરણને અમલ કરવા માગીએ છીએ, સૂચિ મૂળભૂત રીતે ગાંઠોનો ક્રમ છે અને અનુક્રમમાં દરેક નોડ આગામી નોડમાં મૂલ્ય અને ગુણો ધરાવે છે. તેથી, સૂચિમાં જવા માટે આપણે શરૂઆતમાં શરૂ કરવું પડશે અને આ પોઇન્ટર નીચે ચાલવું પડશે જ્યાં સુધી આપણે છેલ્લા પોઇન્ટર સુધી પહોંચતા નથી જે કંઈ નહીં નિર્દેશ કરે છે. અમારી પાસે પાયથોન નોટેશનમાં ફોર્મ v1, v2, v3, v4 ની સૂચિ છે. પછી આ રીતે આપણે કલ્પના કરી શકીએ છીએ તે વાસ્તવમાં રજૂ થાય છે. 4 ગાંઠો છે આ સૂચિ. જે આપણે આ સૂચિમાં પ્રથમ નોડ પર નિર્દેશ કરી છે, v 1 બિંદુઓ વી 2, v 2 પોઇન્ટ વી 3 અને v 3 બિંદુઓ v 4. અંતિમ નોડ કંઈ નિર્દેશ કરે છે અને તે દર્શાવે છે કે આપણે યાદીના અંત સુધી પહોંચી ગયા છો. આ પ્રતિનિધિત્વમાં ખાલી યાદી શું દેખાશે, તેવું માનવું સ્વાભાવિક છે કે ખાલી સૂચિમાં એક નોડનો સમાવેશ થશે જેમાં મૂલ્ય અને આગળના નોડ પોઇન્ટર બંનેમાં રાખવામાં આવશે, ઉદાહરણ તરીકે સિંગલટોન એક નોડ હશે જેમાં આપણી પાસે મૂલ્ય v 1 અને આગળ કોઈ સેટ નથી. તેથી, આ એક સંમેલન છે જે અમે યાદીના અમારા પ્રતિનિધિત્વ માટે અનુસરવું જોઈએ. તેથી, નોંધ લો કે જ્યાં સુધી અમારી પાસે એક નોડ નહીં સાથેની ખાલી સૂચિ છે, ત્યાં સૂચિમાં કોઈ અન્ય નોડ મૂલ્ય નહીં, કોઈ જ નહીં. આ કંઈક એવી છે કે જે અમે ગર્ભિત રીતે ધારણ કરીશું અને મૂલ્ય માટે તે ચકાસણીનો ઉપયોગ કરીશું નહીં તે અમને જણાવશે કે તે એક ખાલી સૂચિ છે અને સૂચિમાં મધ્યમાં કોઈને અમે ક્યારેય શોધીશું નહીં. અમે ફક્ત સિંગલટોન અને એક ખાલી સૂચિ વચ્ચેનો તફાવત સ્પષ્ટ રીતે મૂલ્ય પર આધારિત છે. તેમાંના બંને એક નોડ ધરાવે છે. હવે આનું કારણ આપણી પાસે છે કારણ કે વાસ્તવમાં પાયથોન આપણને ખાલી યાદી બનાવવા માટે પરવાનગી આપતું નથી જો આપણે કહીએ કે 1 કંઈક બરાબર છે અને આપણે ખાલી યાદીને દર્શાવવું જોઈએ તો સમસ્યા એ છે કે કોઈ પણ પાસે નથી જ્યાં સુધી પાયથોન ની કિંમત સિસ્ટમ ચિંતિત છે. તેથી, એક વાર અમારી પાસે કોઈ નહીં હોય, અમે આ સૂચિ પ્રકાર માટે બનાવવા માટેના ઓબ્જેક્ટ ફંક્શનોને લાગુ કરી શકતા નથી. તેથી, અમને કંઈક બનાવવાની જરૂર છે જે યોગ્ય પ્રકારથી ખાલી છે. તેથી, અમારે ઓછામાં ઓછા 1 નોડ બનાવવાની જરૂર છે અને એટલે જ આપણે ખાલી યાદીને દર્શાવવા માટે આ પ્રતિનિધિત્વનો ઉપયોગ કરવાની જરૂર છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 02:26)

અહીં તે મૂળભૂત વર્ગ છે જેનો આપણે ઉપયોગ કરવા જઈ રહ્યા છીએ, તે ક્લાસ નોડ છે. તેથી, દરેક નોડની અંદર આપણી પાસે 2 એટ્રીબ્યુટ્સની કિંમત છે અને આગળ આપણે કહ્યું હતું અને તે યાદ રાખવું જોઈએ કે સ્વયં હંમેશા વિધેયને ધ્યાનમાં રાખીને દરેક કાર્ય સાથે વપરાય છે. અમે આ ડિફોલ્ટ યોજનાનો ઉપયોગ કરીશું, જો આપણે કંઈપણ ન કહીશું તો અમે ખાલી યાદી બનાવીશું. Init કિંમત, આ init val હોવો જોઈએ. પ્રારંભિક મૂલ્ય ડિફોલ્ટ કોઈપણ દ્વારા છે જ્યાં સુધી હું તમને એક પ્રારંભિક મૂલ્ય પ્રદાન કરતો નથી જે કિસ્સામાં તમે તે મૂલ્ય સાથે સૂચિ બનાવો છો અને ખાલી યાદી વિશેની અમારી ધારણાને લીધે આપણે યાદી ખાલી છે કે નહીં તે ચકાસવાની જરૂર છે કે કેમ તે તપાસવું એ મૂલ્ય છે કે નહીં પ્રારંભિક નોડમાં કંઈ નથી અથવા નથી. અમે ફક્ત તે સૂચિને લઈએ છીએ જે આપણે તરફ સંકેત કરી રહ્યા છીએ અને તે ખૂબ જ પ્રથમ મૂલ્યને જુઓ જે સ્વ ડોટ વેલ્યુ હશે અને પૂછશે કે તે કંઈ નથી. જો

તે કંઈ નથી, તે ખાલી છે. જો તે કંઈ નથી, તો તે ખાલી નથી. અહીં એક લાક્ષણિક વસ્તુ છે. આપણે કહીએ છીએ કે 11 બરાબર નોડ છે; આ એક ખાલી સૂચિ બનાવે છે કારણ કે તે કોઈ મૂલ્ય પ્રદાન કરતું નથી. તેથી, ડિફોલ્ટ પ્રારંભિક મૂલ્ય કોઈ પણ બનશે નહીં. જો હું કહું કે 12 એ નોડ 5 બરાબર છે તો તે 5 નોડ સાથે નોડ બનાવશે. તે સિંગલટન સૂચિ બનાવશે જે આપણે સામાન્ય રીતે આની જેમ પાચથોન માં લખીશું. જો હું પૂછીશ કે એલ 1 ખાલી છે, તો જવાબ સાચો હશે. જો હું પૂછીશ કે શું 12 ખાલી છે, તો જવાબ ખોટી હશે કારણ કે સ્વ ડોટ વેલ્યુ કંઈ નથી.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 03:50)

હવે, એકવાર અમારી પાસે સૂચિ છે કે આપણે શું કરવા ઈચ્છીએ છીએ તે ચાલાકી છે. પ્રથમ વસ્તુ જે આપણે કરવા માંગીએ છીએ તે સૂચિના અંતે મૂલ્ય ઉમેરે છે. જો સૂચિ પહેલાથી જ ખાલી હોય, તો આપણી પાસે એક નોડ હોય છે જે વેલ્યુ ન હોય અને આપણે તેને સિંગલટન નોડ, મૂલ્ય v સાથે સિંગલટન સૂચિ બનાવવા માંગીએ છીએ. તેથી, આપણે આને અહીંથી જવું છે, યાદ રાખો કે સિંગલટન નોડ, આપણે બધાની જગ્યાએ છીએ, અહીં અમારી પાસે કિંમત v છે તેથી તે આપણે કરવાની જરૂર છે. જો આપણે સૂચિના અંતિમ ભાગમાં છીએ અને અમે જાણીએ છીએ કે આપણે સૂચિના અંતિમ ભાગમાં છીએ તો આગામી વેલ્યુ કંઈ નથી તેથી આપણે શું કરવાની જરૂર છે તે નવું વેલ્યુ બનાવશે. અમે સૂચિના અંત સુધી જઈએ છીએ અને પછી અમે કોઈ નહીં પહોંચીએ છીએ. હવે, આપણે મૂલ્ય સાથે અહીં એક નવું ઘટક બનાવીએ છીએ અને આ ઘટક આને નિર્દેશિત કરીએ છીએ, આપણે નોડ વી સાથે એક નવું ઘટક બનાવીએ છીએ અને નવા નોડને આગળ ધપાવવા માટેના નોડના આગામી ફીલ્ડને સેટ કરો અને જો તે ન હોય તો પછી છેલ્લું મૂલ્ય, આપણે બાકીની યાદીને ફરીથી યાદ કરી શકીએ છીએ તે પછીના તત્વ પર શરૂ થતી નવી સૂચિ તરીકે આને ધ્યાનમાં લઈએ, આગામી તત્વ લઈએ અને તે માટે ફરી જોડવું.

(સ્લાઈડ સમયનો સંદર્ભ લો: 05:05)

આ આપણને એક ખૂબ સરળ રેકર્સિવ વ્યાખ્યા ઉમેરે છે તેથી, અમે જોડીએ છીએ અને અમે આ સૂચિમાં v ઉમેરવું છે. જો તે ખાલી છે, તો આપણે ફક્ત v ની કિંમત સુયોજિત કરી છે. તેથી, તે માત્ર એક નોડને વેલ્યુ ઓફ સાથે એક નોડમાં રૂપાંતરિત કરે છે, અન્યથા જો આપણે છેલ્લા નોડ પર છીએ જે સ્વ ડોટ આગળ છે તો પછી કંઈ નથી અમે કિંમત v સાથે એક નવો નોડ બનાવીએ છીએ અને નવા નોડ પર નિર્દેશ કરવા માટે અમે અમારી આગામી પોઇન્ટર સુયોજિત કરીએ છીએ, યાદ રાખો કે જ્યારે આપણે નવો નોડ બનાવીએ ત્યારે નવો નોડ આપમેળે અમારી init ફંક્શન દ્વારા આગામી કોઈ પણ સાથે બનાવી નથી. અમે હવે નવા નોડ બનાવશું જે v અને કંઈ જેવું દેખાય છે અને અમે તે નિર્દેશ કરવા માટે આગામી પોઇન્ટરને સેટ કરીશું અને અંતિમ વસ્તુ એ છે કે જો તે કોઈ નહીં હોય તો અમારી પાસે બીજું કંઈક બીજું છે. તેથી, આપણે તે આગળના તત્વ સ્વ ડોટ આગળ જઈએ છીએ અને તે પછીના તત્વના સંદર્ભમાં આપણે મૂલ્ય સાથે એપેન્ડ ફંક્શન પુનઃપ્રાપ્ત કરીએ v, આ ફરી યાદ આવવું કોલ છે. અમે ખાતરી કરવા માટે કે આ વિશ્લેષણાત્મક છે તેટલા પ્રમાણમાં સાવચેતી રાખવામાં આવી છે તેથી, આપણે આ કૌંસને કહીએ છીએ કે આપણે ઓબ્જેક્ટ સ્વ ડોટ આગળ લઈએ છીએ અને એપેન્ડને લાગુ પાડીએ છીએ કે ખરેખર પિથન આને યોગ્ય રીતે કરશે. આપણે વાસ્તવમાં કૌંસ મૂકવાની જરૂર નથી, આપણે ફક્ત સ્વ ડોટ આગળ ડોટ લખી શકીએ છીએ. એપેન્ડ વી અને પિથન એ યોગ્ય રીતે ડોટ ડોટ next ડોટ એપેન્ડ તરીકે બ્રેક કરે છે. તેથી, આ ડોટ જમણેથી લેવામાં આવે છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 06:23)

હવે, સૂચિના અંતમાં ફરી જવાની બદલે આપણે અંતના અંત સુધી તે અંત સુધી સ્કેન કરી શકીએ છીએ. આપણે એક લૂપ લખી શકીએ છીએ જે આ પોઇન્ટરને સરકાવવા સુધી ચાલે છે જ્યાં સુધી આપણે કોઈ

નોડ સુધી પહોંચતા નથી જેની આગળ કોઈ નથી. જો સૂચિ ખાલી હોય તો પહેલાં વેલની કિંમત ન બદલી શકીએ, અન્યથા અમે યાદીને સ્કેન કરીએ છીએ જ્યાં સુધી આપણે છેલ્લા ઘટક સુધી પહોંચતા નથી અને પછી એકવાર આપણે છેલ્લા ઘટક સુધી પહોંચીએ છીએ, જેમ કે પહેલાંના કિસ્સામાં આપણે નવું નોડ બનાવીએ અને છેલ્લું ઘટક બિંદુ બનાવીએ તે માટે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 06:53)

આ અમને એક એપેન્ડ આપે છે જે પુનરાવર્તિત છે. તેથી, અમે તેને કહીએ છીએ કે હું માત્ર તે સૂચક છે કે તે પુનરાવર્તન છે. પ્રથમ ભાગ એ જ છે જો વર્તમાન સૂચિ ખાલી હોય તો અમે ફક્ત v ની કિંમત નક્કી કરીએ છીએ અને અમે પાછા આવીએ છીએ, નહીં તો આપણે હવે સૂચિને નીચે જવું છે. અમે વર્તમાન નોડ પર નિર્દેશ કરવા માટે એક અસ્થાયી નિર્દેશકની સ્થાપના કરી છે અને તેટલો લાંબો સમય છે કે આગામી કંઈ નથી અમે આગામી મૂલ્યમાં `temp` સ્થાનાંતરિત રાખીએ છીએ. તેથી, આપણે ફક્ત લૂપ લખીએ છીએ જે કહે છે કે `temp dot` પછી કોઈ નહીં તો ફક્ત `temp` થી `temp dot` આગળ જવાનું ચાલુ રાખો. તેથી, ફક્ત `temp` સ્થાનાંતરિત રાખો. છેલ્લે જ્યારે આપણે આ બિંદુમાંથી આ બિંદુથી બહાર આવીએ છીએ ત્યારે આપણે જાણીએ છીએ કે `temp dot next none` છે. આ એવી શરત છે જેના હેઠળ આપણે લૂપથી નીકળી જઈએ છીએ. અમે પહોંચી ગયા છો, નોડ `temp` હવે વર્તમાન સૂચિમાં છેલ્લા નોડને નિર્દેશ કરી રહ્યું છે. આ બિંદુએ આપણે રિકરિવ કેસમાં બરાબર કરીએ છીએ કે આપણે મૂલ્ય v સાથે નવો નોડ બનાવીએ છીએ અને આ નવા નોડને આ છેલ્લો નોડ બિંદુ બનાવીએ છીએ. તેથી, અમે કોઈ નવી થી નવો નોડ પર કામચલાઉ નોટસ રીસેટ કરીએ છીએ. (સ્લાઇડ સમયનો ઉલ્લેખ કરો: 07:57) જો આપણે ઉમેરવાની ઇચ્છા ન રાખીએ, પણ આપણે દાખલ કરવા માંગીએ છીએ. હવે તે સામાન્ય રીતે દેખાય છે કે શામેલ કરવું સરળ હોવું જોઈએ, પરંતુ વાસ્તવમાં શામેલ કરવું થોડી મુશ્કેલ છે. તેથી, દાખલ કરીને અમારો અર્થ છે કે આપણે શરૂઆતમાં મૂલ્ય મૂકેલ છે. અમે અહીં એક નોડ મૂકવા માંગીએ છીએ જે v છે અને આ નિર્દેશક બનાવે છે. આ હવે આપણે શું કરવું છે તે છે. આની સાથે સમસ્યા એ છે કે આપણે એક નવો નોડ બનાવ્યાં પછી આપણે આ બિંદુ અહીં ન બનાવી શકીએ અને અહીં આ બિંદુએ નવા નોડ બિંદુને v 1 માં બનાવવામાં કોઈ સમસ્યા નથી, પરંતુ જો આપણે ઓલ ની કિંમત અથવા ઓબ્જેક્ટની અંદર પુનઃ નિર્ધારિત કરીએ તો આપણે સ્વની મૂલ્યને પુનઃ સોંપણી કરીએ છીએ તો આ એક સંપૂર્ણપણે અલગ ઓબ્જેક્ટ બનાવે છે. અમે આ જોયું જ્યારે અમે જોઈ રહ્યા છીએ કે કેવી રીતે પરિમાણો પસાર થાય છે અને નિર્વિકાર મૂલ્ય પસાર થાય છે. અમે જણાવ્યું હતું કે જો આપણે કંટેબલ વેલ્યુને ફક્શનમાં પસાર કરીએ છીએ, તો જ્યાં સુધી આપણે તે વસ્તુને ફરીથી સોંપણી નહીં કરીએ ત્યાં કાર્યની અંદર કોઈપણ પરિવર્તન વિધેયની બહાર પ્રતિબિંબિત થશે, પરંતુ જો આપણે ફક્શનમાં સૂચિ અથવા શબ્દકોશમાં ફરીથી સોંપણી કરીશું તો અમને એક નવી નકલ મળી જશે. અને તે પછી અમે જે ફેરફાર કરીએ છીએ તે બંધ છે. એ જ રીતે જો આપણે નવા નોડને નિર્દેશ કરવા માટે 1 અથવા સ્વયંને ફરીથી સોંપવું હોય તો આપણે ફક્શનમાં પસાર કરેલ પરિમાણ અને પાછું પામેલા પેરામીટર વચ્ચેનું જોડાણ ગુમાવશું. તેથી, આપણે સાવચેત રહેવું જોઈએ કે આ વાતને લીટી ન કરો. અમે જ્યાં 1 પોઇન્ટ માટે બદલી શકતા નથી. તો, આપણે આ સમસ્યાને કેવી રીતે મેળવી શકીએ? અમે એક નવું નોડ બનાવ્યું છે, અમે તેને લે બિંદુ બનાવવા માંગીએ છીએ, પરંતુ અમને આવું કરવાની મંજૂરી નથી, કારણ કે જો આપણે આમ કરીએ છીએ, તો પાયથોન જૂના એલમાંથી નવું એલ ડિસ્કનેક્ટ કરશે. તેથી, એક ખૂબ સરળ યુક્તિ છે. આપણે શું કરીએ છીએ તે આપણે નોડની ઓળખ બદલતા નથી, આપણે તેમાં જે ફેરફાર કર્યો છે તે બદલીએ છીએ. તેથી, હવે આપણે જાણીએ છીએ કે v 1 એ જૂના પ્રથમ નોડ છે અને v એ એક નવો પ્રથમ નોડ છે, પરંતુ અમે નવો પ્રથમ નોડને લી બિંદુ બનાવી શકતા નથી, તેથી આપણે મૂલ્યોનું વિનિમય કરીએ છીએ. તેથી, આપણે શું કરીએ છીએ આપણે v 1 દ્વારા v અને v દ્વારા v 1 ને બદલીએ છીએ. હવે, મૂલ્યો સ્વેપ થઈ ગયા છે અને તે પણ શું કરવું જોઈએ તે માટે આપણે પણ એવી જ વસ્તુ કરવી પડશે. તેથી, હવે હું પ્રથમ નોડ તરીકે વીને પોઇન્ટ કરું છું, પરંતુ હવે આપણે v 1 ને બાજુએ રાખ્યા છે જે એક ભૂલ છે. અમે હવે નવા નોડ અને જૂના બીજા નોડ માટે નવા નોડ બિંદુ માટે પ્રથમ નોડ બિંદુ બનાવવા જ જોઈએ. તેથી, આ પ્રકારના પ્લમ્બિંગ દ્વારા અમે જે ખાતરી કરી છે તે એ છે કે નવી સૂચિ

દેખાય છે જેમ આપણે વી 1 પહેલા વી શામેલ કર્યો છે, પરંતુ વાસ્તવમાં અમે વી અને વી 2 ની વચ્ચે એક નવો શામેલ કર્યો છે અને અમે ફક્ત લિંક્સને બદલી છે તે નવો નોડ બીજા છે અને પ્રથમ ન હોવા છતાં તે દેખાય છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 10:22)

અહીં શામેલ કરવા માટેનું કોડ છે. હંમેશની જેમ, જો તમારી પાસે ખાલી સૂચિ સામેલ હોય તો તે સરળ છે. આપણને ફક્ત કોઈને v માં બદલવાની જરૂર નથી. તેથી, ખાલી સૂચિ સાથે બંને રીતે દાખલ કરો અને તે જ રીતે વર્તે. અમે ખાલી સૂચિમાંથી સૂચિમાં જઈએ છીએ v. તે કોઈ બાબત નથી કે તમે શામેલ કરી રહ્યા છો અથવા શામેલ કરી રહ્યા છો. નહિતર, અમે આ નવો નોડ બનાવીએ છીએ અને ત્યારબાદ આપણે તે વર્તમાન નોડ વચ્ચેના મૂલ્યોની સ્વેપિંગ કરીએ છીએ જે સ્વયં તરફ સંકેત આપે છે, તે તે યાદીના મુખ્ય અને નવા નોડ છે. અમે કિંમતો વિનિમય; અમે આ ડોગ સાથે એકસાથે અસાધનમેન્ટનો ઉપયોગ કરીને સ્વ ડોટ વેલ્યુને નવા નોડ ડોટ વેલ્યુ અને એકસાથે નવા નોડ ડોટ વેલ્યુને સ્વ ડોટ વેલ્યુ પર સેટ કરી છે. અને તે જ રીતે અમે સ્વ ડોટ લઈએ છીએ જે આગળના નોડ તરફ સંકેત આપતો હતો અને તેને નવા નોડ અને નવા નોડ તરફ નિર્દેશિત કરવાને બદલે આપણે અગાઉ જેનું નિર્દેશન કરતા હતા તે નિર્દેશ કરવા જોઈએ. તેથી, નવા નોડ ડોટ આગળ સ્વ ડોટ આગળ છે. આ રીતે આપણે શામેલ કરીએ છીએ અને શામેલ કરીએ છીએ કે આપણે જોયું છે કે એન્ડએન્ડ કરતાં થોડું વધારે જટિલ છે કારણ કે પ્રારંભિક રીતે હેન્ડલ કરવાથી કે જેમાં સૂચિમાં સૂચિ અથવા સ્વયંચાલિત બિંદુઓ નિર્દેશ કરે છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 11:19)

જો આપણે નોડ કાઢી નાખવા માંગો છો તો શું? નોડને કાઢી નાખવા માટે અમે કેવી રીતે સ્પષ્ટ કરીએ છીએ? વેલ અમે તેને મૂલ્ય દ્વારા સ્પષ્ટ કરી છે, પરંતુ ચાલોઅમને લાગે છે કે તમે કાઢી નાખવા માંગો છો આ સૂચિમાં બીજા નોડ કહે છે. હવે, આપણે તેને કેવી રીતે કાઢી નાખીશું? ઠીક છે કે જેમ આપણે દાખલ કર્યું છે તેમ અમે કેટલીક ફરી પ્લમ્બિંગ અથવા ફરીથી કનેક્શન કરીશું. તેથી, અમે નોડ લે છે જે આપણે કાઢી નાખવા માંગીએ છીએ અને અમે ફક્ત તે લિંકને બનાવીએ છીએ જે v 2 ને બાયપાસ કરે છે. તેથી, અમે v 1 થી લીંક લઈએ છીએ અને તે સીધી રીતે v 3 તરફ નિર્દેશ કરે છે. તેથી, સારાંશમાં, તે બધા કાઢી નાખવાની જરૂર છે, કાઢી નાંખી નોડ પછી પોઇન્ટર સાથે રદ થયેલ નોડ પહેલાથી પોઇન્ટરને ફરીથી સોંપવાની જરૂર છે. તે વાસ્તવમાં ભૌતિક રીતે તે ઓબ્જેક્ટને મેમરીમાંથી દૂર કરે છે, પરંતુ તે ફક્ત તેને લિંક ઓવરનેથી દુર્ગમ બનાવે છે. અમે મૂલ્ય v પૂરી પાડીએ છીએ અને અમે પ્રથમ વિધીને દૂર કરવા માંગીએ છીએ. અમે પ્રથમ વી માટેની સૂચિને સ્કેન કરીએ છીએ. હવે નોંધ લો કે આ પ્લમ્બિંગની કાર્યપ્રણાલીમાં આપણે v3 ને નિર્દેશ કરવા માટે v 1 ની જરૂર છે. જો આપણે આગળના નોડને કાઢી નાખવા માંગીએ છીએ તો આપણે સારા આકારમાં છીએ કારણ કે આપણે આગામી ડોટ આગળ લઈ જઈ શકીએ છીએ અને તેને આગામી આગળના ભાગમાં દાખલ કરી શકીએ છીએ. તેથી, આપણે 1 પગલું આગળ જોવું જોઈએ. જો તમે પહેલેથી v 2 પર હોવ તો, આપણે પહેલાની વી 1 તરફ જઈ ગયા છીએ અને અમે v 1 પર પાછા જઈ શકતા નથી, જે રીતે અમે અમારી સૂચિ સેટ કરી છે, કારણ કે તે માત્ર આગળ જાય છે; અમે વી 1 પર પાછા જઈ શકતા નથી અને તેને બદલી શકીએ છીએ.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 12:38)

આપણે ખરેખર શું કરીશું તે આપણે આગામી મૂલ્યને જોઈને સ્કેન કરીશું. જો સ્વ ડોટ આગળ કોઈ વેલ્યુ v છે કે જો પછીનો નોડ કાઢી નાખવાનો હોય તો આપણે કહી શકીશું કે વર્તમાન નોડની આગામી નળ આગામી નોડ નથી પરંતુ આગામી નોડની આગામી છે. તેથી, સ્વ ડોટ આગળ સ્વ ડોટ આગળ ડોટ આગામી માટે ફરીથી સોંપવામાં આવે છે - આગામી નોડ બાયપાસ પહેલાની જેમ જ સામેલ થવું તે પહેલાં જ આપણે સાવચેત રહેવું જોઈએ કે જો આપણે ખરેખર યાદીમાં પ્રથમ મૂલ્ય કાઢી નાખવી પડશે.

(સ્લાઈડ ટાઇમ નો ઉલ્લેખ કરો: 13:12)

જો તમે સૂચિમાં પ્રથમ વેલ્યુ કાઢી નાંખવા માંગતા હો, જેમ કે આપણે શામેલ કર્યું હોય તો કુદરતી વસ્તુ હશે, હવે કહે છે કે 1 એ સૂચિમાં બીજો મૂલ્ય નિર્દેશ કરે છે, પરંતુ અમે અહીં બિંદુ છે કારણ કે જો આપણે નોડને ફરીથી સોંપવું છે જે 1 નો નિર્દેશ કરે છે તો તે એક નવું ઓબ્જેક્ટ બનાવશે અને તે આપણે પસાર કરેલ પરિમાણ અને જે વસ્તુ આપણે પાછા મેળવીએ તે વચ્ચેના જોડાણને તોડશે. અમે એ જ યુક્તિનો ઉપયોગ કરીએ છીએ આપણે શું કરીએ તે પછીના નોડથી વેલ્યુ v 2 ની નકલ કરીએ છીએ અને પછી ... તો આપણે આ વેલ્યુને અહીંથી નકલ કરીએ છીએ અને ત્યારબાદ આપણે v 2 કાઢી નાંખીએ છીએ. તેથી, આપણે પ્રથમ નોડને રદ કરવા માંગીએ છીએ, અમને મંજૂરી નથી. પ્રથમ નોડને રદ કરવા માટે, કારણ કે આપણે 1 નો ઉલ્લેખ કરી શકતા નથી. તેથી, તેના બદલે અમે બીજો નોડમાં મૂલ્ય લઈએ છીએ જે v 2 હતી, અહીં કોપી કરો અને પછી ડોળ કરો કે આપણે ત્રીજાને પ્રથમ નોડ બિંદુ બનાવીને v 2 રદ કર્યો.

(સ્લાઈડ સમયનો સંદર્ભ લો: 14:07)

અહીં કાઢી નાંખો કાર્યનો એક ભાગ છે. સૌ પ્રથમ, જો આપણે વી માટે શોધી રહ્યા હો અને પછી અમે તેને શોધી શકતા નથી. તેથી, આ કોડમાં માફ કરશો, તેને x કહેવામાં આવે છે. તેથી, જો તમે ઇચ્છો તો આ મૂલ્ય x કાઢી રહ્યું છે. જો આપણે કહીએ કે સૂચિ ખાલી છે, તો દેખીતી રીતે, આપણે તેને રદ કરી શકતા નથી કારણ કે કાઢી નાંખો કહે છે કે આનું મૂલ્ય છે ... મૂલ્ય x સાથે નોડ તો તેને કાઢી નાખો. જો તે ખાલી હોય તો અમે કશું કરીએ નહીં; અન્યથા જો આ સ્વ ડોટ વેલ્યુ x છે પ્રથમ નોડ કાઢી નાખવાનો છે. પછી જો ત્યાં માત્ર 1 નોડ હોય, તો આપણે એક્સથી ખાલી થઈ જઈએ છીએ, આ સરળ છે. જો કોઈ આગળનો નોડ અધિકાર ન હોય તો, જો અમારી પાસે માત્ર સિંગલટન હશે તો અમે ફક્ત મૂલ્યને કોઈ જ નહીં સુયોજિત કરીશું અને આપણે પૂર્ણ કરીશું. આ સરળ કેસ છે, પરંતુ જો તે પ્રથમ નોડ નથી, તો તેનો અર્થ, તે પહેલો નોડ છે અને આ સૂચિમાં એકમાત્ર નોડ નથી, તો આપણે શું કરીએ છીએ તે આપણે કરીએ છીએ તે પહેલાં આપણે શું કર્યું. અમે આગામી મૂલ્યની નકલ કરીએ છીએ. અમે ડોળ કરવો કે અમે બીજા નોડને કાઢી નાખીએ છીએ. તો, આપણે બીજા વેલ્યુને પ્રથમ વેલ્યુમાં નકલ કરીએ છીએ અને આપણે બાયપાસ કરીને આગળના નોડને રદ કરીએ છીએ. આ બાયપાસ છે આ કાર્યનો એક ભાગ છે; આ કપટી ભાગ છે જે તમે પ્રથમ મૂલ્યને કેવી રીતે રદ કરો છો? જો તે માત્ર 1 મૂલ્ય છે, તો તેને કંઈ ન બનાવો; જો નહિં, તો બીજા નોડને બીજા નોડની નકલ કરીને પ્રથમ નોડને બાયપાસ કરો.

(સ્લાઈડ સમયનો ઉલ્લેખ કરો: 15:24)

અને જો આ કિસ્સો ન હોય તો આપણે ફક્ત નીચે જઈએ છીએ અને કાઢી નાખવા માટે પ્રથમ x શોધીએ છીએ. અમે પ્રારંભ કરીએ છીએ ... આ અમારી પુનરાવર્તનની જેમ છે. અમે સ્વ તરફ સંકેત કરવાનું શરૂ કરીએ છીએ અને જ્યાં સુધી આપણે સૂચિના અંત સુધી પહોંચ્યા નહીં જો અમને લાગે કે આગામી મૂલ્ય x છે અને તે પછી આપણે તેને બાયપાસ કરીએ છીએ અને જો તમે સૂચિના અંત સુધી પહોંચી ગયા છો, તો અમને તે મળ્યું નથી, અમે કંઈ નથી, અમે માત્ર પાછા છે આ કિસ્સામાં તે ઉમેરાની જેમ નથી કે જ્યાં અમે સૂચિના અંતમાં પહોંચી ગયા છો, ત્યાં અમને અહીં ઉમેરવાની જરૂર છે, જો આપણે સમયની આગળ પહોંચતા સમય સુધી આગળ નહીં મળે તો, પૂર્ણ કરવા માટે કશું જ નથી.

(સ્લાઈડ સમયનો ઉલ્લેખ કરો: 15:54)

તેથી, પૂર્ણતા માટે, અહીં પૂર્ણ કાર્ય છે, આ પહેલી સ્લાઈડ જે અમે જોયેલી છે જે કિસ્સો હોય ત્યારે કાઢી નાખવા માટેની કિંમત પ્રથમ નોડમાં હોય છે અને આ બીજું કેસ છે આપણે કાઢી નાખવા માટે પ્રથમ x માટે જોઈ રહેલી યાદી નીચે જઈએ છીએ.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 16:09)

જેમ જેમ એપેન્ડને બન્ને પુનરાવર્તિત અને પુનરાવર્તિત કરી શકાય છે, આપણે તે ફરીથી રદબાતલ કાઢી નાખી શકીએ છીએ, જો તે પ્રથમ નોડ છે, તો આપણે તેને બીજા મૂલ્યને પ્રથમ અને બાયપાસ કરીને વિશેષ રૂપે નિયંત્રિત કરીએ છીએ. જેમ આપણે પહેલાં કર્યું નહિતર અમે આગળના નોડ પર જ નિર્દેશ કરીએ છીએ અને આગળના નોડને પૂછો, પછીની નોડમાં શરૂ થતી સૂચિ, જે સામાન્ય રીતે સૂચિની પૂંછડી કહેવાય છે, તે પોતાની જાતે v કાઢી નાખવા માટે. આમાં યાદ રાખવું એ જ વસ્તુ છે કે જો આપણે સૂચિના અંત સુધી પહોંચીએ અને આપણે છેલ્લી નોડ કાઢી નાંખો. તે તારણ કાઢે છે, કાઢી નાખવા માટેની કિંમત v અહીં છે. તો, આપણે અહીં આવીએ છીએ અને પછી આપણે તેને રદ કરીએ છીએ. આપણે શું કરીશું તે કોઈ વેલ્યુ શોધતું નથી, કારણ કે જ્યારે આપણે તેને અહીંથી કાઢી નાંખીએ છીએ, તેમ છતાં આપણે એક સિંગલટોન વ્યુ લઈએ છીએ અને સિંગલટોનમાંથી v કાઢી નાખીએ છીએ અને કંઈ પણ બનાવીશું નહીં. તેથી, આ મૂળભૂત કેસ છે, જો આપણે પાછલી યાદમાં કાઢી નાંખીએ છીએ જ્યારે આપણે છેલ્લી નોડમાંથી કાઢી નાખીએ છીએ, તે આ પ્રમાણે છે કે આપણે મૂલ્ય વી સાથે સિંગલટોન સૂચિમાંથી કાઢી નાખીએ છીએ અને અમને કોઈ મૂલ્ય ન બનાવવાની મંજૂરી નથી. અંત આપણે માત્ર ત્યારે જ તપાસ કરવી જોઈએ જ્યારે આપણે આગલી વસ્તુ બનાવીએ છીએ જો આપણે આગામી મૂલ્ય કાઢી નાંખીએ અને તે વેલ્યુ કંઈ નહીં હોય તો આપણે યાદીમાંથી તે વસ્તુ દૂર કરવી જોઈએ. તેથી, આ એકમાત્ર કાવતરાબાજ વસ્તુ છે કે જ્યારે આપણે ફરી યાદ આવવું હોય તો અમે તમને કાઢી નાખીએ તે પછી તમારે સાવચેત રહેવું જોઈએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 17:32)

આ ભાગ પહેલાનો ભાગ છે અને હવે આ ફરી યાદ આવવું ભાગ છે. તેથી, ફરી યાદ આવવું ભાગ એકદમ સીધા છે. તેથી પ્રથમ ભાગ એ છે કે જ્યારે આપણે સૂચિમાંથી પ્રથમ એલિમેન્ટ કાઢી નાંખો, પરંતુ ફરી યાદ આવવું ભાગ આપણે તપાસીએ છીએ કે સ્વ ડોટ આગળ કોઈની બરાબર નથી, તો આપણે ફરી યાદ કરીએ છીએ કે તે સારું છે. તેથી, આ કાઢી નાંખો કોલ છે હવે, કાઢી નાંખવાનું પૂર્ણ થયા પછી અમે તપાસીએ છીએ કે આગલી કિંમત વાસ્તવમાં કોઈ નહીં બની. શું અમે ખરેખર છેલ્લા નોડમાં અંત કર્યો છે અને છેલ્લા નોડને કાઢી નાખીએ છીએ? જો એમ હોય, તો આપણે તેને દૂર કરી શકીએ છીએ, આપણે ક્યાં તો સ્વ ડોટ લખી શકીએ છીએ, આગામી સ્વ ડોટ આગળ છે, આગામી ડોટ આગળ છે અથવા આપણે સ્વયં ડોટ પણ લખી શકીએ છીએ જે કોઈકની બરાબર છે જે સંભવતઃ કહીને ક્લીનર રસ્તો છે કારણ કે તે ફક્ત છેલ્લા નોડ પર થાય છે. તેથી, તમે આ નોડને છેલ્લા નોડ કરો છો. યાદ રાખો કે જો આગામી નોડ કંઈ નથી, તો તે આગામી કોઈ પણ હોવું જોઈએ નહીં. આ એક જ અસર છે: સ્વ ડોટ આગામી ડોટ આગામી કંઈ પણ હોવું જોઈએ નહીં. તેથી, આપણે સીધી સોફ્ટ ડોટને આગામી કોઈપણ બરાબર ન આપી શકીએ અને તે મૂળભૂત રીતે આ નોડને છેલ્લા નોડ બનાવશે. રિકર્સિવ ડિલિટ વિશે યાદ રાખવા માટેની એકમાત્ર વસ્તુ છે જ્યારે આપણે સૂચિના અંત સુધી પહોંચીએ છીએ અને આ સૂચિને કાઢી નાંખીએ છીએ તો આ કંઈ બનશે નહીં તો અહીં સૂચિને સમાપ્ત કરવું જોઈએ અને આ નોડ દૂર કરવું જોઈએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 18:34)

છેલ્લે ચાલો એક યાદી છાપવા માટે એક કાર્ય લખીએ. તેથી, આપણે શું ચાલી રહ્યું છે તેનો ટ્રેક રાખી શકીએ છીએ. આપણે ફક્ત પાચથોન ની યાદી બનાવીશું અને પછી પાચથોન ની યાદીમાં str વાપરીશું. તેથી, આપણે આપણા સૂચિમાંના મૂલ્યોમાંથી પાચથોન યાદી બનાવવા માંગીએ છીએ. તેથી, આપણે પ્રથમ અમારી સૂચિને પ્રારંભ કરીએ છીએ કે અમે ખાલી યાદી માટે ઉત્પાદન કરવા જઈ રહ્યા છીએ. જો

આપણી સૂચિ, નોડની પાસે કંઈ નથી તો આપણે ખાલી યાદીની સ્ટ્રિંગ વેલ્યુ પાછી આપીએ છીએ, નહિતર અમે સૂચિમાંથી નીચે જઈએ છીએ અને અમે દરેક કાર્યને એન્ડેન્સ ફંક્શનનો ઉપયોગ કરીને ઉમેરીએ છીએ. તેથી, અમે દરેક મૂલ્યને સમાવીએ છીએ જે દરેક નોડમાં સંગ્રહિત છે અને આ પ્રક્રિયામાં પાચથોન ની યાદી તૈયાર કરી છે અને છેવટે, તે સૂચિની સ્ટ્રિંગ વેલ્યુ પરત આવે છે. ચાલો આપણે કેટલાક પાચથોન કોડને જોઈએ અને જુઓ કે કેવી રીતે તે વાસ્તવમાં કામ કરે છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 19:24)

અહીં આપણી પાસે કોડ છે જે સ્લાઇડ્સમાં અમે જે કર્યું તે બરાબર પ્રતિબિંબિત કરે છે. અમે બંને જોડો અને કાઢી નાંખો માટે આવર્તક સંસ્કરણોનો ઉપયોગ કરવાનું પસંદ કર્યું છે. તેથી, અમે આ પ્રારંભિક આરંભથી શરૂ કરીએ છીએ જે પ્રારંભિક મૂલ્ય ડિફોલ્ટ રૂપે કંઈ ન હોવું અથવા અન્યથા વીને દલીલ તરીકે પ્રદાન કરે છે.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 19:44)

ત્યારબાદ તે તપાસ કરે છે કે સ્વયં ડોટ વેલ્યુ કંઈ નથી, આપણે સ્લાઇડમાં વધુ કોમ્પેક્ટ સ્વરૂપે લખ્યું છે તે કહીને ફક્ત સ્વયં ડોટ વેલ્યુને બરાબર વળે છે, પણ આપણે તેને વિસ્તૃત કર્યું છે અહીં એક if સ્ટેટમેન્ટ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 19:56)

હવે, આ એપેન્ડ ફંક્શન છે. તો, ફક્ત તપાસ કરો કે જો વર્તમાન નોડ ખાલી છે તો તે અહીં મૂકી દેશે નહિ તો તે નવો નોડ બનાવશે ... જો આપણે છેલ્લા નોડ સુધી પહોંચ્યા છીએ તો તે નવા નોડ બનાવે છે અને નવા નોડને છેલ્લા નોડ બિંદુ બનાવે છે, અન્યથા તે પુનરાવર્તિત રીતે જોડાય છે પછી આપણી પાસે આ insert ફંક્શન અહીં છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 20:29)

આ શામેલ કાર્ય: ફરીથી જો તે ખાલી હોય તો તે માત્ર એક જ સૂચિ બનાવે છે અન્યથા તે નવો નોડ બનાવે છે અને પ્રથમ નોડ અને નવા નોડને એક્સચેન્જ કરે છે. તેથી, અહીં આ ચોક્કસ વસ્તુ તે સ્થાન છે જ્યાં અમે આ બનાવીએ છીએ, પોઇન્ટરોને સ્વેપ કરો જેથી સ્વ પોઇન્ટ શું બદલાતું નથી, પરંતુ અમે નવા નોડ અને પ્રથમ નોડની પુનઃ ગોઠવણી બનાવીએ છીએ. તેથી, નવો નોડ બીજા નોડ બની જાય છે અને પ્રથમ નોડ પાસે હવે મૂલ્ય છે જે આપણે હમણાં ઉમેર્યું છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 21:02)

છેલ્લે, આપણે ફરી યાદ આવતાં કાઢી નાંખો. તેથી, ફરી યાદ આવવું ફરી કહે છે કે જો સૂચિ ખાલી હોય તો આપણે કંઈ પણ કરી શકીએ નહીં, અન્યથા જો પ્રથમ મૂલ્ય કાઢી નાખવામાં આવે તો આપણે સાવચેત રહેવું જોઈએ અને અમારે બીજા નોડની કોપી કરીને બીજા મૂલ્યને કાઢી નાખવાની જરૂર છે. પ્રથમ અને છેવટે, જો તે કિસ્સો ન હોય તો આપણે ફક્ત રિફ્રેશબલ ડિલિટ કરીએ છીએ, પરંતુ પછી જ્યારે આપણે ડિલિટ સમાપ્ત કરીએ છીએ ત્યારે, આપણે અકસ્માતે તેને બનાવ્યું હોય તો યાદીના અંતે ખોટી ખાલી નોડને કાઢી નાખવી પડશે. તેથી, અહીંની આ 2 લીટીઓ ફક્ત સુનિશ્ચિત કરે છે કે અમે સૂચિના અંતમાં ખોટી ખાલી નોડ છોડીએ નહીં. અને છેવટે, આપણી પાસે આ સ્ટ્ર ફંક્શન છે જે આપણી કિંમતોમાંથી પાચથોન યાદી બનાવે છે અને છેવટે તે સૂચિનું સ્ટ્રિંગ પ્રતિનિધિત્વ આપે છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 21:54)

જો આપણે આને આયાત કરીને ચલાવીએ છીએ, તો આપણે કહી શકીએ છીએ, ઉદાહરણ તરીકે, તે મૂલ્ય 0 સાથેની સૂચિ છે અને જો આપણે પ્રિન્ટ એલ કહીએ તો આપણે આ પ્રતિનિધિત્વ 0 મેળવીશું, આપણે દાખલા તરીકે, આને લૂપમાં મૂકો અને હું શ્રેણી 1 માં 11 માં કહીશ, 1

(સ્લાઇડ ટાઇ નો ઉલ્લેખ કરો: 22:34)

અને પછી જો આપણે આ બિંદુએ 1 પ્રિન્ટ કરીએ તો આપણને પહેલાથી 0 થી 10 મળશે. (સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 22:40) હવે આપણે કહી શકીએ છીએ એલટ ડીટોટ 4 દાખલા તરીકે અને આપણે છાપીએ છીએ 4 પછી 4 ની રચના થાય છે. જો આપણે કહીશું કે '1' ડોટ શામેલ કરો 12 અને છાપી તો, 12 શરૂ થશે. તેથી, તમે તપાસ કરી શકો છો કે આ કામ કરે છે નોંધ લો કે આપણે આ ખાલી કૌંસ મેળવી રહ્યા છીએ, આ રીટર્ન મુલ્ય છે. તેથી, જ્યારે અમે આ વળતર લખ્યું ત્યારે અમે ખાલી દલીલ સાથે લખ્યું હતું. અને પછી અમે આ ખાલી ટુપલ મેળવીએ છીએ, આપણે કંઈ સાથે વળતર લખી શકીએ છીએ અને તે પછી આ મજાની રીટર્ન મુલ્ય દર્શાવશે નહીં, પરંતુ વાસ્તવમાં શું મહત્વનું છે કે આપણી સૂચિનું આંતરિક પ્રતિનિધિત્વ યોગ્ય રીતે બદલાયું છે જે આપણે લખ્યું છે .

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રોફેસર. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટીટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
અઠવાડિયું - 07 વ્યાખ્યાન - 04 શોધ વૃક્ષો

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 00:02)

વપરાશકર્તા વ્યાખ્યાયિત ડેટાનું આખરી ઉદાહરણ તરીકે માળખું અમે દ્વિસંગી શોધ ક્રીઓ પર જોવા મળશે. અમે એક એવી પરિસ્થિતિને જોઈ રહ્યા છીએ કે જ્યાં આપણે સઘન રીતે ગતિશીલ ડેટા જાળવી રાખવાની જરૂર છે, યાદ રાખો કે સોર્ટિંગના બાય પ્રોડક્ટ્સમાંથી એક એ છે કે આપણે દ્વિસંગી શોધનો ઉપયોગ મૂલ્ય શોધવા માટે કરી શકીએ છીએ, પરંતુ જો સોર્ટ કરી શકીએ તો દ્વિસંગી શોધનો ઉપયોગ કરી શકાય છે માહિતી એકવાર અને બધા માટે અને તે સોર્ટ ક્રમમાં રાખવા જો ડેટા ગતિશીલ બદલાતી રહે છે પછી દ્વિસંગી શોધનો દુરુપયોગ કરવા માટે માહિતી કે જે કાર્યક્ષમ નથી આશરો રાખવો પડશે. ધારો કે અમારી પાસે વસ્તુઓની શ્રેણી છે અને વસ્તુઓ સમયાંતરે દાખલ કરવામાં આવે છે અને કાઢી નાખવામાં આવે છે, જેમ કે આપણે ઢગલાઓ સાથે જોયું, દાખલા તરીકે, જો આપણે સોર્ટ કરેલ સૂચિને જાળવી રાખવાનો પ્રયાસ કરીએ અને પછી ઇન્વેર્સર્સનો ટ્રેક રાખીએ તો દરેક કેસ દાખલ કરો અથવા કાઢી નાખો, આ કિસ્સામાં હુકમ અને સમય લેવો અને તે ખર્ચાળ પણ રહેશે. જો કે, તે તારણ આપે છે કે આપણે એક વૃક્ષ જેવા કે માળખું અથવા એક વૃક્ષની માળખું જેમ કે પ્રાધાન્ય કતારમાં જઈએ છીએ તે એક ઢગલા પર ખસેડી શકે છે અને તે શોધ અને શોધ સાથે પણ અસરકારક રીતે કાઢી નાંખો.

(સ્લાઇડ સમયનો સંદર્ભ લો: 01:11)

ડેટા માળખું જે આપણા ધ્યાનમાં છે તે દ્વિસંગી શોધ વૃક્ષ તરીકે ઓળખાય છે. તેથી, બાઈનરી શોધ વૃક્ષમાં આપણે દરેક મૂલ્યની બરાબર એક નકલ રાખીએ છીએ. તે સમૂહની જેમ છે કે આપણે ડુપ્લિકેટ્સ ન રાખીએ અને મૂલ્યો દરેક નોડ માટે નીચે પ્રમાણે સંગઠિત થાય છે જે વર્તમાન નોડ મૂલ્ય કરતા નાની હોય તે તમામ મૂલ્ય ડાબે છે અને વર્તમાન નોડ મૂલ્ય કરતાં મોટું મૂલ્ય જમણી બાજુ છે. અહીં બાઈનરી શોધ વૃક્ષનું ઉદાહરણ છે, તમે દાખલા તરીકે તપાસ કરી શકો છો કે રૂટ 5 ની ડાબી તરફ, આપણી પાસે બધા મૂલ્યો 1, 2 અને 4 છે, જે 5 કરતા નાની છે અને 5 ની જમણી બાજુ છે, અમારી પાસે મૂલ્યો 8 અને 9 જે મોટા છે, હવે આ ફરી યાદ આવવું મિલકત છે. દાખલા તરીકે, જો તમે નીચે જાઓ, જો તમે નોડ લેબલને બે જોશો તો પછી તેની પાસે કિંમતો 1 અને 4 છે, કારણ કે 1 2 કરતા નાની છે, 1 એ 2 ની ડાબી બાજુ છે અને 4 થી 2 કરતાં મોટી છે, 4 એ 2 ની બરાબર. એ જ રીતે, જો આપણે નોડ 8 પર નજર રાખીએ, તો તેની પાસે માત્ર 9 ની નીચે એક જ મૂલ્ય છે અને તેથી, તેમાં કોઈ ડાબે બાળક નથી, પરંતુ 9 ની જમણી પેટા-રેખામાં છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 02:14)

અમે બાયનરી સર્ચ ટાઈલ્ડ નો ઉપયોગ કરીને નોડોનો ઉપયોગ કરી શકીએ છીએ, જેમ કે યાદી સિવાય અમે યુઝર વ્યાખ્યાયિત યાદીઓ માટે કર્યું છે અને એક બાઈનરી સર્ચ ટ્રીમાં આપણી પાસે કિંમત અને આગામી પોઇન્ટર છે, જે દરેક નોડ નીચે બે મૂલ્યો ધરાવે છે જે સંભવિત ડાબેરી બાળક અને જમણી બાજુ છે બાળક. તેથી, દરેક નોડમાં હવે ત્રણ વસ્તુઓનો સમાવેશ થાય છે જે ડાબા બાળક અને જમણા બાળકને સંગ્રહિત કરવામાં આવે છે. જો આપણે તે જ ઉદાહરણ જોઈએ કે આપણી પાસે 4 ની જમણી બાજુ છે, તો રૂટ નોડ 5 ને ગાંઠો 2 અને 8 સાથે નિર્દેશક હશે, નોડ 2 પાસે ગાંઠો 1 અને 4 નો પોઇન્ટર હશે,

આ હવે શું છે પણ ગાંઠો કહેવાય છે. તેથી, તેઓને કોઈ બાળકો નથી. તેમના ડાબા અને જમણા પોઇન્ટરો હશે નહીં તે દર્શાવે છે કે તે દિશામાં કંઈ નથી. એ જ રીતે, 8 ને તેના ડાબા નિર્દેશક તરીકે કોઈ નહીં મળ્યું કારણ કે તેની પાસે કોઈ ડાબે બાળક નથી અને જમણી નિર્દેશક 9 છે અને નોડ સાથે નોડ ફરીથી બે નથી કારણ કે તે પણ નોડ છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 03:10)

હવે, તે ચાલુ થશે કે આપણે આ પ્રતિનિધિત્વને વિસ્તૃત કરવા માંગીએ છીએ જેથી તે રિકર્સિવ પ્રસ્તુતિઓ માટે સારી રીતે શોષણ કરે. તેથી, આપણે શું કરીશું કે આપણે ફક્ત મૂલ્ય સાથે વૃક્ષને સમાપ્ત નહીં કરી શકીએ અને બે પોઇન્ટરો કોઈ નહીં તો તમે બધા ક્ષેત્રો સાથે ખાલી નોડસની એક સ્તર ઉમેરશો નહીં. આની સાથે ખાલી વૃક્ષ ખાલી ખાલી નોડ અને પાંદડાની હશે નહીં. નોડ કે જેનું કંઈ મૂલ્ય નથી અને તે બન્ને બાળકો ખાલી ગાંઠો હશે. તે સીધું જ નહીં, કેમ કે તે ડાબે અને જમણે પોઇન્ટર છે તે વાસ્તવમાં ખાલી બાળકોને મદદ કરશે. તેથી, તે રિકર્સિવ કાર્યો લખવાનું સરળ બનાવે છે અને જો આપણે આ ન કાર્ય હોય તો તે ફરીથી આવર્તક વિધેયો સીધી અમલમાં મૂકવા માટે થોડી કઠીન છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 03:56)

ફક્ત અમારા વૃક્ષનું માળખું કેવી રીતે બદલાય છે તે સમજવા માટે તે જ ઉદાહરણ પહેલાનું માળખું છે. અહીં, નોંધ લો કે પણ નોડમાં પણ નોડસ પાસે મૂલ્ય છે અને બંને બાળક પોઇન્ટરો ડાબે અને જમણે સીધી જ નહીં. તેથી, આપણે આને બદલવા ઈચ્છીએ છીએ, આપણે શું કરવા માંગીએ છીએ તે આપણે આને ખાલી નોડ નીચે દરેક જગ્યાએ દાખલ કરવા માંગીએ છીએ, જ્યાં આપણે કોઈ નહીં જુઓ, અમે નોડ શામેલ કરવા અને બહાર કાઢવા માંગીએ છીએ. આ અલબત્ત, વૃક્ષને થોડો વધે છે, પરંતુ તે પછી આ કિંમત એટલી જ નથી કે જે તમે ગણતરી કરી શકો. તેથી, જો આપણે આમ કરીએ તો આપણે એક નવું ટ્રી મેળવીએ છીએ જે આ જમણા ખૂણે છે

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 04:28)

દરેક પાંદડાની નોડ નીચે આપણે સામાન્ય રીતે કોઈ પણ પોઇન્ટર ધરાવતું નથી, જે સૂચવે છે કે પાથનો અંત આવ્યો છે, અમે સ્પષ્ટપણે એક વધારાનો નોડ ઉમેરીશું જે ત્રણેય ક્ષેત્રોમાં નથી અને તે ચાલુ થશે કે પછીથી આપણા પ્રોગ્રામિંગને સાફ કરવા માટે તે ખૂબ ઉપયોગી છે. તેથી, આ પ્રતિનિધિત્વ છે કે આપણે ટ્રિસંગી શોધ વૃક્ષ માટે ઉપયોગ કરીશું દરેક નોડમાં ત્રણ પોઇન્ટર અને પણ પર અમારી પાસે ખાલી નોડસની એક વધારાની સ્તર છે જે ત્રણેય મૂલ્યો સાથે કોઈ નહીં, કોઈ નહીં, કોઈ નહીં.

(સ્લાઇડ સમયનો સંદર્ભ લો: 04:58)

અહીં મૂળભૂત ક્લાસ ટ્રી છે. તેથી, પહેલાં આપણી પાસે એક init ફંક્શન છે જે પ્રારંભિક મૂલ્યો લે છે જે મૂળભૂત રીતે કંઈ નથી આ init કાર્ય નીચે પ્રમાણે કાર્ય કરે છે, આપણે પ્રથમ મૂલ્યને ઇનિટિયલ તરીકે સુયોજિત કરી શકીએ છીએ જે મૂલ્ય હોય તો કંઈ ન હોઈ શકે, તો પછી આપણે એક નોડ સાથે એક વૃક્ષ બનાવીએ છીએ, જેમાં આપણે આ ખાલી ગાંઠો બનાવવો પડશે. હવે, નોંધ લો કે જો આપણે પાછા જઈએ અને આપણે કોઈ મૂલ્ય ન આપીએ. તેથી, કદાચ આ કેસને જોવું વધુ સારું છે જો આપણે કોઈ મૂલ્ય ન આપીએ તો આપણે એક વૃક્ષ સાથે અંત પામીએ છીએ. તેથી, આ આપણું ખાલી છે. પ્રારંભિક મૂલ્ય એ કંઈ નથી કે આપણે આ વૃક્ષ મેળવીએ જો init મૂલ્ય કંઈ ન હોય તો આપણને એક વૃક્ષ મળે છે જેમાં આપણે મૂલ્ય v મુક્યું છે અને પછી આપણે ડાબી બાજુએ જમણી પોઇન્ટરમાં આ બન્ને પોઇન્ટરને કોઈ નહીં, કોઈ નહીં. તેથી, આ એક વૃક્ષ છે જેમાં ચોક્કસ મૂલ્ય હશે. તેથી, તેના આધારે કે init કિંમતો કોઈ નહીં અથવા નહીં તો આપણે ત્રણ નોડો સાથે બે ડમી ગાંઠો અથવા ફક્ત ખાલી ખાલી નોડને દર્શાવતી

એક ખાલી નોડ સાથેના એક વૃક્ષને સમાપ્ત કરે છે. તેથી, આપણી પાસે આ ફંક્શન છે તે પહેલાં આપેલ છે જે મૂળભૂત રીતે તપાસ કરે છે જો મૂળ્ય કંઈ નથી, જો હું કોઈ વૃક્ષ પર નજર શરૂ કરું અને ખૂબ પ્રથમ નોડ કહે છે કે પછી તે વૃક્ષ ખાલી નથી, તો તે ખાલી નથી.

(સ્લાઇડ સમયનો સંદર્ભ લો: 06:15)

ચાલો આપણે પહેલા વૃક્ષના મૂલ્યોને વ્યવસ્થિત રીતે અન્વેષણ કરવાના માર્ગ પર નજર કરીએ. આને ટ્રાવર્સલ્સ અને એક ટ્રાવર્સલ કહેવાય છે જે દ્વિસંગી શોધ વૃક્ષ માટે ખૂબ જ ઉપયોગી છે એક ઇનરડર ટ્રાવર્સલ છે. તેથી, એક ઇનરવર્ડ ટ્રાવર્સલ શું કરે છે તે છે કે તે પહેલાં ડાબી બાજુની શોધ કરે છે. તેથી, તે પહેલીવાર આક્રમણને ફરીથી આક્રમણ કરશે, પછી તે પ્રદર્શિત કરશે અને પછી તે જમણી બાજુએ શોધશે. જો તમે કોડ જોશો તો તમે જોઈ શકો છો કે જો વૃક્ષ ખાલી ન હોય તો તમારે પહેલાં ડાબા સ્વ વૃક્ષની અંદરની બાજુમાં વર્તુળ કરો છો, પછી તમે વર્તમાન નોડમાં મૂલ્ય પસંદ કરો છો અને પછી તમે સ્વયં વૃક્ષના આક્રમણમાં વર્તુળ કરો છો અને તે કિંમતો યાદી પેદા કરે છે તેથી, જો આપણે આ પગલું પગલું દ્વારા ચલાવો, 5 જો આપણે તે પર પહોંચીએ તો પ્રથમ કહેવું છે કે ડાબી બાજુના અંતર્દેશીય ટ્રાવર્સલલ કરો. અમે બેથી નીચે આવીએ છીએ તે તુચ્છ ઝાડ પર ફરી નથી. તેથી, ફરીથી આપણને એક આક્રમણનું વર્તન કરવું પડશે. તેથી, અમે જઈએ છીએ કે તે બાકી છે અને હવે જ્યારે આપણી પાસે એક હોય અને જ્યારે એકની અંદરની બાજુમાં સંકળાયેલ હોય ત્યારે તેમાં એક બાળક બાકી છે જે ખાલી છે અને પછી તે સાચું બાળક ખાલી છે. તેથી, આ એક ઉત્પન્ન કરે છે હવે હું પાછા આવું છું અને હું નોડ બે યાદી અને હવે હું એક inorder traversal કરવું અધિકાર છે. તેથી, મને 1, 2, 4 મળ્યું છે. તેથી, તે 5 ની ડાબા ઉપ-સંખ્યાની અંદરની દિશામાં પરિવર્તિત કરે છે. હવે, હું 5 ની બહાર યાદી કરું છું અને પછી હું 8 અને 9 ના આક્રમણમાં સંકળાયેલો છું કારણ કે 8 નું કોઈ બાળક બાકી નથી. મૂલ્યો 8 અને પછી 9 ની બહાર આવે છે. તેથી, તમે શું જોઈ શકો છો, કારણ કે આપણે વર્તમાન મૂલ્ય પહેલાં ડાબી બાળકની કિંમતોને છાપીએ છીએ જ્યારે મૂલ્ય હાલના મૂલ્ય પછી વર્તમાન બાળકને પછી વર્તમાન બાળકને યોગ્ય છે મૂલ્ય આ બધા મૂલ્યોને અલગ પાડવામાં આવે છે કારણ કે તે જ રીતે શોધ કીને આયોજિત કરવામાં આવે છે અને ત્યારબાદ શોધ વૃક્ષની અંદરની દિશામાંના આખરી આઉટપુટની નીચે દરેક સ્તરે પુનરાવર્તિત કરવામાં આવે છે તે હંમેશાં એક સૂચિબદ્ધ સૂચિ હોય છે. આ એક સુનિશ્ચિત કરવાની એક રીત છે કે તમે જે વૃક્ષનું નિર્માણ કર્યું છે તે એક ક્રમમાં ગોઠવવામાં આવ્યું છે, જેમાં તમે બાંધેલું કી એક શોધ વૃક્ષ છે, જે તમે ઇન-ઓવર ટ્રાવર્સલલ કરો છો અને ખાતરી કરો કે આ ટ્રાવર્સલલમાંથી મળેલી આઉટપુટ એક સોર્ટ કરેલ સૂચિ છે.

(સ્લાઇડ સમયનો સંદર્ભ લો: 08:16)

આપણે ઉલ્લેખ કર્યો છે કે આ વ્યાખ્યાનની શરૂઆત, શોધ વૃક્ષ બનાવવા માટેના મુખ્ય કારણો પૈકી એક દ્વિસંગી શોધની જેમ કંઈક કરવાનો છે અને આ ગતિશીલ ડેટા સાથે છે તેથી, અમે પણ કામગીરીમાં સામેલ અને કાઢી નાખીશું, પરંતુ મુખ્ય પાયાનું ઓપરેશન મળે છે. બાઈનરી શોધની જેમ આપણે રુટથી શરૂ કરીએ છીએ. તેથી, તમે કલ્પના કરો કે આ એરેની યાદીનું મધ્યમ છે, દાખલા તરીકે, આપણે આ તત્વ જોયું છે જો આપણે તેને શોધી લીધું છે તો તે સારું છે જો આપણે તેને મળ્યું ન હોય તો પછી આપણે યોગ્ય પેટા વૃક્ષમાં જોવાની જરૂર છે. શોધ વૃક્ષને યોગ્ય મૂલ્યમાં નાના મૂલ્યો સાથે સંગઠિત કરવામાં આવે છે, જો આપણે જે શોધ કરી રહ્યાં છીએ તે વર્તમાન નોડ કરતા નાની છે અને તે વર્તમાન નોડ કરતાં મોટું છે, તો અમે યોગ્ય જઈએ છીએ. તેથી, આ ઝાડમાં દ્વિસંગી શોધનું બહુ સામાન્યકરણ છે.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 09:04)

અહીં કોડ છે, તે ખૂબ જ સીધા આગળ છે, આપણે v ની યાદ રાખવું જોઈએ, આ પાયથોન વાક્ય છે. આપણા કાર્ય માટે પ્રથમ પેરામીટ તરીકે આપણી પાસે હંમેશા સ્વયં છે તેથી, જો વર્તમાન નોડ ખાલી છે

તો અમે તેને શોધી શકતા નથી. તેથી, જો આપણે v શોધીએ તો ખોટા બદલાવ આવે છે. તો પછી આપણે સાચા ઠરી શકીએ છીએ અને જો અમને v ન મળે તો તે જો. આ તે સ્વયં હોવું જોઈએ જો તે આ વર્તમાન મૂલ્ય કરતાં નાનું હોય તો આપણે ડાબી બાજુએ જઈએ છીએ અને તેના માટે શોધ કરીએ નહીં તો અમે યોગ્ય જઈએ અને તેના માટે શોધ કરીએ. તેથી, આ બાઈનરી શોધ બરાબર છે અને તે ખૂબ જ સરળ ફરી યાદ આવતી વસ્તુ છે જે શોધ વૃક્ષની રચનાનું અનુસરણ કરે છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 09:43)

તે પછી શોધ વૃક્ષના સૌથી નાનું અને સૌથી મોટું મૂલ્યો શોધવા માટે સક્ષમ બનશે. તેથી, નોંધ લો કે આપણે જે ચાલુ રાખીએ છીએ તે આપણે નાના અને નાનું જઈએ છીએ. તેથી, જ્યાં એક વૃક્ષની લઘુત્તમ કિંમત છે તે નાના ડાબા સૌથી વધુ પાથ સાથે છે. તેથી, જો મને ડાબેરી માર્ગમાંથી જવું પડે અને જો હું આગળ ન જઈ શકું તો હું તેને શોધી શકું છું. તેથી, અમે હંમેશા આ ફંક્શનને ત્યારે જ લાગુ કરીશું જ્યારે વૃક્ષ બિન ખાલી હોય. ચાલો ધારો કે અમે બિન ખાલી વૃક્ષમાં ન્યૂનતમ મૂલ્ય શોધી રહ્યા છીએ. ઠીક છે, આપણે ડાબા સૌથી વધુ પાથ શોધીએ છીએ. જો હું વધુ આગળ નહીં જઈ શકું તો મને તે મળ્યું. આ કિસ્સામાં, જો હું એક સુધી પહોંચું છું કારણ કે હું આગળ જઈ શકતો નથી, તો ઓછામાં ઓછો કિંમત છે, જો હું ડાબી બાજુ જઈશ તો હું એક વધુ પગલા લઈશ. તેથી, જો આપણે આને શરૂ કરીએ, દાખલા તરીકે, 5 કહેશે કે 5 એ ઉપવૃક્ષ છોડી દીધું છે તેથી, 5 ની નીચે ન્યૂનતમ મૂલ્ય એ લઘુત્તમ મૂલ્ય છે જે બાળકને છોડી દે છે. તેથી, તમે ત્રણ પર જાઓ છો, હવે અમે કહીએ છીએ કે લઘુત્તમ મૂલ્ય ત્રણ ની નીચે છે તે નીચેનું લઘુત્તમ મૂલ્ય તે બાળકને છોડી દે છે. તેથી, તમે એક પર જાઓ ત્યારે અમે કહીએ છીએ કે ન્યૂનતમ મૂલ્ય એ લઘુત્તમ મૂલ્ય 1 છે કારણ કે ત્યાં કોઈ ડાબે બાળક નથી અને તેથી, આપણને જવાબ 1 મળે છે અને તે ખરેખર 2 છે, જે નીચેથી જે કંઈ છે તે જમણી બાજુ 2 છે.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 10:50)

ડ્યૂઅર, જમણા સૌથી વધુ પાથને અનુસરીને મહત્તમ મૂલ્ય શોધી શકાય છે, જો કોઈ નહીં હોય તો પછી આપણે વર્તમાન મૂલ્ય પાછીશું નહીં તો અન્યથા આપણે વારંવાર અમારા જમણા માટે મહત્તમ મૂલ્ય જોઈએ છીએ. આ કિસ્સામાં અમે 5 થી શરૂ કરીએ છીએ, આપણે 7 થી નીચે જઈએ છીએ, પછી આપણે 9 નીચે જઈએ છીએ અને 9, 9 થી આગળ કોઈ સાચો માર્ગ આ વૃક્ષમાં મહત્તમ મૂલ્ય હોવો જોઈએ નહીં. અમે પાછળથી પાછળથી આવીશું અને શા માટે અમને આ મિનિમની જરૂર છે તે જુઓઅમ અને મહત્તમ મૂલ્ય, પરંતુ કોઈપણ રીતે તે શોધવા માટે સક્ષમ થવા માટે ઉપયોગી વસ્તુ છે અને તે કરવું ખૂબ જ સરળ છે, ફરીથી દ્વિસંગી શોધનું માળખું વાપરી રહ્યું છે.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 11:25)

તેથી, આપણે જે વસ્તુઓ કહીએ છીએ તે એક છે કે આપણે ઝાડમાંથી તત્વોને ગતિશીલ રીતે ઉમેરવા અને દૂર કરવા માટે સક્ષમ થવું જોઈએ. પ્રથમ કાર્ય કે જે આપણે શોધીએ છીએ તે શામેલ છે, આપણે વૃક્ષમાં એક ઘટક કેવી રીતે શામેલ કરીએ છીએ, તે ખૂબ સરળ છે જો આપણે તેને શોધી ન શકીએ તો તે સ્થળ જ્યાં શોધ નિષ્ફળ છે તે બરાબર છે કે જ્યાં તે હોવું જોઈએ. તેથી, આપણે તેને તે જ બિંદુ પર શામેલ કરીએ છીએ દાખલા તરીકે, આપણે આ વૃક્ષમાં 21 ને દાખલ કરવાનો પ્રયાસ કરીએ છીએ, જ્યારે આપણે 52 ને જોતા હોઈએ છીએ અને જ્યારે આપણે ત્યાં જઈએ છીએ ત્યારે આપણે પાછા જઈએ છીએ અને અમે જમણી તરફ જઈએ છીએ, પછી આપણે 21, 28 આવે છે અને આપણે શોધી કાઢીએ છીએ કે અમે ખાલી થઈ ગયા છીએ. આ માર્ગ અને આ વૃક્ષમાં 21 શક્ય નથી, પરંતુ અમને 21 મળ્યા તે 28 ની ડાબી બાજુએ હોવી જોઈએ. તેથી, અમે તેને ત્યાં દાખલ કરીએ. તેથી, અમે તેને શોધવા જોઈએ જ્યાં શોધીએ છીએ અને જો અમને તે મળતી નથી તો અમે તેને યોગ્ય સ્થાન સાથે શામેલ કરીએ છીએ. એ જ રીતે, તમે 65 ની શરૂઆત કરી શકો છો. તેથી, 65 એ 52 કરતા મોટો છે. તેથી, આપણે ઉપર જઈએ

છીએ, તો તે 74 કરતાં નાનું છે. તેથી, આપણે બાકી છીએ, પણ બાકી 74 માટે કંઈ નથી. તેથી, આપણે બહાર તે 74 ની ડાબી બાજુએ. હવે, શામેલ છે તે મૂલ્ય ન મૂકાશે જે પહેલાથી જ ત્યાં છે કારણ કે અમારી પાસે કોઈ ડુપ્લિકેટ્સ નથી. તેથી, જો આપણે દાખલા તરીકે હવે નેવું ને દાખલ કરીએ તો આપણે 52 થી જ જઈએ છીએ, આપણે 74 થી જ જઈએ છીએ અને હવે અમે શોધીએ છીએ કે 91 વૃક્ષની અંદર પહેલેથી હાજર છે. તેથી, નેવું દાખલ કરો આ વૃક્ષ પર કોઈ અસર થતી નથી.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 12:50)

આ શોધ એલ્ગોરિધમનું ખૂબ સરળ ફેરફાર છે. તેથી, અમે શોધ રાખીએ છીએ અને જો આપણે પર્ણ નોડ પર પહોંચીએ છીએ તો આપણે ખાલી નોડના અધિકારમાં આવીએ છીએ. જો તમને લાગે કે આપણે ખાલી નોડ સુધી પહોંચી ગયા છીએ તો આપણે નવો નોડ બનાવવાની સમકક્ષ છીએ, અહીં આપણે આ મુલ્ય ને વ્યુ સેટ કરીએ છીએ અને ડાબી અને જમણી બાજુએ બે ખાલી ગાંઠો ઉમેરીને આપણે નવા સરહદ બનાવીશું. . પ્રશ્નની બીજી બાજુએ, જો આપણે વૃક્ષની કિંમતને શોધીએ તો આપણે કંઈ જ નથી અને જો આપણે મૂલ્ય ન શોધીએ તો આપણે ફક્ત શોધ વૃક્ષની મિલકતનો ઉપયોગ કરીએ છીએ અને ક્યાં તો યોગ્ય પર ડાબે અથવા જમણે દાખલ કરવાનો પ્રયાસ કરો.

(સ્લાઈડ સમયનો સંદર્ભ લો: 13:32)

કેવી રીતે કાઢી નાંખો તેથી, કાઢી નાખવું શામેલ કરતાં થોડું વધારે જટિલ છે. તેથી, મૂળભૂત રીતે જ્યારે આપણે વૃક્ષમાં v શોધીએ છીએ અને તે ફક્ત એક જ હોઈ શકે છે યાદ રાખો કે આ સૂચિમાંથી કાઢી નાંખવાની જેમ નથી કે જ્યાં પહેલાં અમે શોધ વૃક્ષમાં વીની પ્રથમ નકલને દૂર કરી રહ્યા હતા ત્યારે અમારી પાસે ફક્ત એક જ નકલ છે બધા જો કિંમત. જો આપણે v શોધીએ તો આપણે તેને રદ કરવો જોઈએ. તેથી, હવે અમે વી તરીકે હંમેશાં શોધીએ છીએ જો નોડ કે જે અમે શોધી રહ્યા છીએ તે એક પાંદડી છે, તો અમે તેને પૂર્ણ કરીએ છીએ અને કંઈ જ નહીં થાય, જો તેના એક માત્ર બાળક હોય તો આપણે બાળકને પ્રોત્સાહન આપી શકીએ છીએ. જો તેની પાસે બે બાળકો હોય તો આપણી પાસે એક છિદ્ર હોય છે કે આપણી પાસે એક નોડ છે, જે આપણને મુલ્ય દૂર કરવાની હોય છે, પરંતુ તેની પાસે નીચે બંને બાજુ મૂલ્યો છે અને હવે આ કામ કરવા માટે આપણે આ મહત્તમ ફલન મહત્તમ અવયવ અથવા મિનિમલનો ઉપયોગ કરીશું. . ચાલો આપણે જોઈએ કે આ અમુક ઉદાહરણો કેવી રીતે કામ કરે છે. તેથી, આપણે સૌ પ્રથમ 65 ને કાઢી નાખીએ તો આપણે સૌ પ્રથમ 65 ની શોધ કરીશું, કારણ કે તે એક પાંદડું છે તેથી આ કિસ્સામાં આપણે આ કેસમાં પ્રથમ કેસ મેળવીએ છીએ અને આ પર્ણ કાઢી નાખીએ છીએ અને અમારું કામ થઈ ગયું છે.

(સ્લાઈડ સમયનો સંદર્ભ લો: 14:40)

હવે, અમે 74 ને કાઢી નાખવાનો પ્રયાસ કરીએ છીએ. તેથી, અમે 74 શોધીએ છીએ અને અમને લાગે છે કે તેની પાસે માત્ર એક જ બાળક છે. તેથી, જો તે માત્ર એક જ બાળક હોય તો આપણે તેને ખસેડીએ પછી આપણે અસરકારક રીતે આ લિંકને ટૂંકું સર્કિટ કરી શકીએ છીએ અને આ સમગ્ર બાબતને ખસેડી શકીએ છીએ અને સીધા 52 પોઇન્ટ 91 કરી શકીએ છીએ. તેથી, અમે આ બીજો કેસમાં છીએ, બાળકનો પ્રમોટ કરવા માટે આનો અર્થ એ થાય છે. તેથી, કાઢી નોડના એક માત્ર બાળકને આપણે બાળકને બાયપાસ કરી શકીએ છીએ અને કાઢી નાંખી નોડના માતાપિતાને રદ થયેલ નોડના એક બાળક સાથે સીધા કનેક્ટ કરી શકીએ છીએ. તેથી, આનું પરિણામ 91 ત્યાં આગળ વધશે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 15:14)

હવે, છેવટે, અમારી પાસે મુશ્કેલ કિસ્સો છે કે જે તમે નોડને કાઢી નાખવા માંગો છો જે વૃક્ષની મધ્યમાં છે, જો આ કેસ 37. તો, આપણે 37 પર આવીએ છીએ અને આપણે આને દૂર કરવા માટે હવે, જો આપણે તેને દૂર કરીએ તો ખાલી જગ્યા ખાલી હશે, અમે ખાલી જગ્યા ખાલી કેવી રીતે ભરીશું અને વૃક્ષની આકાર કેવી રીતે ગોઠવીશું? તેથી, આપણે ડાબી બાજુએ છીએ અને મહત્તમ મૂલ્યને યાદ રાખીએ છીએ કે ડાબી બાજુની બધી વસ્તુઓ 37 કરતા ઓછી છે અને બધી જ વસ્તુઓ જમણી બાજુ 37 કરતાં મોટી છે. તેથી, ડાબી ગાંઠો વચ્ચે આપણે સૌથી મોટું લે છે અને અમે તેને ત્યાં જ ખસેડીએ છીએ. ડાબી બાજુ તે નોડ કરતા નાનું રહેશે તો તમે બીજી રીત પણ કરી શકો છો અને જમણી બાજુથી નાના મૂલ્યો લઈ શકો છો, પરંતુ તમે તે નહીં કરો કે તમે ડાબી બાજુથી મહત્તમ મૂલ્ય લેવા વળગી રહેશો. આપણે ડાબી તરફ જઈએ છીએ અને મહત્તમ મૂલ્ય 28 મળે છે. તેથી, મૂળભૂત રીતે આપણે આ 28 લીધું છે અને તેને અહીં ખસેડી રહ્યા છીએ, હવે આપણે 28 ની બે નકલો ન હોવી જોઈએ. તેથી, આપણે આ 28 દૂર કરવાની જરૂર છે. અમે આ ઉપવૃક્ષની અંદર તે સારી રીતે કરીએ છીએ, આપણે 28 કાઢી નાંખો હવે આ સમસ્યા જેવું દેખાય છે કારણ કે નોડ કાઢવા માટે આપણે ફરીથી નોડને કાઢી નાખી રહ્યા છીએ, યાદ રાખો કે જે મહત્તમ મૂલ્યને વ્યાખ્યાયિત કરવામાં આવ્યું હતું તે જ રીતે તે સૌથી વધુ રસ્તાની સાથે છે. તેથી, સૌથી વધુ રસ્તાનો અધિકાર પૂર્ણમાં સમાપ્ત થશે અથવા તે નોડમાં સમાપ્ત થશે જેનો એક માત્ર બાળક છે અને અમે જાણીએ છીએ કે જ્યારે અમારી પાસે પાંદડા હોય અથવા માત્ર એક જ બાળક હોય ત્યારે આપણે પ્રથમ બે કેસોમાં છીએ જે અમે વિના સંભાળી શકીએ છીએ. આ મહત્તમ મૂલ્ય કરવાથી તે તો, આપણે ફક્ત 28 ને દૂર કરવા અને 21 ને પ્રમોટ કરવા પ્રેરાઈએ છીએ.

(સ્લાઇડ સમયનો સંદર્ભ લો: 16:37)

હવે આપણે આ ફંક્શનને જોઈ શકીએ છીએ. જો ત્યાં કોઈ મૂલ્ય ન હોય તો અમે ફક્ત સરળ કેસો પાછી મેળવીએ છીએ જ્યારે અમને નથી લાગતું કે આપણે વર્તમાન વસ્તુ છે. તેથી, જો તે વર્તમાન મૂલ્ય કરતા ઓછો હોય તો આપણે ડાબી બાજુએ જઈએ છીએ અને કાઢી નાંખો જો તે વર્તમાન મૂલ્ય કરતા વધુ હોય તો આપણે જમણી તરફ જઈએ અને કાઢી નાંખો. તેથી, કઠણ કામ આવે છે ત્યારે આપણે વાસ્તવમાં વર્તમાન મૂલ્ય પર v શોધીએ છીએ. જો આ એક પાંદડું છે તો આપણે આ બતાવ્યું નથી, આ પૂર્ણ કેવી રીતે લખવું એનો અર્થ એ થયો કે તે ડાબી અને જમણી બાળક બન્ને ખાલી ગાંઠો છે, જો તે પાંદડા છે તો આપણે તેને ખાલી બનાવીશું આપણે જોશું કે કેવી રીતે આપણે કરીશું આ એક મિનિટમાં હું તમને આ માટે કોડ બતાવીશ. તેથી, જો આ એક પાંદડું છે તો આપણે તેને હટાવ્યું છે આ પહેલું કેસ સરળ કેસ છે, આ પાંદડાનું આપણે તેને કાઢી નાખીએ છીએ અને આપણે આ નોડ ખાલી કરીએ છીએ જો બીજી તરફ જો તે માત્ર એક જ બાળક હોય, તો તેથી, વાસ્તવમાં આ કિસ્સામાં જો ડાબી બાજુ ખાલી હોય તો આપણે ફક્ત જમણી તરફ પ્રમોટ કરીએ છીએ અને જો તે બાકી છે તો ખાલી નથી, તો આપણે ડાબી બાજુથી મહત્તમ કિંમતની નકલ કરીશું અને ડાબી બાજુએ મહત્તમ મૂલ્ય કાઢી નાખીશું. આપણે આ બે ફંક્શનો ખાલી બનાવવાની જરૂર છે ખાલી અને નકલ કરો. તેથી, ખાલી ખાલી કરો ખાલી ખાલી નોડમાં કન્વર્ટ કરો, ખાલી નોડ એ એક છે જે ત્રણેય ક્ષેત્રો કંઈ નથી. તેથી, આપણે કહીશું કે સેલ્ફ ડોટ મૂલ્ય કંઈ નથી અને કોઈ ડોટ એવું કંઈ નથી પણ કોઈ ડોટ નથી પણ છે જો તેમાં કોઈ ખાલી નોડને અટકી હોય તો તે ખાલી ગાંઠો હવે વળતરથી ડિસ્કનેક્ટ થઈ ગયા છે. આ તે ખાલી કાર્ય કરે છે અને હવે કોંપિ જમણા કાર્ય ફક્ત જમણેથી બધું જ લે છે અને તેને ખસેડે છે તે યોગ્ય મૂલ્ય લે છે અને તેને વર્તમાન મૂલ્યને ડાબા મૂલ્યની બરાબર ડાટ અને ડાબી બાજુથી બનાવે છે. તેથી, આપણે ફક્ત આ નોડને મૂળભૂત રીતે લઈએ છીએ અને આ મૂલ્યોને એક પછી એકની નકલ કરીએ છીએ. તેથી, આપણે વર્તમાન ડોટને જમણી ડોટ મૂલ્ય ને વર્તમાન જમણા ડાબા જમણી ડોટ પર છોડીએ છીએ.

(સ્લાઇડ ટાઇમ નો ઉલ્લેખ કરો: 18:36)

તો, જો આ વસ્તુની કાળજીપૂર્વક તપાસ કરવામાં આવે, તો તમે એ સમજો છો કે દરેક કિસ્સામાં આપણે કિંમત શોધવા માટે એક પાથ નીચે જઈ રહ્યા છીએ અને તે પાથ સાથે આપણે શોધી શકીએ છીએ. તે

અથવા અમે અંત સુધી નીચે જાઓ અને પછી અમે બંધ. તેથી, દરેક ઓપરેશનની જટિલતા વાસ્તવમાં વૃક્ષની ઊંચાઈએ લખવામાં આવે છે જો અમારી પાસે એક સંતુલિત વૃક્ષ હોય તો સંતુલિત વૃક્ષ એક છે જ્યાં તમે વ્યાખ્યા આપી શકો છો કે દરેક વખતે જ્યારે આપણે યોગ્ય ખુરશીમાં ડાબી બાજુએ નોડ પર જઈએ તો લગભગ સમાન માપ . જો આપણી પાસે સંતુલિત વૃક્ષ હોય તો તે જોવાનું મુશ્કેલ નથી, પછી આપણે ઊંચાઈના લઘુગણક ધરાવે છે, આ ઢગલોની જેમ ઢગલો એક સંતુલિત વૃક્ષનું ઉદાહરણ છે હવે શોધ વૃક્ષ સરસ રીતે ઢગલોના સબસેટ નહીં હોય કારણ કે અમારી પાસે કેટલાક હશે છિદ્રો, પરંતુ અમારી પાસે સામાન્ય રીતે લોગરીડમીક ઉંચાઈ હશે, અમે આ ચોક્કસ કોર્સમાં વૃક્ષને કેવી રીતે સંતુલિત કરવું તે સમજાવતા નથી, આપણે તે જોઈ શકીએ છીએ કે તમે AVL વૃક્ષો તરીકે ઓળખાતા વિષયને શોધી શકો છો, જે એક સંતુલિત વૃક્ષો છે જે પેટાસરવાળા દ્વારા સંતુલિત છે. વૃક્ષો તેથી, દરેક નોડ પર સંતુલન જાળવવા માટે શામેલ કરવું અને કાઢી નાખવું શક્ય છે અને ખાતરી કરો કે આ બધી કામગીરીઓ લઘુગણક છે. ચાલો આપણે કોડને સીધી રીતે જોઈ અને તેને અમલ કરીએ અને પોતાને સગવડ કરીએ કે જે બધી વસ્તુઓ અમે અહીં લખી છે તે વાસ્તવમાં હેતુપૂર્વક કામ કરે છે.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 19:55)

અહીં આપણી પાસે ક્લાસ ટ્રી માટે પાયથોન કોડ છે જે આપણે વ્યાખ્યાનમાં દર્શાવ્યું છે. તેથી, અમે હમણાં જ કેવી રીતે ખાલી નોડનું આયોજન કરવામાં આવે છે અને પાંદડાઓ કેવી રીતે ગોઠવવામાં આવે છે તે વિશે ટિપ્પણી ઉમેરવામાં આવે છે. તેથી, ત્યાં કન્સ્ટ્રક્ટર છે જે ખાલી નોડ અથવા બે નક્ષત્ર સાથે બે ખાલી બાળકો સાથે સુયોજિત કરે છે, પછી અમે નિષ્કપટ અને ઇડીએલિક્ઝ છે કે આપણે તપાસ કરીએ છીએ કે વર્તમાન મૂલ્ય કંઈ નથી અથવા બન્ને ડાબે અને જમણા બાળકો ખાલી છે, અનુક્રમે.

(સ્લાઈડ સમયનો ઉલ્લેખ કરો: 20:26)

ત્યારબાદ આપણી પાસે આ ઇન્કશન મેમેક્રીરી છે જે પર્ણને ખાલી નોડ કોપિરાઇટમાં રૂપાંતરિત કરે છે, વર્તમાન નોડમાં યોગ્ય બાળક મૂલ્યોની નકલ કરે છે અને પછી અમારી પાસે મૂળભૂત પુનરાવર્તન કાર્યો છે.

(સ્લાઈડ સમય જણાવો: 20:39)

તેથી, અમે શોધવા સાથે શરૂ કરો. તો, તે શોધો જે દ્વિસંગી શોધની સમકક્ષ હોય તે શોધવું છે અને શોધે છે ત્યાં તેને શામેલ કરવાનો પ્રયાસ કરે છે.

(સ્લાઈડ ટાઇમ નો ઉલ્લેખ કરો: 20:48)

અને છેવટે, અમારી પાસે મોટાભાગનું છે જે કાઢી નાખવા અને કાઢી નાખવા માટે બનાવવામાં આવ્યું છે જ્યારે આપણે પરિસ્થિતિ પર પહોંચીએ છીએ જ્યાં તે એક પર્ણ હોય તો કાઢી નાખવાની જરૂર છે તે મૂલ્ય મળે છે પછી આપણે પર્ણને દૂર કરીએ છીએ. અને તે ખાલી કરો જો તે ડાબે બાળક ખાલી હોય તો અમે યોગ્ય બાળકની નકલ કરીએ છીએ નહીં તો આપણે ડાબી બાજુથી મહત્તમ કાઢી નાખીશું અને મહત્તમ મૂલ્ય વર્તમાનમાં પ્રમોટ કરીશું.

(સ્લાઈડ ટાઇમ નો સંદર્ભ લો: 21:12)

છેલ્લે, આપણી પાસે આ ઇનેરલ ટ્રાવર્સલ છે જે વૃક્ષ મૂલ્યોમાંથી સોર્ટ કરેલી સૂચિ બનાવે છે અને str ઇન્કશન માત્ર ઇનરવર્ડ ટ્રાવર્સલ દર્શાવે છે.

(સ્લાઇડ સમયનો ઉલ્લેખ કરો: 21:22)

હવે જો આપણે આ પર જઈએ તો આપણે આ પેકેજને એક ખાલી વૃક્ષની સ્થાપના કરી શકીએ છીએ અને પછી કેટલાક રેન્ડમ મૂલ્યોમાં મૂકીએ છીએ તે મહત્વનું છે ક્રમમાં ગોઠવતું નથી, અન્યથા એક સોર્ટ ટ્રી જો તમે ફક્ત એક સમયે એક દાખલ કરો તે માત્ર એક લાંબા પાથ પેદા કરશે. તેથી, હું ફક્ત કેટલાક રેન્ડમ ક્રમમાં તેને મૂકવાનો પ્રયાસ કરી રહ્યો છું. અમે આ તમામ મૂલ્યોને વૃક્ષમાં શામેલ કરીએ છીએ અને હવે આપણે છાપીએ છીએ કે મને આનું સોર્ટ કરેલ સંસ્કરણ નથી. તેથી, 1, 2, 3, 4, હું હવે વધુ મૂલ્યો શામેલ કરી શકું છું. તેથી, હું રેન્ડમ શામેલ 17 માટે કરી શકું છું અને ખાતરી કરી શકું છું કે હવે 17 અને તે પહેલાં 14 અને 18 છે અને હું આ કરી શકું છું, હું દાખલ કરી શકું છું, હું 4.5 ની જેમ વચ્ચે મૂલ્યો પણ ઉમેરી શકું છું કારણ કે મેં પૂર્ણાંકોને જમણી બાજુએ નિર્દિષ્ટ કર્યો નથી. તેથી, તે 4 અને 5 ની વચ્ચે મૂકે છે અને હવે હું કાઢી શકું છું, ઉદાહરણ તરીકે જો હું 3 કાઢી નાંખું તો મને લાગે છે કે મારી પાસે 1, 2, 4. જો હું કાઢી નાંખું છું, ઉદાહરણ તરીકે, 14 તો પછી મારી પાસે 14 વચ્ચે 7 અને 17 અને તેથી પર.

(સ્લાઇડ ટાઇમ નો સંદર્ભ લો: 22:45)

આ વધારો ચોક્કસપણે કાર્ય કરે છે, જો કે અમારી પાસે સંતુલિત છે. જો આપણે તેને સંતુલિત નહીં કરીએ તો ભય એ છે કે જો આપણે સોર્ટિંગ અને સોર્ટ ક્રમાંક રાખીએ તો અમે તેને વધુ મૂલ્ય દાખલ કરીએ છીએ જે તે યોગ્ય બાળક પર ઉમેરતી રાખે છે. તેથી, વૃક્ષ વાસ્તવમાં એક લાંબો રસ્તાની જેમ દેખાય છે. પછી તે એક સોર્ટ કરેલી સૂચિની જેમ બને છે અને દરેક ઇન્સ્ટમેશન ઓર્ડર એન સમય લેશે, પરંતુ જો આપણે જેમ આપણે કરેલા પરિભ્રમણ કરીએ છીએ, તો આપણે AVL ટ્રીનો ઉપયોગ કરી શકીએ છીએ અને પછી અમે ખાતરી કરી શકીએ છીએ કે ટ્રી લોગ એન. તેથી, બધી કામગીરીઓ શામેલ છે, શોધો અને કાઢી નાંખો છે, તે હંમેશા જાળવી રાખવામાં આવતી મૂલ્યોની સંખ્યાનું લઘુગણક માન છે.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રો. માધવન મુકુન્દ
ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટીટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
વીક - 08
લેક્ચર - 01
મેમોઇઝેશન એન્ડ ડાયનેમિક પ્રોગ્રામિંગ

કાર્યોને પકડવા માટે સરસ માર્ગ પ્રદાન કરો જે અમે ગણતરી કરવા માગીએ છીએ. હવે અમે નંબરો માટે ઇન્ક્રશન સાથે પરિચિત છે. દાખલા તરીકે, આપણે બેઝ કેસ એફની દ્રષ્ટિએ ફેક્ટોરિયલ ફંક્શનને વ્યાખ્યાયિત કરી શકીએ છીએ, અને ઇન્ક્રીવ કેસની દ્રષ્ટિએ n નો એફ કહીએ તે n વખત ફેક્ટોરિયલ છે. આપણે જે જોયું છે તે આપણે પર આધારિત કાર્યને પણ વ્યાખ્યાયિત કરી શકીએ છીએ. યાદીઓ જેવા માળખાં, દાખલા તરીકે, અમે બેઝ કેસને એક ખાલી યાદી તરીકે લઈ શકીએ છીએ, અને પ્રત્યક્ષ પ્રમાણમાં, અમે પ્રારંભિક ઘટક સાથે કંઈક કરી યાદીમાં સોર્ટ કરવાનું અને બાકીના સાથે કંઈક અલગ કરી શકીએ છીએ. નિવેશ સોર્ટને નીચે પ્રમાણે શામેલ કરેલ કાર્યની દ્રષ્ટિએ વ્યાખ્યાયિત કરી શકાય છે. તેથી, ખાલી કેસ માટેના બેઝ કેસની ઇસોર્ટ ખાલી ખાલી યાદી આપે છે. અને પછી જો તમે n તત્વો સાથે સૂચિને સોર્ટ કરવા માંગતા હોવ, તો અમે પ્રથમ તત્વને યોગ્ય રીતે ખેંચીશું અને પછી તેને અનુક્રમે બાકીના સોર્ટ કરવાના પરિણામમાં શામેલ કરીશું. આ વિધેયની નિર્ભરતાને વર્ણવવાની આ એક ખૂબ જ આકર્ષક રીત છે કે જેને આપણે મૂલ્યની ગણતરી કરવા માગીએ છીએ જે અમે નાના મૂલ્યો પર ગણતરી કરવા માટે પ્રયાસ કરી રહ્યા છીએ, અને તે અમને કેવી રીતે તેને ગણતરી કરવા વિશે શીખવું તે હેન્ડલ આપે છે.

એક સવિમુખ્ય વ્યાખ્યાના મુખ્ય લાભ એ છે કે તે સીધી રીતે ફરી યાદ આવવું કાર્યક્રમ પેદા કરે છે. તેથી, અમે ફેક્ટોરિયલ માટે આ પ્રકારના પ્રોગ્રામર માટે જોયું છે જે લગભગ સીધી વ્યાખ્યા હેઠળ છે, જે કહે છે કે 0 નો એફ 1 છે અને f એ n એ એફએન -1 બાદ આવે છે. તેથી, આપણે સીધી રીતે તેને વધુ કે ઓછું બોલવું અને તેને અનુવાદિત કરી શકીએ છીએ. આપણે જે વસ્તુ કરી છે તે જ અમે કેટલાક ભૂલ કેસની કાળજી લીધી છે, જ્યાં કોઈ વ્યક્તિ નકારાત્મક સંખ્યાને ફાળવે છે, અમે હજી પણ 1 કહીશું, અને લૂપમાં જઈશું નહીં.

સામાન્ય રીતે, જ્યારે આપણી પાસે આવી પ્રત્યક્ષ વ્યાખ્યા હોય છે, ત્યારે આપણે શું કરીએ છીએ તે ઉપ સમસ્યા છે જેને આપણે પહોંચવાનો પ્રયાસ કરવા માટે ઉકેલવા માટે છે. તેથી, દાખલા તરીકે, n નો ફેક્ટોરિયલ ગણતરી કરવા માટે, જે વસ્તુઓની જરૂર છે તેમાંથી એક એ n નો બાદ 1 નું કારણદર્શી ગણાય છે. તેથી, આપણે ફેક્ટોરિયલ n ની પેટા સમસ્યા તરીકે n ઓછા ફંક્શનલ તરીકે કોલ કરીએ છીએ. હવે બદલામાં n ઓછા 1 નો પરિબળ આપણને ફકચરક n નો 2 ની ગણતરી કરવાની જરૂર છે, તેથી વાસ્તવમાં જો તમે સાંકળમાં નીચે જાઓ તો, ફેક્ટોરલ એન પેટા સમસ્યા એ n કરતા નાના મૂલ્યો માટે તમામ ફેક્ટોગ્રામ છે. તેવી જ રીતે, નિવેશ સોર્ટ માટે, સંપૂર્ણ સૂચિને સોર્ટ કરવા માટે, તમારે પ્રથમ વસ્તુને બાદ કરતાં તમામ ઘટકોને સોર્ટ કરવાની જરૂર છે કે જે યાદીની પૂંછડી કહેવાય છે, અને બદલામાં આપણે તેની પૂંછડીને સોર્ટ કરવાની જરૂર છે. સામાન્ય રીતે, જ્યારે આપણે દાખલ કરવાના સોર્ટ કરીએ છીએ ત્યારે આપણને મળશે કે આપણે સૂચિની એક સેગમેન્ટને સોર્ટ કરવાની જરૂર છે. તેથી, આપણે સામાન્ય રીતે x_i થી x_j વિશે વાત કરી શકીએ છીએ. અને આ તમામ કેસોમાં, જે સૂચક વ્યાખ્યા આપણને જણાવે છે તે આ ઉપ સમસ્યાઓના ઉકેલોને સંયોજિત કરીને અમારા આપેલ ઇનપુટ y માટે વાસ્તવિક મૂલ્યની ગણતરી કેવી રીતે કરે છે; દાખલા તરીકે, હકીકતમાં આપણે તેને વર્તમાન ઇનપુટને ગુણાકાર કરીને આગળના નાના ઇનપુટ માટે તેને ઉકેલવાના પરિણામ સાથે જોડીએ છીએ. નિવેશ સોર્ટ માટે, અમે તેને નાની કિંમતનું નિરાકરણ કરવાના પરિણામમાં પ્રથમ મૂલ્ય દાખલ કરીને ભેગા કરીએ છીએ જે સૂચિની પૂંછડી છે.

ચાલો એક ખાસ સમસ્યા જોઈએ, જે એક મુદ્દાને પ્રકાશિત કરશે જે આપણે જ્યારે અમે પ્રત્યક્ષ પ્રમાણ પર નિર્ભર છીએ અને નિખાલસ રીતે તેમને કાર્યક્રમોમાં અનુવાદ કરીએ છીએ ત્યારે તેની સાથે વ્યવહાર કરવો પડશે. ફિબોનાકી નંબરો ખૂબ પ્રસિદ્ધ ક્રમ છે, જે ફિબોનાકી દ્વારા શોધાયા હતા, અને તે પ્રકૃતિમાં જોવા મળે છે અને તે ખૂબ જ સાહજિક છે અને તમે મોટા ભાગના તેમને જોયા હશે. ફિબોનાકી નંબરો 0, 1 છે અને તે પછી તમે ઉમેરો છો. તેથી, 1 વત્તા 0 1 છે, 1 વત્તા 1 એ 2, 3, 5 અને તેથી વધુ. તેથી, તમે પહેલાનાં બે સંખ્યાઓ ઉમેરી રહ્યા છો અને તમે આગળ વધો છો. ઇન્ડેક્સિવ ડિક્રેશન કહે છે કે 0 મી ફિબોનાકી નંબર 0 છે; પ્રથમ ફિબોનાકી નંબર 1 છે; અને તે પછી બે પછીથી, નવમી ફિબોનાકી નંબર ૭૫ બેથી પાછલા બે ઉમેરીને મેળવી શકાય છે. ફિબોનાકી નંબર 2 ફિબોનાકી 1 વત્તા ફિબોનાકી 0 છે. તે પહેલાં આપણે સીધી અનુવાદને રેકર્ડિંગ પ્રોગ્રામમાં ફેરવી શકીએ છીએ. આપણે ફક્ત પિથન ફંક્શન ફાઇબ લખી શકીએ છીએ જે કહે છે કે જો એન 0 હોય અથવા n એ 1 હોય, તો તમે કિંમત n પોતે પાછો ફરો છો. તેથી, જો n એ 0 વળતર 0 છે, જો n એ 1 છે, તો આપણે 1 પરત કરીશું. નહિતર, તમે નિમ્ન-નિકરણ 1 અને n બાદ 2 પર ફરી બોલાવીને ફિબોનાકી માટે મૂલ્યની ગણતરી કરો, આ બે ઉમેરો અને આ મૂલ્ય પાછું આપો. અહીં એક સચોટ પ્રકારનું વ્યાખ્યા છે જે તેના પરથી મેળવેલ કુદરતી રિકર્સિવ પ્રોગ્રામ ધરાવે છે.

ચાલો મૂલ્યની ગણતરી કરવાનો પ્રયત્ન કરીએ અને જુઓ કે શું થાય છે. તેથી, ધારણા છે કે આપણે આ વ્યાખ્યાનો ઉપયોગ કરીને 5 ના ફિબોનાકીની ગણતરી કરવા માગીએ છીએ. તેથી, 5 ની ફિબોનાકી, અમે બીજી કલમમાં જઈશું અને કહેવું પડશે કે આપણે નોબિસ 1 ના ફિબોનાકીને 4 અને n ની 2 ઘાત ગણવાની જરૂર છે. 5 ની ફિબોનાકી તરીકે, ગણતરી કરવા માટે બે સમસ્યાઓ સાથે અમને છોડી દે છે, 4 ના ફિબોનાકી અને ફિબોનાકી 3. તેથી, અમે આને કોઈ ક્રમમાં કરીએ છીએ; ચાલો ડાબેથી જમણે જઈએ. તેથી, અમે 4 નું ફિબોનાકી પસંદ કરીએ છીએ, અને આમાં જ આ વ્યાખ્યાને આ મૂલ્યને લાગુ પાડીને, અમે 3 ની ફિબોનાકી અને 2 ની ફિબોનાકી ગણતરી કરવા માટે જરૂરી છે. તેવી જ રીતે, અમે 3 ની ફિબોનાકીની બે પેટા સમસ્યાઓની ડાબી બાજુએ જઈએ છીએ 2 અને 1; 2 ને 1 અને 0 ની જરૂર છે. હવે 1 અને 0 માટે આપણે ફરી યાદ આવવું કોલ કર્યા વગર બહાર નીકળી શકીએ છીએ; જો n બરાબર 0 અથવા n બરાબર 1 છે, તો આપણે ફક્ત મૂલ્ય પરત કરીએ છીએ. તેથી, આપણે ફિબોનાકી 1 ને 1 અને ફિબોનાકી 0 0 વડે પાછા મેળવીએ છીએ. તેથી, આ સાથે, અમે ફિબોનાકી 2 નું ગણતરી પૂર્ણ કરી શકીએ છીએ, આપણને મૂલ્ય 1 વત્તા 0 માં બીજા શબ્દોમાં મળે છે 1. તેથી, 2 નું ફિબોનાકી 1 છે. હવે, અમે 3 ની ફિબોનાકી પાછા આવ્યા છે. તેથી, અમે 3 ની ફિબોનાકી, 2 ની ડાબા કેસ ફિબોનાકી માટે ગણતરી કરી છે. હવે અમારે જમણા કેસની ગણતરી કરવી પડશે. અને ફરી એક વાર અમે 1 નું ફિબોનાકી શોધીએ છીએ તે એક આધાર કેસ છે જે અમને 1 આપે છે, અને હવે આપણે આને ભેગા કરી શકીએ છીએ અને 3 નો ફિબોનાકી મેળવી શકીએ છીએ. હવે, આપણે 4 ની ફિબોનાકી પર પાછા આવીએ છીએ. અને અમે ડાબી બાજુની ગણતરી કરેલ છે 4 નું ફિબોનાકી. તેથી, આપણે જમણી બાજુની ગણતરી કરવાની જરૂર છે. અને હવે શું થાય છે આપણે બે વખત ફિબોનાકીને ગણતરીમાં લઈએ છીએ, ભલે આપણે પહેલાથી જ કિંમત જાણીએ છીએ. અમે નિખાલસ રીતે 2 કોલ 1 અને 0 ના ફિબોનાકીને એક્ઝેક્યુટ કરવા પડે છે, ફરી મૂલ્યો 1 અને 0 બેક અપ વધારવા તેમને ઉમેરો અને 1 મેળવો. હવે, આપણે 4 નું ફિબોનાકી 2 વત્તા 1 3 ની ગણતરી કરી શકીએ છીએ. અને હવે અમે છેલ્લે, બેક મૂળ કોલમાં જ્યાં આપણે 4 ની ફિબોનાકી અને 4 ની ફિબોનાકી ગણતરી કરવાની હતી. તેથી, અમે 4 સાથે પૂર્ણ કરી લીધું છે. હવે, અમે 3 ની ફિબોનાકી કરવા માંગીએ છીએ. નોંધ લો કે આપણે 3 ની ફિબોનાકી ગણતરી કરી લીધી છે, પરંતુ આ અકારણ અમને ફરીથી આ ફંક્શનને કોલ કરવા માટે. તેથી, આપણે ફરી આ સંપૂર્ણ વૃક્ષ ચલાવવું પડશે, બધી રીતે નીચે જાઓ, બધી રીતે જાઓ અને છેવટે 3 ની ફિબોનાકી, અલબત્ત, અમને 2 એ જ જવાબ આપો, જે અમે પહેલાથી જ જાણતા હતા, પરંતુ અમે નથી લેતા. શોષણ અથવા આપણે એ હકીકતનો લાભ લઈશું નહીં કે અમે તેને જાણતા હતા. અને આ રીતે, આપણને 3 વત્તા 2 મળે છે અને તેથી 5 નો ફિબોનાકી 5 છે.

આમાં નોંધવું છે કે આપણે ઘણી વસ્તુઓ ફરીથી અને ફરીથી કરી રહ્યા છીએ આ ચોક્કસ ગણતરીમાં, સૌથી મોટી વસ્તુ જે આપણે પુનરાવર્તન કરીએ છીએ તે 3 નું ફિબોનાકી છે. તેથી, આ મૂલ્યનો ફરીથી અને ફરીથી પુનઃ ગણતરીના પરિણામ સ્વરૂપે, જો કે સિદ્ધાંતમાં ફક્ત n ઓછા 1 પેટા સમસ્યાઓની જરૂર છે. જો આપણી પાસે 5 નું નકામું હોય તો, આપણે 4, ફાઇબ, 2 ની ફિબ અને તેથી વધુના ફીબની જરૂર છે. એન સબપ્રોબ્લેમ્સ, જો આપણે 0 ની ફીબનો સમાવેશ કરતા નથી, પરંતુ આમાંની કેટલીક સબ સમસ્યાઓ જેમ કે આ કિસ્સામાં 3 ની ફિબોનાકી,

અમે વારંવાર ઉડાઉ રીતે ગણતરી કરીએ છીએ. પરિણામે, અમે ઉપ સમસ્યાઓ હલ કરવા એક ઘાતાંકીય સંખ્યાને હલ કરવાનું સમાપ્ત કરીએ છીએ, તેમ છતાં માત્ર હુકમ કરવા માટેની વાસ્તવિક સમસ્યા છે.

તો, આપણે શું કરવા માંગીએ છીએ તે પ્રક્ષેપાત્મક વ્યાખ્યાના આ નિષ્ક્રિય પુનરાવર્તિત અમલીકરણથી દૂર છે, અને પેટા સમસ્યાને જ્યારેય ફરી મૂલ્યાંકન કરવા તરફ કામ કરવાનો પ્રયાસ કરો. આ કરવું સહેલું છે, જો આપણે પહેલા ઉપરાત કરેલ સબ તકલીફોને યાદ રાખી શકીએ તો, અમારે જે કરવાનું છે તે છે તે મૂલ્ય છે જે આપણે પહેલાથી ગણતરીમાં લેવાને બદલે તેની ગણતરી કરતાં નથી. તેથી, અમારે શું કરવાની જરૂર છે તે એક ટેબલ, એક કોષ્ટક છે જ્યાં અમે ગણતરી કરેલ મૂલ્યોને સંગ્રહિત કરીએ છીએ અને કોઈ મૂલ્યની ગણતરી કરીએ છીએ તે પહેલાં, આપણે સૌ પ્રથમ ટેબલને તપાસીએ છીએ. કોષ્ટકનો જવાબ હોય તો, અમે કોષ્ટકનો જવાબ લઈએ છીએ અને આગળ વધીએ છીએ. જો કોષ્ટકનો કોઈ જવાબ નથી, તો અમે ફરી યાદ આવવું વ્યાખ્યાને તેની ગણતરી કરીએ છીએ, અને પછી અમે તે કોષ્ટકમાં ઉમેરીએ છીએ. આ કોષ્ટકને યાદ રાખવા માટે સામાન્ય રીતે મેમરી ટેબલ કહેવાય છે; અને આમાંથી, આપણે આ શબ્દનું સંસ્કરણ મેળવીએ છીએ. તે ખરેખર મેમો છે અને યાદ નથી; જાતે મેમો લખવાના અર્થમાં સંસ્મરણ, મેમો રીમાઇન્ડર જેવું છે, તમારી જાતને એક રિમાઇન્ડર લખો કે જે આ પહેલાં કરવામાં આવ્યું છે સંમિશ્રણ એ પ્રક્રિયા છે કે જેના દ્વારા આપણે પુનરાવર્તિત કાર્યની ગણતરી કરી રહ્યા છીએ, ત્યારે અમે એક સમયે એક જ મૂલ્યોનું ગણતરી કરીએ છીએ અને જ્યારે આપણે તેમને ગણતરી કરીએ છીએ ત્યારે આપણે તેમને કોષ્ટકમાં સંગ્રહિત કરીએ છીએ અને કોઈ પણ રિસમ્પ્યુટ પહેલાં તે કોષ્ટક શોધી કાઢીએ છીએ.

અહીં આપણે કેવી રીતે ટેબલ રાખીશું, 5 ની ફિબોનાકીની ગણતરી કેવી હશે. આ આપણું ટેબલ અહીં છે. તેથી, આપણી પાસે એક ટેબલ છે, જેમાં અમુક ક્રમમાં, તે ખરેખર હવે માટે વાંધો નથી; કેટલાક ક્રમમાં અમે જ્યારે કેટલાક કેય માટે કેબ ઓફ ફિબોનાકી શોધવા તરીકે, અમે ફક્ત તેને રેકોર્ડ. અને નોંધ લો કે આ કોષ્ટક ખાલી છે, ભલે આપણે 0 ની ફિબોનાચી અને 1 ની ફિબોનાચીનો આધાર કેસ અનુક્રમે 0 અને 1 છે, અમે તમને એમ ન માનતા કે તમને તે ખબર છે, કારણ કે તે પ્રથમવાર આપણે બેઝ કેસ તે ફરી યાદ આવવું વ્યાખ્યા બહાર આવશે ચાલો જોઈએ કે તે કેવી રીતે યોગ્ય છે. તેથી, અમે 5 ની ફિબોનાચીને હંમેશાની જેમ શરૂ કરીએ છીએ, તે કહે છે કે 4 અને 3, 4 કહે છે કે 3 અને 2, 3 કહે છે કે 2 અને 1, 2 કહે છે કે 1 અને 0. અને હવે અમારા મૂળભૂત કેસમાંથી, આધાર કેસમાં ફંક્શન, આપણે 1 નું તાળું 1 પાછું મેળવીશું. તેથી, આપણે તેને આ કોષ્ટકમાં સંગ્રહિત કરીએ છીએ, આ પહેલું મૂલ્ય છે જે આપણે ખરેખર ગણતરીમાં કર્યું છે. નોટિસ અમે ધારી ન હતી કે અમે તે જાણતા હતા, જ્યારે અમે તેને બેઝ કેસમાં આવ્યા; અમે તે કોષ્ટકમાં મુકીએ છીએ. એ જ રીતે, 0 ની ફિબ 0 છે જેને આપણે પહેલાં જાણ્યું ન હતું; અમે તે કોષ્ટકમાં મુકીએ છીએ હવે આપણે આગળ આવીએ છીએ અને અમને ખ્યાલ આવે છે કે 2 નું ફિબ હવે અમારા માટે ઉપલબ્ધ છે, તે 1 વત્તા 0 1 છે. તેથી, આપણે તેને કોષ્ટકમાં સંગ્રહિત કરીએ છીએ. આપણે કહીએ છીએ કે k બરાબર 2, k ની ફિબ 1 છે. હવે આપણે 3 ની ફિબ પર પાછા આવીએ છીએ, અને હવે આપણે નીચે જઈએ છીએ અને તે આપણને ફરીથી 1 નું ફિબ ગણતરી કરવા કહે છે. જોકે આ અમને કોઈ કામ ન લઈ શકે, કારણ કે તે મૂળભૂત કેસ છે, તો અમે તે હકીકતનો ખરેખર ઉપયોગ કરી શકતા નથી. આપણે સૌ પ્રથમ કોષ્ટકમાં જોયું અને કહેવું કે કિટ 1 બરાબર માટે પ્રવેશ છે, હા ત્યાં છે અને તેથી અમે તેને પસંદ કરીએ છીએ. અમે નારંગીમાં એ હકીકતને પ્રકાશિત કરીએ છીએ કે આ મૂલ્ય વાસ્તવમાં ફરીથી ગણિત નથી, પરંતુ ટેબલમાં જોવામાં આવ્યું છે, તેથી 1 વત્તા 1 થી, હવે અમે 3 નું ફિબોનાચી છે. હવે અમે 4 ના ફિબોનાકી સુધી પાછા જઈએ છીએ અને તે અમને પૂછે છે તેની પેટા સમસ્યાઓના બીજા અડધા, એટલે કે 2 ની ફિબોનાચીની ગણતરી કરવા. ફરી એક વાર અમે શોધીએ છીએ કે અમારા ટેબલમાં 2 માટે પ્રવેશ છે. તેથી, અમે તેને નારંગીમાં માર્ક કરીએ છીએ, અને આપણે વૃક્ષના વિસ્તરણ અને કમ્પ્યુટિંગ વગર કોષ્ટકમાંથી મૂલ્ય લઈએ છીએ કારણ કે જ્યારે આપણે નિષ્પક્ષ ગણતરી કરી ત્યારે. હવે, 2 વત્તા 1 બરાબર 3 છે, તેથી અમારી પાસે 4 નું ફિબોનાચી છે. તેથી, હવે આપણે પાછા જાઓ અને 3 ની બીજી શાખા ફિબોનાચીની ગણના કરો, પરંતુ ફરી એક વાર 3 એ દલીલ છે કે અમારા ટેબલમાં છે. તેથી, આપણી પાસે 3 માટે અહીં પ્રવેશ છે. તેથી, આપણે 3 નું ફિબોનાચી જોઈ શકીએ છીએ અને ઓહ કહીએ, તે બે છે. તેથી, ફરી એકવાર અમે તેને નારંગીમાં માર્ક કરીએ છીએ. અને તેથી હવે, આપણી પાસે 5 નું ફિબોનાચી 3 વત્તા 2 છે અને તે 5 છે, અને પછી હવે આ એક નવું મૂલ્ય છે, તેથી અમે તે દાખલ કરીએ છીએ. તેથી, નોંધ કરો કે, દરેક મૂલ્યની ગણતરી કરવામાં આવી છે, અમે કાર્યની વ્યાખ્યા મુજબ વૃક્ષનું વિસ્તરણ કર્યું છે અથવા તો માત્ર એક જ વખત બેઝ કેસ જોયો છે. દરેક અનુગામી સમય, અમને મૂલ્યની જરૂર હતી જે આપણે તેને ટેબલમાં જોઈ હતી, અને આપણે જોઈ શકીએ છીએ કે કોષ્ટક બરાબર

તેટલી વખત વધે છે, કારણ કે ઉપ સમસ્યાઓને હલ કરવામાં આવે છે અને અમે કોઈ ઉપ સમસ્યાને બે વખત ક્યારેય ઉકેલી નથી. તેને બે વાર ગણતરી કરતા અમે તે કોષ્ટકને જોઈને ઉકેલીએ છીએ

આ અમારી ફિબોનાચી કોષ્ટકમાં સમાવિષ્ટ કરવા માટે ખૂબ જ સરળ પગલું છે, ફિબોનાકી ફંક્શન. તેથી, આપણે ફક્ત આ લાલ કોડ ઉમેરીએ છીએ. લીલી લીટીઓ તે છે જે અમારી પહેલાથી જ પહેલાં હતા હવે ફિબોનાચી શું કહે છે, જ્યારે તમે કોઈ નંબર મેળવો છો ત્યારે પ્રથમ વસ્તુ તમે કરો અને કોષ્ટક જુઓ. જો n ની મૂલ્ય ફિબોનાકી હોય તો, તે વ્યાખ્યાયિત કરવામાં આવે છે પછી તે મૂલ્ય પાછું લાવો; અન્યથા, અમે ફરી યાદ આવવું ગણતરી મારફતે જાઓ. આ સામાન્ય ગણતરી છે જે ફરી યાદ આવવું કોલ કરશે અને છેવટે એક નવું મૂલ્ય આવે છે જે આ ચોક્કસ n માટે મૂલ્ય છે. તેથી, આ ફંક્શનના પરિણામે આ વેલ્યુ પાછી આપતા પહેલાં, આપણે તેને કોષ્ટકમાં સંગ્રહિત કરીએ છીએ. અત્યારથી, જો આ મૂલ્યને ફરી ફરી બોલાવવામાં આવે, તો આપણે તેને ક્યારેય ગણતરી ન કરવું જોઈએ; તે કોષ્ટકમાં જ હશે તે ખૂબ જ સરળ છે કારણ કે જ્યારે તમે દલીલ n મેળવ્યા હતા કે જેના માટે તમે કાર્ય ગણતરી કરવા માંગો છો, તો તમે સૌ પ્રથમ કોષ્ટક તપાસો, જો તે કોષ્ટકમાં હોય તો તમે કંઈપણ કરતા નથી, ફક્ત તમે કોષ્ટક મૂલ્ય પરત કરો છો. જો તે કોષ્ટકમાં નથી, તો તમે તેને રિકર્સિવ વ્યાખ્યાને ગણતરી માટે લાગુ કરો છો, જેમ તમે સામાન્ય રીતે જ કરો છો. તેને ગણતરી કર્યા પછી તમે તેને કોષ્ટકમાં પ્રથમ સંગ્રહિત કરો, જેથી ભવિષ્યમાં આ વિધેયને એક્સેસ કરવામાં આવ્યાં વગર આ રિકર્ઝન કરવું પડશે અને પછી તમે જે મૂલ્ય મેળવશો તે પાછું આપશે.

આ દલીલોનાં કોઈપણ સંયોજન માટે કામ કરી શકે છે. તમારે ફક્ત એક અજોડ જરૂર છે, પાયથોન રૂપે; તમારે ફક્ત દલીલોના દરેક સંયોજન માટે એક શબ્દકોશની જરૂર છે જો તે કીની ગણતરી કરવામાં આવી છે તે પહેલાં તે ટેબલમાં હશે. પાયથોન માં સામાન્ય રીતે આ કોષ્ટક એક શબ્દકોશ છે અને એક સૂચિ નહીં કારણ કે દલીલો કોઈપણ ચોક્કસ મૂલ્યો હોઈ શકે છે. તેઓ કેટલાક શબ્દમાળા હોઈ શકે છે, કેટલાક નંબરો હોઈ શકે છે, તેઓ સતત છે કે બધા છે જરૂર નથી અમે મૂળભૂત રીતે દલીલોના ચોક્કસ સંયોજનને આપેલ છે, અમે જોશું કે શું કીની સંયોજન શબ્દકોશમાં છે કે નહીં તો આપણે તેને જોવું જોઈએ અને તે પાછું મેળવવું જોઈએ. નહિંતર, અમે આ સંયોજન માટે એક નવું મૂલ્ય ગણતરી કરીએ છીએ, તેને સંગ્રહિત કરીએ છીએ અને તે પછી તે પાછો આપો. તેથી, દાખલા તરીકે આપણે થોડીક વસ્તુઓ ઉપર ચમક્યું છે, જો તમે ખરેખર પાયથોન ને યોગ્ય રીતે લખવા માંગો છો તો અમને કેટલાક અપવાદો અને તે

આ મુખ્ય મુદ્દા પર અમને લાવે છે કે આપણે આ અઠવાડિયે બે પ્રવચનોમાં કરવા માંગીએ છીએ જે ગતિશીલ પ્રોગ્રામિંગ કહેવાય છે. તેથી, ગતિશીલ પ્રોગ્રામિંગ માત્ર આ સંસ્મિત રિકર્ઝનને વધુ અનુકૂળ બનાવવા માટેની એક વ્યૂહરચના છે. આ સંસ્મિત રિકર્ઝન અમને કહે છે કે અમે મૂલ્યોને કોષ્ટકમાં અને જ્યારે ગણતરી કરવામાં આવે છે ત્યારે સંગ્રહિત કરીશું. હવે તે ખૂબ જ સ્પષ્ટ છે કે એક સચોટ વ્યાખ્યામાં, આપણે બેઝ કેસ તરફ જઈએ છીએ અને પછી બેઝ કેસથી પાછળથી પોતાને ઉપર હાથ ધરીએ છીએ. એનો અર્થ એ થાય કે, હંમેશા કેટલાક મૂલ્યો કેટલાક આધાર કિંમતો છે કે જેના માટે કોઈ વધુ મૂલ્યોની ગણતરી કરવાની જરૂર નથી; આ કિંમતો આપમેળે ઉપલબ્ધ છે. અમારી પાસે કેટલીક સમસ્યાઓ છે જે પેટા સમસ્યાઓ ધરાવે છે, અને કેટલીક અન્ય સમસ્યાઓ કે જેમાં કોઈ ઉપ સમસ્યા નથી. જો કોઈ સમસ્યામાં ઉપ સમસ્યા હોય, તો આપણે સમસ્યા ન મેળવી શકીએ, જ્યાં સુધી આપણે તેના પેટા સમસ્યાઓ, ફિબોનાકી 4 અને 3 ને હલ નહીં કરીએ ત્યાં સુધી અમે પાંચની ફિબોનાકી મેળવી શકતા નથી. પરંતુ જો અમારી પાસે બેઝ કેસ છે તો 1 નું ફિબોનાચી અથવા 0 ની ફિબોનાચી, અમે કરીએ છીએ કોઈ ઉપ સમસ્યા નથી, તેથી અમે તેમને સીધા જ ઉકેલવા કરી શકો છો. આ એક પ્રકારનું નિર્ભરતા છે. તેથી, આપણે નિર્ભરતા હુકમમાં પેટા સમસ્યાઓને ઉકેલવી પડશે, જ્યાં સુધી કંઈક બીજું હલ કરવામાં આવ્યું ન હોય ત્યાં સુધી આપણે કંઈક પર આધાર રાખતા નથી, પરંતુ થોડો વિચાર આપણને કહે છે કે આ નિર્ભરતા હુકમ એસેકિક હોવા જ જોઈએ, આપણી પાસે નથી નિર્ભરતા ચક્ર. તેથી, જો એક મૂલ્ય તે 5 ની જેમ હોય તો 3, 3 પર આધાર રાખે છે, 1 પર આધાર રાખે છે, અને 1 ફરી 5 પર આધાર રાખે છે, કારણ કે ખરેખર આ અધિકારનો ઉકેલ લાવવાનો કોઈ રસ્તો નથી. ત્યાં હંમેશાં એક પ્રારંભિક બિંદુ હશે, અને અમે સબ સમસ્યાઓને ફરી યાદ આવતાં કોલ્સમાંથી જવાને બદલે ડિપેન્ડન્સીઝના ક્રમમાં હલ કરી શકીએ છીએ. અમે પ્રત્યક્ષ પ્રમાણનું માળખુંનું પાલન કરવાની જરૂર નથી, અમે સીધા જ કહી શકીએ છીએ, બધી સબ તકલીફો છે જે તેમને કઈ રીતે હલ કરવાની કોઈ જરૂર નથી, જે તે બધા છે જે આનો ઉકેલ ફક્ત તેના પર અને તેથી જ પર આધારિત છે. આ

એક પુનરાવર્તન માર્ગ તરીકે આપે છે. જો આપણે 5 ની ફિબોનાકી માટે પેટા સમસ્યા પર ધ્યાન આપીએ છીએ, ઉદાહરણ તરીકે, તે કહે છે કે આ તમામ પેટા સમસ્યાઓ જરૂરી છે પરંતુ ડિપેન્ડન્સી ખૂબ સરળ છે; 5 4 અને 3 પર આધાર રાખે છે; 4 3 અને 2 પર આધાર રાખે છે; 3 2 અને 1 પર આધાર રાખે છે; 2 1 અને 0 પર આધાર રાખે છે; અને 0 અને 1 પાસે કોઈ નિર્ભરતા નથી તેથી, આપણે તળિયે શરૂ કરી શકીએ, અને પછી આપણી જાતને કામ કરી શકીએ, આપણે 0 ની ફિબોનાકી કહી શકીએ છીએ, કોઈ ડિપેન્ડન્સીની જરૂર નથી, તો ચાલો હું તેના માટે મૂલ્ય લખું. 1 ની ફિબોનાકીને કોઈ નિર્ભરતાની જરૂર નથી, તો ચાલો હું તેના માટે મૂલ્ય લખું. હવે આપણે જોઈએ છીએ કે 2 ની ફિબોનાકી માટે તે જરૂરી વસ્તુઓની ગણતરી કરવામાં આવી છે. તો, હું ટેબલમાં ફાઈબ 2 ને સીધું જ 5 થી લઈ જવા વગર લખી શકું છું, હું 5 માંથી નીચે આવતા નથી, અમે ફક્ત સીધું જ કોષ્ટક ભરી રહ્યા છીએ, ફક્ત કયા મૂલ્યો પર આધાર રાખે છે તેનું મૂલ્યાંકન કરવું. તેથી, એમ ધારી રહ્યા છીએ કે આપણે કાર્ય જાણીએ છીએ, આપણે આ અવલંબનની ગણતરી કરી શકીએ છીએ અને માત્ર મૂલ્યોની ગણતરી કરી શકીએ છીએ જ્યારે નિર્ભરતા સંતોષ થાય છે. હવે, આપણી પાસે 1 અને 2 છે. તેથી, આપણે 3 ની ફિબોનાકી ગણતરી કરી શકીએ છીએ. તેથી, આપણે તેને ગણતરી કરીએ છીએ. આપણી પાસે 3 અને 4 છે, તેથી આપણે ગણતરી કરી શકીએ છીએ મારી અર્થ 2 અને ત્રણ છે. તેથી, આપણે 4 નું ફિબોનાકી ગણતરી કરી શકીએ છીએ, તેથી આપણે ગણતરી કરીએ છીએ. અને છેલ્લે, આપણી પાસે 2 અને ફિબ 3 અને ફિબ 4 છે, તેથી આપણે ફિબ 5 મેળવી શકીએ છીએ. આ મૂલ્ય તમે જોઈ શકો છો હવે એક રેખીય ગણતરી થાય છે, હું ફક્ત 0 થી 5 સુધી જઈ શકું છું અને મૂલ્યો ભરી શકું છું. હકીકતમાં, આ આપણે હાથ દ્વારા કરીએ છીએ. જ્યારે તમે મને દસમા ફિબોનાકી નંબર માટે કહી શકો છો, હું તેને લખી શકું છું. હું 0 કહી શકું છું. વત્તા 1 એ 1 છે, અને પછી 1 વત્તા 1 એ 2, 2 વત્તા 1 એ 3, 5, 8, 13, 21, 34 છે, તેથી સ્પષ્ટ રીતે તે ખૂબ જ જટિલ પ્રક્રિયા નથી કારણ કે તે ક્યારે લાગે છે આપણી પાસે આ ઘાતાંકીય રિકર્ઝન છે હું ફ્લાય પર તે વધુ કે ઓછું કરી શકું છું, કારણ કે મને જે કરવું છે તે બધા ક્રમમાં મૂલ્યો પેદા કરવાનું ચાલુ રાખે છે, અને આ ગતિશીલ પ્રોગ્રામિંગનો ગુણ છે. તે આ રિકર્ઝનને એક પ્રકારની પુનરાવર્તન પ્રક્રિયામાં રૂપાંતરિત કરે છે, જ્યાં તમે મૂલ્યો ભરી શકો છો કે જે સામાન્ય રીતે તમે એમઇએમ ટેબલમાં રિકર્ઝન દ્વારા ભરવા માટે તેને પુનરાવર્તન કરો છો, જે કોઈ નિર્ભરતા ધરાવતી નથી.

પછી ફિબોનાકીનું ગતિશીલ પ્રોગ્રામિંગ વર્ઝન ફક્ત આ પુનરાવર્તનશીલ વાદળી છે. તેથી, તે કહે છે કે તમે મૂલ્ય 0 અને 1 ની કિંમતથી પોતાને શરૂ કરી છે, જે આપણે પહેલા કહ્યું હતું કે જો n એ 0 અથવા 1 હશે તો તમે n પરત કરશો. તેથી, અહીં આપણે તેને સીધા ટેબલમાં સંગ્રહ કરીએ છીએ; આપણે કહીએ છીએ કે 0 ની ફીબ કોષ્ટક 0 છે, 1 નું ટેબલ ટેબલ 1 છે. અને પછી આપણે 2 થી n સુધી જઈએ છીએ, તેથી પાયથોન નોટેશનમાં રેન્જ એન્ડ છે n વત્તા 1; અને દરેક તબક્કે, આપણે માત્ર 1 નું મૂલ્ય i ની માપદંડ 1 ની ગણતરી કરવા માટે લઈએ છીએ, હું ઓછા 2 મૂલ્યો જે આપણે પહેલાથી ગણતરીમાં લીધેલ છે કારણ કે આપણે આ ચોક્કસ ક્રમમાં જઈ રહ્યા છીએ કારણ કે આપણે માન્યતા હુકમ 0 થી n . અને છેવટે, આપણે જે જવાબ જોઈએ તે nth એન્ડ્રી છે, તેથી આપણે તે પાછું જ કરીશું.

સારાંશમાં, નેઇવ રિકર્ઝનને વધુ કાર્યક્ષમ બનાવવાનો મૂળભૂત વિચાર બે વખત કંઈક ગણતરી કરતું નથી. અને બે વાર કોઈ પણ ગણતરી કરવા માટે, અમે જે કિંમતોને ગણતરી કરીએ છીએ તેને આપણે મેમો કોષ્ટક કહીએ છીએ જેને આ સંસ્કરણ તરીકે ઓળખવામાં આવે છે. અને ફરી યાદ આવવું તે પહેલાં અમે હંમેશા ટેબલ જુઓ. અથવા આ વધુ ઓપ્ટિમાઇઝ થઈ શકે છે, તેથી અમે ફરી યાદ આવતી કોલ્સને એકસાથે બનાવવાનું ટાળીએ છીએ અને અમે સીધું જ ટેબલમાં ડિપેન્ડન્સી ઓર્ડર ભરીએ છીએ જે એસાયક્લિક હોવું જોઈએ અન્યથા આ પેટા સમસ્યા હળવા થતી નથી. અને આ પુનરાવર્તન મૂલ્યાંકનને પુનરાવર્તન મૂલ્યાંકનમાં રૂપાંતરિત કરે છે, જે ઘણી વખત વધુ કાર્યક્ષમ છે.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ અને એલ્ગોરિધમ્સ ઇન પાયથોન
 પ્રોફેસર. માધવન મુકુન્દ
 ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
 ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ
 વીક - 08 લેક્ચર - 02
 ગ્રીડ પાથ્સ

છેલ્લા ભાષણમાં આપણે કેવી રીતે જોયું નિરંતર પુનરાવર્તન કરતા વધુ કાર્યક્ષમ પુનરાવર્તિત અથવા ઉત્તેજક વ્યાખ્યાઓ કરો, અને અમે આ કરવા માટેના સાધન તરીકે સંસ્કરણ અને ગતિશીલ પ્રોગ્રામિંગ જોયું છે. હવે, ચાલો એક સામાન્ય સમસ્યા જુઓ અને જુઓ કે કેવી રીતે આપણે આ ટેકનિકનો ઉપયોગ કરી શકીએ. તેથી, અહીં ગ્રીડ પાથની સમસ્યા છે. તેથી, આપણી પાસે ગ્રીડ અહીં છે, તમે કલ્પના કરી શકો છો કે ત્યાં રસ્તા છે જે લંબચોરસ બંધારણમાં ગોઠવાય છે. અમે કલ્પના કરી શકીએ છીએ કે આંતરછેદોની સંખ્યા છે. તેથી, નીચે ડાબા ખૂણે અમારી પાસે (0, 0) છે અને આ કિસ્સામાં, આપણી પાસે (5, 10) ડાબેથી જમણી તરફ જઈને આપણી પાસે 1, 2, 3, 4, 5 અલગ અલગ આંતરછેદો અને 10 અપ જાય છે. તેથી, આપણી પાસે (5, 10) ટોચે જમણા ખૂણે છે જો આ રસ્તાઓ અમારી પાસે છે તો તે અવરોધ છે કે તે ફક્ત પ્રવાસ કરી શકે છે અથવા જમણી બાજુ. તેથી, તમે રસ્તા ઉપર જઈ શકો છો અથવા તમે જઈ શકો છો, પણ તમે નીચે આવવું શકતા નથી. આને મંજૂરી નથી. આ એક માર્ગ છે જે ઉપર અને જમણી તરફ જાય છે, અને આપણે શું પૂછીએ છીએ તે છે કે નીચે ડાબા ખૂણેથી ઉપર જમણે ખૂણે જવા માટે કેટલી રીતો છે. તેથી, આપણે ગ્રીડ પાથ તરીકે ઓળખાતી વખતે સંખ્યાને ગણતરી કરવા માંગીએ છીએ. તેથી, ગ્રીડ પાથ તે છે જે આ અધિકારને અનુસરે છે. તેથી, આપણે જાણીએ છીએ કે આવા વિવિધ રસ્તાઓ કેટલા છે જે આપણને (0, 0) થી (5, 10) સુધી જ ચાલે છે અથવા જમણી બાજુએ છે

તો, અહીં એક વાદળી રંગ દોરેલો છે.

અહીં લાલ દોરેલા એક અલગ પાથ છે અને નોંધ લો કે આ 2 રસ્તા વાસ્તવમાં, પ્રથમ બિંદુથી જુદી જુદી દિશામાં શરૂ થાય છે અને તેઓ લક્ષ્ય સિવાય ક્યારેય મળતા નથી. તેઓ બધા પર ઓવરલેપ કરતા નથી. બીજી તરફ આપણે રસ્તાઓ ધરાવતા હોઈ શકે છે જે ઓવરલેપ કરે છે. આ પીળા પાથ આ ભાગમાં વાદળી માર્ગ સાથે તેના માર્ગના એક ભાગને ઓવરલેપ કરે છે અને તે 2 ભાગોમાં લાલ પાથ સાથે ઓવરલેપ થાય છે. ત્યાં ઘણાં જુદા જુદા રીત છે કે જેમાં આપણે તેને ઉપર અને જમણી તરફ ખસેડવાનું પસંદ કરી શકીએ છીએ અને પ્રશ્ન એ છે કે આવા ઘણા બધા રસ્તા કેટલા છે?

એક ખૂબ જ સ્ટાન્ડર્ડ અને ભવ્ય સંયોજક ઉકેલ છે. તેથી, આ વિશે વિચારવાનો એક રસ્તો એ નક્કી કરવા માટે છે કે, કેટલી ચાલે છે. આપણે એક દિશામાં 0 થી 5 અને બીજી દિશામાં 0 થી 10 સુધી જવું પડશે. તેથી, અમારે કુલ પાંચ હરોળની ગતિ અને 10 ઊભી ચાલની સંખ્યા કરવી પડશે, બીજા શબ્દોમાં કહીએ તો દરેક દિશામાં આપણે જે દિશામાં શરૂઆત કરીશું અને કયા પગલાઓ ચાલે છે, જે ચાલની પસંદગી આપણે 15 પગલાંઓ કરવી જોઈએ અને આમાંથી 5 હોરીઝોન્ટલ હોવો જોઈએ. પગલાં અને 10 ઊભી પગલાં હોવા જોઈએ, કારણ કે તે બધા અમને (0, 0) થી (5, 10) સુધી લઈ જાય છે. તેથી, આપણે જે કરવું છે તે આપણે જાણીએ છીએ કારણ કે આપણે જાણીએ છીએ કે આ 5 પગલાઓ આડી છે અને 10 ઊભી છે, જે માત્ર આડી છે અને જે આડી છે અને જે ઊભી છે. હવે એકવાર આપણે જાણીએ છીએ કે કયા રાશિઓ આડી છે, આપણે જાણીએ છીએ કે તેઓ કયા અનુક્રમમાં આવે છે, કારણ કે પ્રથમ આડી પગથિયા આપણને કોલમ 1 થી 1 સ્તંભ સુધી લઈ જાય છે, બીજું 1 આપણને એકથી 2 સુધી લઈ જાય છે. તેથી, આપણે તે સિવાયના કોઈપણ ક્રમમાં તે કરી શકીએ નહીં. . તેથી, આપણી પાસે અન્ય સ્લોટ હોય છે, જ્યાં આપણે 15 સ્લોટ કરી શકીએ છીએ, જ્યાં આપણે ચાલ કરી શકીએ છીએ અને પછી આપણે કહીએ છીએ કે આપણે પ્રથમ ચાલીએ છીએ, પછી અમે એક યોગ્ય ચાલીએ છીએ, પછી આપણે એક અપ ખસેડીએ છીએ અને બીજી તરફ ચાલો આગળ વધીએ છીએ. તેથી, દરેક પાથને આ રીતે 10 અપ ચાલ અને 5 જમણી તરફ દોરી જાય છે અને જો આપણે 5 જમણી ચાલને ઠીક કરીએ તો આપમેળે બાકી રહેલી સ્લોટ 10 અપ ચાલ અથવા તેનાથી વિપરીત હશે. તે ક્યાંતો 15 પસંદ કરે છે 5, તે 15 માંથી જમણે ખસેડવા

માટે 5 સ્થાનો પસંદ કરવાનો રસ્તો છે, અને તે તારણ આપે છે કે 15 ની પસંદગી 5 ની વ્યાખ્યા સ્પષ્ટપણે 15 પસંદ કરેલા 10 જેટલી જ છે કારણ કે અમે 10 અપ ચાલ અને વ્યાખ્યા મૂળભૂત છે ... જો તમે વ્યાખ્યાઓ જાણો છો ... પછી n પસંદ કરો કે ફેક્ટરિયલ દ્વારા n ફેક્ટોરિયલ n બાદ k ફેક્ટોરિયલ. આ k અને n ઓછા ક્યુ મૂળભૂત રીતે કહે છે કે 15 ઓછા 5 બરાબર 10 છે. તેથી, આપણે k અને n બાદ k ની દ્રષ્ટિએ સપ્રમાણ કાર્ય કરીએ છીએ. આ કિસ્સામાં અમે આ ફોર્મુલાને લાગુ કરી શકીએ છીએ જો તમે તેને કોલ કરવા માગતા હો અને સીધું જ એ જવાબ 3003 મેળવશો. મોટા ફેક્ટોરિયલ્સને લખ્યા સિવાય અને પછી નંબર શું આવે છે તે જોવું નહીં તે કરતાં વધુ ગણતરીમાં નથી.

પરંતુ સમસ્યા વધુ રસપ્રદ બની જાય છે, જો આપણે એમ કહીને પ્રતિબંધિત કરીએ છીએ કે આમાંના કેટલાક આંતરછેદને અવરોધિત કરવામાં આવ્યા છે, તો એવું લાગે છે કે કેટલાક માર્ગો ચાલુ છે અને અમે આ આંતરછેદ (2, 4). આ આંતરછેદ 2 અલ્પવિરામ 4 બીજો કોલમ છે અને ચોથા પંક્તિ નીચેથી ગણાય છે. તે વાસ્તવમાં 2 અલ્પવિરામ 3 છે, પરંતુ 1, 2, 3, 4 હા 2 અલ્પવિરામ 4. હવે, જો આપણે આ મારફતે જઈ શકતા ન હોઈએ તો કોઈ ચોક્કસ માર્ગ કે જે આ ચોક્કસ બ્લોક આંતરછેદ દ્વારા જાય છે તે હવે ગણાશે નહીં. તે 3003 સુધી કેટલાક રસ્તા લાંબા સમય સુધી માન્ય પાથ નથી.

દાખલા તરીકે, અગાઉની વસ્તુમાં આપણે જે દોરડું દોર્યું હતું તે વાસ્તવમાં આ દ્વારા પસાર થાય છે, લાલ પાથ નથી, જ્યાં આ ખરાબ વિભાગમાં કમનસીબે પીળા પાથ કમનસીબે વાદળી પાથ સાથે ઓવરલેપ થાય છે. તે આમાંથી પસાર થાય છે. કેટલાક રસ્તાઓ છે જે 3003 થી મંજૂર છે અને કેટલાંક નથી. તો, આ પ્રકારનાં બ્લોકમાં કેટલા પાથો બચી ગયા તે અમે કેવી રીતે નક્કી કરી શકીએ છીએ.

તેથી, ફરીથી આપણે (2, 4) અને પછી (2, 4) થી (5, 5) સુધી અવરોધિત થવા માટે એક અવરોધિત દલીલનો ઉપયોગ કરી શકો છો. જો આપણે ગણતરી કરી શકીએ કે કેટલા પાથ (0, 0) થી (2, 4) અને પછી કેટલી પાથ (2, 4) થી (5, 10) આવે છે, આ બધા ખરાબ પાથો છે. તેથી, આપણે આ ખરાબ પાથોને ગણતરી કરી શકીએ છીએ અને તેમને સારા રસ્તાઓમાંથી બાદ કરી શકીએ છીએ. અમે કેવી રીતે ખરાબ પાથને ગણતરીમાં લઈએ છીએ તે જ અમે સમસ્યાના નાના સંસ્કરણને હલ કરી શકીએ છીએ. તેથી, અમારી પાસે મધ્યવર્તી લક્ષ્ય છે તેથી, અમે આ ગ્રીડને હલ કરીએ છીએ કે કેટલા અહીં પાથ અહીંથી અહીં આવે છે, કેટલા પાથો અહીંથી અહીં આવે છે તેથી, (0, 0) થી (2, 4) આપણને 4 વત્તા 2 મળે છે તે યાદ રાખો 10 વત્તા 5 તે વળાંક હતો અથવા 10, 4 વત્તા 2 પસંદ કરો 2. તેથી, આપણને 15 મળે છે અને અહીંથી અહીં તફાવત એ છે કે આપણે બંને દિશામાં 3 કરવું જોઈએ અને તેથી, માફ કરશો, આપણે 6 ઉપર જવું પડશે અને અમારે 3 બરાબર જવું પડશે, અમે (2, 4) પર છીએ તેથી, આપણે 4 થી 10 અને 2 થી 5 સુધી જવું પડશે. તેથી, આપણી પાસે 6 વત્તા 3 પસંદ કરો 3, 84 (2, 4) થી આગળ વધવા માટેની રીતો અને આ તળિયેના દરેક માર્ગો, ભેગા થઈ શકે છે. ટોચ પર એક માર્ગ સાથે. તેથી, આપણે આને વધારીએ છીએ અને અમને 1260 પાથ મળે છે જે આ ખરાબ આંતરછેદમાંથી પસાર થાય છે, આપણે તેને મૂળ નંબર 3003 થી બાદ કરીએ છીએ અને અમને 1743 પાથ મળે છે જે રહે છે. તેથી, સંયોજક વલણ હજુ પણ કામ કરે છે.

હવે, શું થાય છે જો આપણે 2 આવા આંતરછેદો મૂકીએ? તેથી, આપણે એ જ વસ્તુ કરી શકીશું જે આપણે પ્રથમ અવરોધોને કારણે અવરોધિત થતા તમામ ભાગોને ગણતરી કરી શકીએ છીએ, તો આપણે તે બધા પાથને ગણતરી કરી શકીએ છીએ જે આ કિસ્સામાં પસાર થાય છે (4, 4) એ બીજા આંતરછેદ છે જે અવરોધિત કરવામાં આવ્યો છે. તો, આપણે આ બધા ભાગોને ગણતરી કરી શકીએ છીએ જે પસાર થાય છે (4, 4). આ આપણે જાણીએ છીએ કે કેવી રીતે કરવું: અમે તેને (2, 4) માટે માત્ર ગણ્યા, પરંતુ સમસ્યા એ છે કે પીળા પાથ જેવા કેટલાક પાથ છે જે બંને (2, 4) અને (4, 4) દ્વારા પસાર થાય છે. તેથી, આપણને ત્રીજા ગણનાની જરૂર છે કે જેમાં આપણે આ બંનેમાંથી પસાર થનારા માર્ગોની ગણતરી કરવાની જરૂર છે અને ખાતરી કરો કે અમે તેમને ગણતરી કરતા નથી. તેથી, એક રીત એ છે કે આપણે આ બેક ઉમેરવું. આ એક એવી વસ્તુ છે જેને સંયોજકતા સમાવેશ અને બાકાતમાં કહેવામાં આવે છે. તેથી, જ્યારે અમે આ ઓવરલેપિંગ એક્સક્લુઝન્સ ધરાવીએ છીએ, ત્યારે અમારે ઓવરલેપ ગણવું પડશે અને તેમને પાછા શામેલ કરવું પડશે. અમે આ પગલું દ્વારા કરવાનું ચાલુ રાખવું

પડશે. જો અમારી પાસે ૩ છિદ્રો હોય તો અમે વધુ જટિલ સમાવેશ બાકાત ફોર્મુલા મેળવીએ છીએ અને તે ઝડપથી ખૂબ જ જટિલ બની જાય છે, જે અમને મળવાની જરૂર છે તે સૂત્રની ગણતરી પણ થાય છે. આ કરવા માટે એક સરળ માર્ગ છે?

ચાલો સમસ્યાનું આચ્છાદન માળખું જોઈએ, ધારો કે આપણે કહીએ છીએ કે આપણે એક તબક્કામાં આંતરછેદ (આઇ, જે) માં જવું છે. આપણે કેવી રીતે આ એક તબક્કામાં પહોંચી શકીએ છીએ કારણ કે આપણાં રસ્તાઓ માત્ર ડાબેથી જમણા અને નીચેથી ઉપર જ જઈએ છીએ, એકમાત્ર રસ્તો આપણે પહોંચી શકીએ છીએ (આઇ, જે) તેના ડાબેરી પડોશી પાસેથી યોગ્ય ધાર લઈને. તેથી, અમે $(i-1, j)$ થી (i, j) થી જઈ શકીએ અથવા આપણે નીચેથી $(i, j-1)$ થી (i, j) સુધી જઈ શકીએ છીએ. નોંધ લો કે જો પાથ ડાબી બાજુથી આવે તો તે નીચેથી આવે છે તે પાથથી અલગ હોવો જોઈએ. તેથી, ડાબે આવનાર દરેક પાથ દરેક પાથથી અલગ છે જે નીચેથી આવે છે. તો, આપણે આ ઉમેરી શકીએ છીએ

બીજા શબ્દોમાં કહીએ તો જો આપણે કહીએ કે પાથ (આઇ, જે) એ સંખ્યા છે જે આપણે ગણતરી કરવા માંગીએ છીએ, તો આપણે $(0, 0)$ થી (i, j) સુધીની પાથની સંખ્યા ગણતરી કરવા માંગીએ છીએ. આ પાથને ૨ અલગ અલગ સેટ્સમાં વિભાજિત કરવી જોઈએ, જે ડાબેથી આવે છે, જે તમને યાદ છે કે બરાબર જથ્થો પથ $(i-1, j)$ છે. કેટલા પાથો છે જે $(i-1, j)$ સુધી પહોંચે છે તે દરેક પાથને એક જમણી ધારથી વિસ્તૃત કરી શકાય છે (i, j) અને, તે બધા સમાન હશે $(i, j-1)$ તે તમામ પાથો જે નીચેથી આવે છે, કારણ કે તે બધા જ બિંદુ સુધી પહોંચે છે (i, j) ત્યાંથી તેમાંથી દરેકને એક અનન્ય રીતે વિસ્તૃત કરવામાં આવશે (i, j) આ અમને આપણી સરળ આગવી સૂત્ર, પાથ (i, j) માત્ર પાથ $(i-1, j)$ અને પાથ $(i, j-1)$ નો સરવાળો છે. પછી અમે અલબત્ત, આધાર કેસોની તપાસ કરવી જોઈએ: આ કિસ્સામાં વાસ્તવિક બેઝ કેસ માત્ર પાથ $(0, 0)$ છે: કેટલા માધ્યમો હું $(0, 0)$ થી જઈ શકું? અને માત્ર $(0, 0)$ માં રહીએ? વેલ એકમાત્ર રસ્તો છે, તે ૦ રીતે કહેવું આકર્ષાય છે, પણ તે ૦ રીતે તેની એક રીત નથી કે અન્યથા કશું થશે નહીં. તેથી, અમારી પાસે $(0, 0)$ રહેવા માટે કંઈ કરવાનું નથી અને જો આપણે હવે ડાબા સ્તંભમાં આગળ વધીએ છીએ, જો તમે ડાબા સ્તંભમાં આગળ વધી રહ્યા છો, તો તેના ડાબામાંથી આવતા કોઈ રસ્તા નથી કારણ કે અમે પહેલાથી જ છીએ ડાબી સૌથી કોલમ પર તેથી, $(0, જ)$ માટેના તમામ રસ્તાઓ નીચેથી $(0, જ-1)$ સુધીના રસ્તાના વિસ્તરણ હોવા જોઈએ. તેવી જ રીતે જો તમે નીચેની પંક્તિ પર હોવ તો નીચેથી આવવાની કોઈ રીત નથી કારણ કે અમે પહેલાથી જ સૌથી ઓછા રસ્તાઓની સેટ પર છીએ. તેથી, પાથ (આઇ, ૦) ફક્ત પાથ $(i-1, 0)$ થી ડાબેથી જ આવી શકે છે.

આ આપણને ખરેખર છિદ્રો સાથે પણ ગણતરી કરવાની એક સીધી રીત આપે છે, કારણ કે, માત્ર એટલો જ તફાવત એ છે કે, જો કોઈ છિદ્ર હોય તો અમે ફક્ત જાહેર કરીએ છીએ કે કોઈ પાથ તે સ્થળ સુધી પહોંચી શકશે નહીં. તેથી, અમે ફક્ત વધારાની કલમ ઉમેરીએ છીએ જે કહે છે કે પાથ (i, j) ૦ છે જો ત્યાં એક છિદ્ર છે (i, j) ; અન્યથા અમે બરાબર એ જ પ્રત્યક્ષ સ્વરૂપ રચનાનો ઉપયોગ કરીએ છીએ અને હવે શું થાય છે, જો મારી પાસે નીચે એક છિદ્ર હોય, જો મારી પાસે નીચે એક છિદ્ર હોય, તો તે દિશામાંથી કોઈ પાથ આવી શકે નહીં કારણ કે તે સમયે વ્યાખ્યાયિત પાથ (i, j) ૦.

તો, ફરી એક વાર જો આપણે આને લાગુ કરીએ અને રીડક્ટીવ પ્રોગ્રામ માટે પ્રત્યક્ષ પ્રમાણભૂત અનુવાદનો ઉપયોગ કરીએ, તો અમે જાણીશું કે આપણે તે જ જથ્થાને ઘણી વાર રિકન્પેટ કરીશું. ઉદાહરણ પાથ $(5, 10)$. જો અમારી પાસે પાથ $(5, 10)$ છે, તો મને આની ગણતરી કરવાની જરૂર પડશે અને આ. આ માર્ગો $(5, 10)$, એટલે કે $(4, 10)$ અને $(5, 9)$ માટે ૨ પેટા સમસ્યાઓ છે, પરંતુ, ગણતરીમાં લેવા માટે $(4, 10)$ મને તેની ડાબી બાજુએ ગમે તે ગણતરી કરવી પડશે તે નીચે અને ગણતરી કરવા માટે $(5, 9)$ મને તેની ડાબા અને નીચેની ગણતરી કરવી પડશે અને હવે આપણે શું શોધી કાઢીએ છીએ કે આ જથ્થા $(4, 9)$ ની ગણતરી બમણી થઈ ગઈ છે પડોશી $(5, 10)$ અને એકવાર નીચે પાડોશીને કારણે $(5, 10)$. તેથી, આપણે યાદ અપાવીએ તે પહેલાં આપણે

મેઘકામનો ઉપયોગ કરીને ખાતરી કરી શકીએ કે આપણે કોષ્ટક i કોમા જો સ્ટોર કરીને બે વખત (i, j) ગણતરી ક્યારેય ન કરીએ, અને દરેક સમયે અમે આઠ કોમમા જ માટે નવું મૂલ્ય ગણીએ છીએ અને આપણે તેને ટેબલમાં સંગ્રહિત કરીએ છીએ અને જ્યારે આપણે જોયું, ત્યારે આપણે ગણતરીની જરૂર છે કે આપણે સૌ પ્રથમ કોષ્ટક તપાસીએ છીએ, જો તે પહેલેથી જ ત્યાં છે, તો આપણે તેને ગણતરી કરીશું અને સંગ્રહ કરીશું, પરંતુ કારણ કે આપણે જાણીએ છીએ કે એક ટેબલ છે અને આપણે જાણીએ છીએ કે કોષ્ટક શું છે માળખું મૂળભૂત રીતે તે ફોર્મની તમામ એન્ટ્રી છે i કોમમા જ. અમે પણ જોઈ શકીએ છીએ કે જો આપણે આ ટેબલને તેમની આધારભૂતપણાઓના સંદર્ભમાં પેટા સમસ્યાઓની તપાસ દ્વારા વારંવાર ભરી શકીએ છીએ.

સામાન્ય રીતે નોડમાં મૂલ્ય તે વસ્તુઓને ડાબા અને નીચે પર આધારિત છે જો ત્યાં કોઈ નિર્ભરતા ન હોય તો, તેના ડાબા અને કંઈ નીચે કશું જ હોવું જોઈએ નહીં અને માત્ર એક જ બિંદુ એટલે કે $(0, 0)$ છે. આ એકમાત્ર બિંદુ છે જે બેઝ કેસ છે જે તેની ડાબે અને કંઈ નીચે નથી તેથી તેની મૂલ્ય સીધી રીતે વાંચવામાં આવે છે. તેથી, આપણે અહીંથી શરૂ કરીએ છીએ.

યાદ રાખો કે $(0, 0)$ પરની બેઝ મૂલ્ય એક છે, અને હવે એકવાર આપણે આ કર્યું છે તે થઈ જાય છે: તમને રસ્તા પરની નિર્ભરતા યાદ છે, તે $(આઈ, 0)$ કહે છે $(i - 1, 0)$. તેથી, આપણે તેને ભરી શકીએ છીએ, કારણ કે તેની માત્ર એક જ નિર્ભરતા છે જે હવે જાણીતી છે. આ રીતે હું આખી હરોળને ભરી આપી શકું છું અને કહું છું કે આ હરોળમાં બધા એકમાત્ર પાથ છે જે પાથ જે યોગ્ય રીતે ચાલવાનું શરૂ કરે છે અને જમણી તરફ આગળ વધે છે. હવે આપણે ઉપર જઈએ અને જોઈ શકીએ છીએ કે આ વસ્તુ પણ જાણીતી છે કારણ કે, તે ફક્ત તેની નીચેની કિંમત પર જ આધાર રાખે છે અને તે જાણીતી પછી તે 2 જાણીતા છે. તેથી હું તેમને ઉમેરી શકું; કોઈપણ પદમાં મૂલ્ય યાદ રાખવું તેની ડાબા વત્તા ની કિંમત તેના તળિયે છે અને હવે હું કેટલાક નજીવી મૂલ્યો મેળવવાનું શરૂ કરું છું, અને આ રીતે હું આ કોષ્ટક પંક્તિને હરોળથી અને દરેક તબક્કે ભરી શકું છું જ્યારે હું આવું છું કંઈક કરવા માટે હું અવલંબન અજ્ઞાત સાથે હકીકત મળશે. આગળની પંક્તિ આની જેમ દેખાય છે અને આગળની પંક્તિ હવે આપણે છિદ્રો સાથે હરોળમાં આવીએ છીએ. તેથી, છિદ્રોવાળી હરોળ માટે, જ્યાં પણ આપણે તેના ડાબા અને નીચલા પાડોશીને ઉમેરીને સામાન્ય રીતે મેળવવામાં આવે તે મૂલ્ય લખવાને બદલે અમે એક છિદ્રને હટાવ્યું છે કારણ કે અમે ઇરાદાપૂર્વક 0 મૂકીશું; તેનો અર્થ એ કે, તે પંક્તિ દ્વારા કોઈ પાથને પ્રચાર કરવાની મંજૂરી નથી. હવે, જ્યારે આપણે આગલી પંક્તિ પર આવીએ છીએ, ત્યારે છિદ્રો આપમેળે ખોટા દિશામાંથી આવતા રસ્તાને બ્લોક કરશે. તેથી, ઉદાહરણ તરીકે અહીં ડાબી બાજુથી ફક્ત 6 રસ્તા આવે છે કારણ કે અમારી પાસે નીચેથી આવતા રસ્તાઓ નથી, તેથી ડાબી બાજુથી આવતા 26 રસ્તાઓ અને નીચેથી આવતા રસ્તાઓ નથી. આ રીતે આપણી પ્રત્યક્ષ પ્રમાણિત વ્યાખ્યા સરસ રીતે આપણને છિદ્રો સાથે વ્યવહાર કરવા દે છે અને તે પ્રભાવી વ્યાખ્યાથી આપણે નિર્ભરતા માળખું ઓળખીએ છીએ અને આપણે મેમો કોષ્ટકની કલ્પના કરીએ છીએ અને હવે અમે આ મેમો કોષ્ટક પંક્તિને હરોળમાં ભરી રહ્યા છીએ જેથી દરેક તબક્કે આપણે પહોંચીએ (i, j) કિંમત તેના આશ્રિત કિંમતો પહેલેથી જ જાણીતા છે તેથી, અમે પંક્તિ દ્વારા આ પંક્તિ કરવાનું ચાલુ રાખી શકીએ છીએ અને છેવટે આપણે 1363 રસ્તાઓ શોધીએ છીએ જે આ બેમાંથી ટાળે છે.

તો, આપણે નીચેની પંક્તિને બદલે એક અલગ રીત પણ કરી શકીએ છીએ, આપણે ડાબા કોલમ કરી શકીએ છીએ અને તે જ તર્ક કહે છે કે આપણે બધી રીતે જઈ શકીએ અમે બીજા સ્તંભમાં શરૂ કરી શકીએ છીએ, બધી રીતે જાઓ અને આ સ્તંભ દ્વારા કોલમ કરો અને અણધારી રીતે નહીં, અમને એક જ જવાબ મળવો જોઈએ. આવું કરવા માટે ત્રીજી રીત છે.

તેથી, એક વખતે અમારી પાસે એક $(0, 0)$ હોય તો આપણે તે ઉપરના પ્રથમ એલિમેન્ટ અને તેના જમણા પ્રથમ તત્વને ભરી શકીએ છીએ. તેથી, આપણે આ કર્યું કરી શકીએ છીએ, હવે નોંધ લો કે આના જેવા કોઈપણ કર્યું કિંમત તેના બંને એન્ટ્રીઓ છે. આ માત્ર એક જ પ્રવેશ છે, આ પણ. તેથી હવે હું આ કર્યું ભરી શકું છું. હું એક વધુ વિકર્ણ કરી શકું છું, તો પછી હું એક વધુ વિકર્ણ કરીશ. તેથી, આપણે આ વસ્તુને ત્રાંસા દ્વારા કર્યું પણ ભરી શકીએ છીએ. નિર્ભરતા માળખું અમને તેને ભરવા માટે ખૂબ જ અલગ અલગ રીતે ભરવા માટે જરૂર ન પડે, આપણે જે

કરવા માગીએ છીએ તે વ્યવસ્થિતપણે આ કોષ્ટકને પુનરાવર્તનત રીતે ભરીને ભરેલું હોતું નથી, તે આપણે ફરીથી કોલ કરવા માંગતા નથી i, j અને પછી i ઓછા $1, j$ ની એક જુઓ. અમે સીધો જ કહીએ છીએ કે જ્યારે અમે (i, j) પહોંચીએ છીએ ત્યારે અમારી પાસે જરૂરી મૂલ્યો છે, પરંતુ જે કિંમતોની અમે જરૂર છે તે ઘણી અલગ ઓર્ડરોમાં આવી શકે છે. તેથી, અમે તે મુજબ પગલે કરી શકીએ છીએ, આપણે તેને કોલમ મુજબના કરી શકીએ છીએ અને અહીં તમે જોઈ શકો છો કે આપણે તે ત્રાંસા કરી શકીએ છીએ, પરંતુ તે ખરેખર લાંબા સમય સુધી વાંઘો નથી કારણ કે આપણે વાસ્તવમાં બધા મૂલ્યો જે આપણને જરૂર છે.

તેથી, એક નાનકડા મુદ્દો, તેથી અમે કહ્યું છે કે આપણે સંસ્કરણ વાપરી શકીએ છીએ અથવા આપણે ગતિશીલ પ્રોગ્રામિંગનો ઉપયોગ કરી શકીએ છીએ. ગતિશીલ પ્રોગ્રામિંગનો ઉપયોગ કરવાના એક ફાયદામાં તે આ ફરી યાદ આવવું કોલને દૂર કરે છે. તેથી, અગાઉ ઉલ્લેખ કર્યો છે તે રિકર્ઝન, કેટલાક પહેલાંના વ્યાખ્યાનમાં પણ ભાવ સાથે આવે છે કારણ કે જ્યારે પણ તમે ફરી યાદ આવવું છો, ત્યારે તમારે ગણતરીને સ્થગિત કરવી પડશે, કેટલાક મૂલ્યો સાચવવા પડશે, તે મૂલ્યોને પુનઃસ્થાપિત કરવો પડશે. રિકર્ઝન સાથે એક વહીવટી ખર્ચ હોય છે. તેથી, વાસ્તવમાં તે વાસ્તવમાં લાગે છે કે તે ફક્ત એક ઓપરેશન છે અને અમે n બાદ 1 અથવા n બાદ 2 ની ફિલ કહીએ છીએ. વાસ્તવમાં આ ઓપરેશનને સસ્પેન્ડ કરવાનું, ત્યાં જઈને અને પાછા આવવા સાથે સંકળાયેલી કિંમત છે. તેથી, રિકર્ઝન પર બચત એ મેમોઆઇઝેશનથી ગતિશીલ પ્રોગ્રામિંગ સુધી જવાનું એક મહત્વનું કારણ છે, પરંતુ અંતિમ મૂલ્ય માટે તેના જવાબમાં ઉપયોગી છે કે નહીં તે ધ્યાનમાં લીધા વગર દરેક મૂલ્યનું મૂલ્યાંકન કરવું તે ગતિશીલ પ્રોગ્રામિંગ છે. ગ્રીડ પાથ વસ્તુમાં એક એવી પરિસ્થિતિ છે જ્યાં તમે આને સમજાવી શકો છો. કલ્પના કરો કે અમારી પાસે આ અવરોધો સરહદની અંદર બરાબર એક પગલું છે. હવે, જો આપણે આ સુધી પહોંચવું હોય તો તે ખૂબ જ સ્પષ્ટ છે કે હું માત્ર ઉપરની પંક્તિ સાથે અથવા રસ્તાની સૌથી નીચલા સ્તરે તમામ રીતે આવી શકું છું, ત્યાં અન્ય કોઈ રીત નથી કે જેનાથી હું તેમને પહોંચી શકું. તેથી, આ પોઝિશન્સની અંદર જે કંઈ પણ છે તે અહીંથી બહાર જવાનું કોઈ રીત નથી. આ બધા મૂલ્યોની ગણતરીમાં કોઈ બિંદુ નથી. આપણી પાસે આ અવરોધ છે જે આ અવરોધોની છાયામાં છે જે ક્યારેય અંતિમ વસ્તુ સુધી પહોંચી શકે નહીં. તેથી, જ્યારે આપણે યાદ કરીએ છીએ જ્યારે આપણે પાછા આવીએ છીએ અને પુનરાવર્તિત રીતે અન્વેષણ કરીએ છીએ ત્યારે તે અમને અહીં આવવા માટે કહી નહીં કારણ કે તે આ સીમાઓ ક્યારેય પસાર કરશે નહીં. બીજી બાજુ આપણી ગતિશીલ પ્રોગ્રામિંગ આંખથી બધું જ ચાલશે. તેથી, તે પંક્તિ દ્વારા પંક્તિ, સ્તંભ દ્વારા કોલમ કરશે અને છેવટે તે 0 સે મળશે, પરંતુ તે સમગ્ર n એ n ગ્રીડને ભરી દેશે. આ કિસ્સામાં કેટલા સંસ્કરણ કાર્ય કરશે? તે મૂળભૂત રીતે માત્ર સરહદ કરશે. તે માત્ર ઓર્ડર $M + n$ કરશે.

તેથી, આપણી પાસે એક મેમો કોષ્ટક છે જે પંક્તિઓ અને કોલમ્સ અને ડાયનેમિક પ્રોગ્રામિંગ એન્ડ્રીની દ્રષ્ટિએ માત્ર એક રેખીય એન્ડ્રીઓ ધરાવે છે, જે વર્ગાત્મક છે; જો બંને ન હતા તો તે n સ્ક્વેર્ડ હશે, આમ $2n$ છે. આ સૂચવે છે કે આ કિસ્સામાં ગતિશીલ પ્રોગ્રામિંગ, મોટા પ્રમાણમાં એન્ડ્રીઓની કમ્પ્યુટિંગ છે. તેથી n સ્ક્વેર્ડ $2n$ યાદ કરતાં ઘણી મોટી છે. તે ગણતરી કરવા માટે અમને ઘણો સમય લેશે, જો આપણે ફક્ત પ્રવેશ દીઠ ખર્ચની ગણતરી કરીએ, પરંતુ ફ્લિપ બાજુ એ છે કે મેમો કોષ્ટકમાં ઉમેરવાની જરૂર છે તે દરેક નોંધને એક ફરી યાદ આવવું કોલની જરૂર છે. વાસ્તવિકતા એ છે કે આ ફરી યાદ આવવું કોલ્સનો સામાન્ય રીતે સમગ્ર કોષ્ટકની કમ્પ્યુટિંગની કચરા કરતાં, તમને વધુ ખર્ચ થશે. સામાન્ય રીતે તમે સમસ્યાનું વિશ્લેષણ કરી શકો છો અને નક્કી કરી શકો છો કે સંસ્કરણ ગતિશીલ પ્રોગ્રામિંગ કરતા ઘણાં ઓછા નવા મૂલ્યોની ગણતરી કરવામાં આવે છે. તે સામાન્ય રીતે માત્ર ગતિશીલ પ્રોગ્રામિંગનો ઉપયોગ કરવા માટે સાબિત થાય છે કારણ કે ગણતરી કરવા માટેનો મૂળભૂત માર્ગ છે.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રોફેસર. માધવન મુકુન્દ ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટીટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ

વીક - 08

લેક્ચર - 03

સૌથી લાંબો સામાન્ય અનુભાગ

અમે આગોતરી વ્યાખ્યાઓ, પુનરાવર્તિત કાર્યો અને આનો ઉપયોગ કરીને કાર્યક્ષમ મૂલ્યાંકન યાદ અને ગતિશીલ પ્રોગ્રામિંગ ક્ષેત્ર ની વાત કરીએ છીએ.

તેથી, આપણે સમસ્યાઓનું ઉદાહરણ શોધી રહ્યા છીએ જ્યાં મુખ્ય લક્ષ્ય એ પ્રત્યક્ષ અર્થતંત્રને ઓળખવા માટે છે અને એકવાર તમે પ્રત્યયાત્મક માળખું ઓળખી લો પછી પ્રોગ્રામનું ફરી યાદ આવવું માળખું સ્પષ્ટ બને છે કે જેનાથી તમે ડિપેન્ડન્સીઝ અને તમારી પાસે કયા પ્રકારનું મેમો-ટેબલ છે અને તમે કેવી રીતે ગતિશીલ પ્રોગ્રામિંગનો ઉપયોગ કરી શકશો. આ એવી પ્રથા છે જે પ્રથામાં આવે છે અને વધુ ઉદાહરણો પર આશા રાખીને પ્રક્રિયા સ્પષ્ટ થઈ જાય છે, પરંતુ ગતિશીલ પ્રોગ્રામિંગની કી વસ્તુ એ પ્રત્યક્ષ અર્થતંત્રને સમજવા માટે સક્ષમ છે. તેથી, તમારે સમસ્યાની જરૂર છે, તેની ઓળખ કેવી રીતે મુખ્ય સમસ્યા તેના પેટા ભાગો પર આધાર રાખે છે અને આનો ઉપયોગ સરસ આગવી વ્યાખ્યા સાથે થાય છે જે તમે રિકર્સિવ પ્રોગ્રામમાં અનુવાદ કરી શકો છો. એકવાર તમારી પાસે ફરી યાદ આવવું પ્રોગ્રામ છે, પછી મેમો-ટેબલ અને ગતિશીલ પ્રોગ્રામિંગ તેમાંથી આપમેળે બહાર આવે છે. આ શબ્દોમાં શામેલ કરવામાં સમસ્યા છે તેથી, તમે શું કરવા માગો છો તે શબ્દોનો એક જોડ લે છે અને સૌથી લાંબો સામાન્ય સોવર્ડ શોધો. ઉદાહરણ તરીકે, અહીં આપણી પાસે ગુપ્ત અને સચિવ છે અને રહસ્ય પહેલેથી જ સેક્ટરીમાં છે અને ગુપ્ત ગુપ્તમાં પોતે સૌથી લાંબો શબ્દ છે. જે સૌથી લાંબો સોનવુડ છે તે શબ્દ ગુપ્ત છે અને તેની પાસે લંબાઈ 6 છે. ચાલો આપણે આગળના વિચારોને દબાવીએ અને ટ્રાયેક્ટ કરીએ તો વાસ્તવમાં આ તે છે જે કહે છે કે લંબાઈ 5 છે, તેવી જ રીતે જો આપણે બાયકાક્ટર અને ગુપ્ત પછી સેકંડ છે. જ્યારે અમે હર્ષાર્થ કહીએ છીએ, અલબત્ત અમે અર્થમાં એક શબ્દ અર્થ નથી; અમે ફક્ત અક્ષરોનો ક્રમ અર્થ કરીએ છીએ. તેથી, સે લંબાઈ 3 માં સૌથી લાંબુ સામાન્ય ઉપજા છે અને જો તમારી પાસે ડિરેક્ટર અને સેક્ટરી જેવા બે અલગ અલગ શબ્દો હોય, તો ક્યારેક તમારી પાસે માત્ર નાની વસ્તુઓ હોય શકે છે, ઉદાહરણ તરીકે, અહીં ફરીથી અને ઇસી સોવર્ડના ઉદાહરણો માટે છે પરંતુ ખરેખર છે ખૂબ લાંબા શબ્દો જે સોક્સવર્ડ માટે સામાન્ય છે તે ફક્ત 2 લંબાઈ છે.

અહીં વધુ ઔપચારિક વર્ણન અધિકાર છે. તેથી, માની લેવું કે મારી પાસે બે શબ્દો છે u અને v. તેથી, u લંબાઈ મીટર છે અને v લંબાઈ n છે અને પિથન નોટેશન અને નંબર 0 થી n ઓછા 1, 0 થી n ઓછા 1 ની મદદથી સંખ્યાઓ છે, પછી હું શું કરવા માંગુ છું A_i થી શરૂ કરવા અને K પગલાંઓ પર જવા માટે સક્ષમ છે તેથી, હું i વત્તા k ઓછા 1 અને b_j to j વત્તા k ઓછા 1 જેમ કે, આ બે સેગમેન્ટ્સ એક સરખા છે, આ એક સામાન્ય ઉપજ છે અને આપણે સૌથી લાંબી આવા સામાન્ય ઉપનામ શોધવા માંગીએ છીએ, કેવ શું છે, આપણે નથી પણ સબવેન કરવા માંગો છો, તે શોધવામાં આવશે કે સબવેન આડપેદાશ હશે. તમારે પહેલા ફક્ત કેવુ શોધવાની જરૂર છે, યુએનની સૌથી લાંબી સામાન્ય ઉપજની લંબાઈ શું છે?

એક જડ ફોર્સ એલ્ગોરિધમ છે જેનો ઉપયોગ તમે જે કરી શકો છો તે ફક્ત બે શબ્દોમાં i અને j થી શરૂ કરો. દરેક શબ્દમાં તમે u_j માં v માં એક સ્થાન શરૂ કરી શકો છો અને જુઓ કે તમે કેવી રીતે ન શોધી શકો તે પહેલાં તમે કેવી રીતે જઈ શકો છો. તેથી, તમે u_i અને b_j અધિકાર સાથે મેળ ખાય છે તેથી, જો એઆઇ અને બીજે કામ પછી તેના ઇન્ડેક્સ. તેથી, તે બીજે હોવી જોઈએ અને જો એઆઇ અને બીજે કાર્ય પછી તમે એઆઇ વત્તા 1 બીજે પ્લસ 1 પર જાઓ છો અને જ્યારે પણ આપણે જુદા જુદા બે અક્ષરો શોધીએ છીએ ત્યારે એજેથી શરૂ થતા કોમન્સનો અંત આવી ગયો છે અને તેથી તમે i_j થી કંઈક સામાન્ય subword હવે, બધા જ જેએસમાં તમે સૌથી લાંબો સમય શોધી શકો છો અને તે તમારું જવાબ બની જાય છે. હમણાં, આ કમનસીબે અસરકારક રીતે હવે એક n સમઘન અલ્ગોરિધમનો છે. અમને લાગે છે કે m અને n સમાન તકનીકી રીતે સ્કેલેડ થઈ શકે છે કારણ કે મે અને જના વિવિધ વિકલ્પો M વખત છે અને સામાન્ય રીતે મેં i_j શરૂ કર્યું અને પછી મને જમણી બાજુથી અને જથી અંત

સુધી જવું પડશે. તેથી, આપણે આ સ્કેન માટે દરેક આઈજે સ્કેન કરવું પડશે, સામાન્ય રીતે ઓર્ડર એન ફેક્ટરમાં ઉમેરાય છે અને તેથી અમારી પાસે તમને ગમે તે ક્રમમાં એમએન સ્કેન્ડ અથવા ઓર્ડર એન ક્યુબ છે.

અમારો ધ્યેય કેટલાક ઇન્ડિવિડુઅલ માળખું શોધવાનું છે જે આ વસ્તુને ગણતરીત્મક રીતે વધુ કાર્યક્ષમ બનાવે છે. તો, આ પ્રત્યક્ષ અર્થતંત્ર શું છે? સારું, આપણે તે પહેલાથી જ જોઈ શકીએ છીએ કે ક્યારે આપણે કહી શકીએ કે એક લંબાઈ K ના શરૂ થતાં આઈએમ માં શરૂ થતા કોમન્સ ઉપનગરીય છે, પ્રથમ વસ્તુ એ છે કે આ એઆઈની જરૂર છે જે બી જ હશે. તેથી, મને આ શરતની જરૂર છે અને હવે જો આ ij પર લંબાઈના એક કોમન્સ સબઝર્ડ છે, તો સવિવારના અવશેષો એટલે કે આ સેગમેન્ટમાં હું 1 થી આ અને જ વત્તા 1 થી આ સાથે મેળ ખાતો હોવો જોઈએ અને તે બદલામાં એ.સી. અહીંથી અહીંથી ઓછા 1 લંબાઈના ઉપજ. તેથી, અમે એમ કહીએ છીએ કે ij પર શરૂ થતી એક લંબાઈના સબસર્ડ છે, જો એઆઈ બીજે બરાબર છે અને આઈ વત્તા 1 અને જ વત્તા 1 થી એસ્ક માઈનસ 1 લંબાઈના સબવેન છે. બીજા શબ્દોમાં કહીએ તો, હવે હું નીચેની વ્યાખ્યા લખી શકું છું, હું કહી શકું છું કે સૌથી લાંબી સામાન્ય લાંબી સામાન્ય ઉપસંહાર એલસીડબલ્યુ લંબાઈ જે આઈ થી શરૂ થાય છે. ઠીક છે, જો તેઓ બે નથી અથવા એ જ નથી જો એઈ એ જ નથી તો બધામાં કોઈ સામાન્ય ઉજાબ્દી નથી કારણ કે જો હું શરૂ કરું તો મને તરત જ બે અલગ અલગ અક્ષરો મળશે તેથી, જ્યારે લંબાઈ 0 હોય છે તો હું અનુલક્ષીને શોધી શકું છું કે સૌથી લાંબો સામાન્ય ઉજા શું છે જે મેં શરૂ કરી શકે છે અને જ 1 વત્તા 1 થી શરૂ કરો. શોધો કે હું ત્યાંથી શું કરી શકું છું અને શબ્દને હું એક અક્ષર ઉમેરી શકું છું કારણ કે આ વર્તમાન અક્ષર એઈ બરાબર બી જે. તેથી, મને એક વત્તા મળે છે અને સીમાની સ્થિતિનો આધાર જ્યારે બે શબ્દોમાંથી એક ખાલી અધિકાર છે. જો મારી પાસે કોઈ અક્ષરો બાકી નથી, જો હું આઈજે ગયો હોય તો હું અને જે. તેથી, જો હું અથવા j શબ્દના અંત સુધી પહોંચ્યો હોય તો તે સમયે એક સામાન્ય સોઝવેવર્સની કોઈ શક્યતા નથી. તેથી, જ્યારે આપણે કોઈ એક શબ્દના અંત સુધી પહોંચીએ છીએ ત્યારે જવાબ 0 હોવો જોઈએ.

આ આપણને નીચેની વ્યાખ્યાઓ આપે છે તેથી, યાદ રાખો કે તમે ખરેખર લંબાઈ મીટર છે. તેથી, તેની પાસે 0 થી ઓછા 1 છે. તેથી, આપણે શું કરીશું, આપણે એમ દર્શાવવું પડશે કે આપણે છેલ્લી અક્ષરને ઓળંગી દીધી છે. તેવી જ રીતે, v માં 0 થી n ની -1 ની માન્ય સ્થિતિ છે. તેથી, આપણે આ 0 થી n માં ઉપયોગ કરીશું. તેથી, જો હું મીટર કે જ બને તો n એનો અર્થ એ થાય છે કે અનુરૂપ ઇન્ડેક્સ શબ્દના અંતથી આગળ વધ્યો છે. તેથી, આ મીટર હોવું જોઈએ અને આ n નો હોવો જોઈએ. આપણી પાસે એ છે કે જો તમે મીટર પર પહોંચશો તો એન કોમમા જ 0 નું એલસીડબલ્યુ 0 હશે કારણ કે અમે લંબાઈના ભૂતકાળમાં ગયા છીએ. એ જ રીતે, જો તમે n સુધી પહોંચો છો, પરંતુ i અલ્પવિરામ નો એલસીડબલ્યુ 0 છે, કારણ કે તમે વીની લંબાઈની લંબાઈ ગયા છો અને જો તમે લંબાઈની બહાર નથી ગયા, જો તમે કોઈ માન્ય સ્થિતિમાં શબ્દની અંદર હોવ તો લંબાઈ થશે. 0 હોઈ શકે જો બે હોદ્દા સમાન ન હોય તો જો Ai બીજેની બરાબર નથી, અન્યથા અનુલક્ષીને હું લંબાઈની ગણતરી આઈ વત્તા 1 અને j વત્તા 1 થી અને તેને એક ઉમેરો. આ કિસ્સો છે જ્યારે એઆઈ બીજે બરાબર છે કારણ કે તે સેગમેન્ટ હું વિસ્તૃત કરી શકું છું. તેથી, આ ફક્ત એક સમીકરણમાં જ કહી રહ્યો છે જે આપણે જાણીએ છીએ તે અગાઉની વ્યાખ્યા છે.

તેથી, અહીં આપણે દ્વિભાજન અને ગુપ્ત માટેનું ઉદાહરણ જોયું છે. અમારી પાસે 0 થી 5 ની સ્થિતિ છે અને ત્યારબાદ આપણી પાસે 6 શબ્દો છે જે શબ્દનો અંત દર્શાવે છે અને હવે યાદ રાખો કે જે રીતે આપણી પ્રત્યક્ષ પ્રમાણ આધારિત વ્યાખ્યા શબ્દપ્રયોગ કરવામાં આવતો હતો ij ફક્ત 1 વત્તા 1 j વત્તા 1 પર જ આધાર રાખે છે. તેથી, વાસ્તવમાં આ રીતે ડિપેન્ડન્સી તીર તે દર્શાવે છે અને ક્રમમાં ઉકેલવા માટે હું આ પ્રથમ ઉકેલવા માટે જરૂર છે. 2 અલ્પવિરામ 3 નું મૂલ્ય મૂલ્ય 3 અલ્પવિરામ 4 પર આધારિત છે. આને ઉકેલવા માટે મને કાંઈ હલ કરવાની જરૂર નથી કારણ કે બધું એક વાર હું. તેથી, આને ઉકેલવા માટે માત્ર એટલો જ આવશ્યક છે. અમે મૂળભૂત રીતે, અમારી પાસે આ સરળ વસ્તુ છે જે કહે છે કે ખૂણા અને ખરેખર જમણા સ્તંભ અને નીચેની વસ્તુને કંઈપણ આવશ્યકતા નથી અને અમે જાણીએ છીએ કારણ કે તે તમામ 0 સે છે

આપણે વાસ્તવમાં તે મૂલ્યો 0 તરીકે ભરી શકીએ છીએ કારણ કે તે વ્યાખ્યા દ્વારા આપણને આપવામાં આવે છે અને હમણાં આપણે શરૂ કરી શકીએ છીએ, દાખલા તરીકે, આપણે આ મૂલ્યથી શરૂ કરી શકીએ છીએ કારણ કે તેનું મૂલ્ય જાણીતું છે. અમે આ ટી સાથે મેળ ખાય છે કે નહીં તે જોવા મળશે. તેથી, અમે એક વત્તા મૂલ્ય બે તળિયે લઈએ છીએ. તેથી, આપણને 1 મળે છે અને પછી આપણે ચાલતા જઈએ છીએ અને દરેક બિંદુએ તે જ કરી શકીએ છીએ જો આપણે કહીશું કે જો સી એ જ નથી. તેથી, જો તમે આ અક્ષરોને અહીં જોશો તો તેમાંના કોઈપણ અક્ષરો જમણા અક્ષરોમાં કંઈ નહીં. તેથી, આ તમામ પત્રો માટે મને સીધો 0 મળશે, કારણ કે તે કહે છે કે એઈ બી જ નથી. હું પણ વત્તા 1 જા વત્તા 1 પર નજર કરતો નથી, હું સીધી કહું છું 0 કારણ કે ત્યાં નથી. તેથી, આ રીતે હું આ સ્તંભ ભરી શકું છું. આ અમારી ગ્રીડ પેકની જેમ છે જે સ્તંભ દ્વારા સ્તંભ દ્વારા ભરી શકે છે તેમ છતાં ત્યાં નિર્ભરતા ડાબી અને નીચેની હતી અને અહીં અવલંબન ત્રાંસી નીચે જમણી બાજુ છે. હું સ્તંભ દ્વારા સ્તંભ ભરી શકું છું અને હું ચાલુ રાખી શકું છું અને જો હું ચાલુ રાખું છું તો હું એન્ટ્રી 3 શોધી શકું છું. તેથી, એન્ટ્રી 3 એ સૌથી મોટી એન્ટ્રી છે જે હું જોઈ શકું છું અને તે આ જવાબ 3 માં વાસ્તવિક જવાબ છે.

અને હવે અમે પહેલાં જણાવ્યું હતું કે અમે સૌથી લાંબો સામાન્ય subword લંબાઈ પર ધ્યાન કેન્દ્રિત કરવામાં આવે છે શબ્દ પોતે કારણ નથી. અમને જરૂર છે; આપણે તે કરવા માટે પરવડી શકીએ છીએ કારણ કે એકવાર આપણી પાસે લંબાઈ મળી જાય પછી અમે ખરેખર જવાબ વાંચી શકીએ છીએ. તેથી, આપણે પોતાને પૂછીએ છીએ કે શા માટે આપણે અહીં 3 મેળવ્યાં આપણને અહીં 3 મળ્યું છે કારણ કે આપણે 1 વત્તા 2 તરીકે આવ્યા છીએ. કારણ કે આપણે 1 વત્તા 2 તરીકે આવ્યા છીએ તેનો અર્થ એ છે કે આ બે અક્ષરો સમાન છે. તેથી, આપણને 2 અહીં મળ્યું કારણ કે તે 1 વત્તા 1 છે. તેથી, આ બે અક્ષરો પણ એક જ હોવા જોઈએ. છેલ્લે, આપણને એક મળ્યું છે કારણ કે આ 1 વત્તા 0 છે. તેથી, આ બે અક્ષરો સમાન છે. તેથી, આ ત્રણ અક્ષરો સમાન હોવો જોઈએ.

જો તમે તે જાદુઈ મૂલ્યથી નીચે જઈ રહ્યા છો, તો સૌથી મોટું મૂલ્ય છે અને અમે અનુક્રમમાં અનુસરીએ પછી આપણે વાંચી શકીએ છીએ અને અનુરૂપ પંક્તિ અથવા સ્તંભ કારણ કે તે સમાન છે, તમારી વાસ્તવિક ઉપજ જે સૌથી લાંબો છે આ બંને માટે સામાન્ય સ્યુડવર્ડ.

અહીં પાયથોન માં ખૂબ સરળ અમલીકરણ છે. તેથી, તે કહે છે કે તમે બે શબ્દો સાથે શરૂ કરો છો અને તમે, તમે આ એલસીવી વસ્તુની શરૂઆત ત્રણેય સ્તંભની એન.આર.ડી.માં n th પંક્તિ પર અને પછી, હવે તમને મહત્તમ મૂલ્ય યાદ છે. તેથી, તમે મહત્તમ મૂલ્ય 0 થી શરૂ કરીને તે રાખો અને પછી તમે આ ચોક્કસ કેસમાં કોલમ ઓર્ડર ભરો. દરેક સ્તંભ માટે, પછી દરેક પંક્તિ અને તે સ્તંભ માટે તમે સમીકરણનો ઉપયોગ કરીને આ વસ્તુને ભરો છો, જો તે 1 વત્તા બરાબર છે તો તે 0 કહે છે અને જો હું જોઉં છું અને જો નવું મૂલ્ય છે તો આ વસ્તુ છે જ્યાં હું અપડેટ કરું છું. જુઓ અને નવી એન્ટ્રી જે એન્ટ્રી કરતાં મોટી છે જે હાલમાં મહત્તમ છે, હું મહત્તમ અપડેટ કરું છું. તેથી, આ મને ઝડપથી શોધી કાઢવા માટે પરવાનગી આપે છે કે તે મહત્તમ લંબાઈ શું છે અને છેવટે, જ્યારે હું આ દેખાવમાંથી પસાર થું છું ત્યારે હું સમગ્ર કોષ્ટક ભરીશ અને હું મહત્તમ મૂલ્ય પરત કરીશ.

તેથી, જ્યારે અમે જડ બળ દ્વારા કર્યું ત્યારે અમારી પાસે ઓર્ડર MN ચોરસ એલ્ગોરિધમ હતું. અહીં આપણે કોષ્ટક ભરીએ છીએ જે કદ ઓર્ડરનો n એ છે અને દરેક પ્રવેશ ફક્ત વિશ્વની જથ્થા સ્થિતિ શબ્દમાં n th પોઝિશન ચકાસવા માટે જરૂરી છે અને તેના આધારે, જો જરૂરી હોય તો એક એન્ટ્રી 1 વત્તા 1 જી વત્તા જુઓ 1. તે સતત સમય અપડેટ છે અમે એક માપ ઓર્ડર એમએન ભરવાની જરૂર છે દરેક સુધારા સતત સમય લે છે. તેથી, આ અલ્ગોરિધમનો અમને ગતિશીલ પ્રોગ્રામિંગનો ઉપયોગ કરીને એમ.ન.

લાંબો સામાન્ય સોડવર્ડ કરતા વ્યવહારમાં વધુ ઉપયોગી સમસ્યા છે જેને સૌથી લાંબી સામાન્ય અનુગામી કહેવામાં આવે છે. તેથી, સબ-વર્ડ અને વારાફરતી વચ્ચે તફાવત એ છે કે આપણે વચ્ચેના કેટલાક પત્રો છોડવાની મંજૂરી છે. તેથી, દાખલા તરીકે, જો તમે પહેલાંના ગુપ્ત અને સચિવના ભૂતકાળના ઉદાહરણો પર પાછા જાઓ, તો તેમાં કોઈ સમસ્યા નથી કારણ કે સબ-વર્ડ વાસ્તવમાં આખી વસ્તુ છે અને ફરીથી દ્વિભાજન અને ત્રિમૂર્તિ માટે પણ તે એક જ વસ્તુ છે, પણ જો આપણે બાઈસેક્ટ અને ગુપ્ત છે પહેલાં જો આપણે અમારી અવગણવાની પરવાનગી ન આપી, તો અમે સેકન્ડ સાથે સેકન્ડ સાથે મેચ કરી શકીએ છીએ, પરંતુ હવે અમે આ

વધારાની ટી લઈ શકીએ છીએ અને અમે ત્યાં 2 અને માર્કઆ ટી તરીકે કહે છે કે સંપ્રદાય યોગ્ય શબ્દ છે, જે અનુગામી અનુગામી સાથે બંધબેસે છે જે જમણી બાજુના ડાબે શબ્દમાં એક સબઝર્વર્ડ છે તે કોઈ સબસ્વર નથી. અનુગામી માટે મને સેકંડ ટી મેળવવા માટે કેટલાક પટ્ટીઓ ટીપાં પડશે. એ જ રીતે, જો મારી પાસે સેક્ટરી અને ડિરેક્ટર હોય તો હું સમજદાર રીતે બોલવાથી બન્નેમાં ઇક્ટોસ્ટર જેવી વસ્તુઓ શોધી શકું છું. તો, શા માટે આ વધુ સારી સમસ્યા છે?

વેલ અમે જોશું, પરંતુ અવગણીને અર્થ શું છે, એટલે અવરોધો એટલે કે હું સેગમેન્ટો મેળવી શકું છું જે ગાબડાથી જોડાયેલા છે. તેથી, હું આ સેગમેન્ટ મેળવું છું પછી હું આ સેગમેન્ટો ચાલુ રાખવા ઈચ્છું છું. તેથી, હું આગામી મેચ માટે જુઓ, હું છોડી દો, પરંતુ આગામી મેચ મારા જમણા આવવા જ જોઈએ. તે જમણી તરફ અને વર્તમાન મેચની નીચે આવવું જ જોઈએ કારણ કે હું કોઈ શબ્દની પાછળ પાછળ જઈ શકતો નથી અને ફરીથી મેચને શરૂ કરી શકતો નથી. તેથી, દાખલા તરીકે, હું અહીં જઈ શકું છું અને કહી શકું છું કે આ એક એક્સ્ટેન્શન છે કારણ કે આ માટે મને પાછા જવાનું અને ફરીથી મેચ કરવા માટે સેકન્ડમાં જોવા મળતી ઇમેલનો ફરી ઉપયોગ કરવાની જરૂર છે. ગુપ્તમાં બીજો ઇ છે જે મંજૂર નથી, હું આગળ વધારી રાખી શકું છું જે ટેબલમાં જમણી તરફ અને જમણી બાજુ નીચે અને નીચે જતા હોય છે. તેથી, હું બંને શબ્દોમાં ઇન્ડેક્સના ક્રમમાં વધારો કરી રહ્યો છું અને હું આ વસ્તુઓને એકસાથે ભેગા કરી શકું છું અને આ તે છે જે સૌથી લાંબી પ્રકારની વાતો છે. તેથી, અમે સૈદ્ધાંતિક રીતે લાંબા સમય સુધીના સામાન્ય ઉપદ્રવના જવાબને જોઈ શકીએ છીએ અને આ હોશિયાર જોડાણોને શોધી શકીએ છીએ, પરંતુ તે તારણ આપે છે કે તે પ્રત્યક્ષ રીતે કેવી રીતે કરવું તે વધુ સરળ માર્ગ છે.

પ્રેરણા, ત્યારબાદ અનુગામી મેચો માટેના મોટા પ્રોત્સાહનોમાંની એક, આનુવંશિકતાની જેમ વસ્તુઓમાંથી આવે છે, દાખલા તરીકે, જ્યારે આપણે બે જીવોના જનીનો ક્રમની સરખામણી કરીએ છીએ ત્યારે તે ભાગ્યે જ સમાન હોય છે. તેથી, જે આપણે શોધી રહ્યા છીએ તે મોટા મેચો છે, જ્યાં કેટલાક જંક જનીનો હોઈ શકે છે, જેની વચ્ચે તમે કાઢી નાંખવા માંગો છો. તેથી, તમે કહી શકો છો કે બે જીન સિક્વન્સ અથવા બે સજીવો સમાન હોય છે, જો સમગ્ર જિનોમ પર મોટા ઓવરલેપ માત્ર વ્યક્તિ સેગમેન્ટ્સ સાથે જ નહીં હોય, પરંતુ બન્ને બાજુની મિનિમલ વસ્તુઓને ફેંકી દેવા દ્વારા, આપણે તે સંરેખિત કરી શકીએ છીએ આપણે કહીએ છીએ અને અન્ય મહત્વના ઉદાહરણને ભેદ કહેવાય છે, જે લખાણ ફાઈલોની સરખામણીમાં યુનિક્સ આદેશ છે. તેથી, આ હકીકતમાં વર્તે છે, શબ્દ તરીકે વાક્ય બે ફાઇલો દ્વારા રેખા. તેથી, દરેક લીટીની બીજી લીટીમાં તેની સરખામણી કરવામાં આવે છે જો લીટી મેચ તેઓ સમાન ગણાય અને ફાઇલના અન્ય સંસ્કરણ સાથે ફાઇલના એક સંસ્કરણની તુલના કરવા માટે આ સારો માર્ગ છે. ધારો કે તમે કોઈ બીજા સાથે કોઈ દસ્તાવેજ અથવા પ્રોગ્રામ પર સહયોગ કરી રહ્યાં છો અને તમે તેને ઇમેઇલ દ્વારા મોકલો છો અને તેઓ તેને દ્વારા મોકલે છે. તેથી, તેઓએ કેટલાક ફેરફારો કર્યા છે અને ત્યારબાદ તમને ઝડપથી જણાવાયું છે, તમે મોકલેલ ફાઇલ વચ્ચે જે તફાવતો છે અને તમે જે ફાઇલની પાછળ જાઓ છો અને ભેદમાં આવશ્યક છે તે જ કરવાનું છે તે જ વસ્તુ જે તમે મોકલેલ છે તે ફાઇલ વચ્ચે સૌથી લાંબી મેચ શોધવાનો પ્રયાસ કરી રહ્યા છો. ફાઇલ જે તમે પાછો મેળવ્યો છે અને ટૂંકી રસ્તો છે કે જેમાં તમે થોડા લીટીઓમાં ફેરફાર કરીને એકને બીજામાં પરિવર્તિત કરી શકો છો. આ સૌથી લાંબી સામાન્ય અનુગામી સમસ્યાના અમુક વિશિષ્ટ ઉદાહરણ છે અને તેથી, તે સામાન્ય રીતે સૌથી લાંબી સામાન્ય ઉપશબ્દ સમસ્યામાં વ્યવહારમાં ઉપયોગી છે.

સૌથી લાંબી સામાન્ય અનુગામી સમસ્યાનું ઇન્ડિવિડુઅલ માળખું શું છે? પહેલાં આપણી પાસે શબ્દો છે. તેથી, આપણે કહી શકીએ કે 0 થી લઘુત્તમ અથવા ઓછા -1 ની બાદબાકી 1, તે તમને કેવી રીતે પસંદ કરે છે તે વાંધો નથી, પરંતુ ચિત્રમાં તે કહે છે, પરંતુ જો તમે ઇચ્છતા હોવ કે તમે આ છેલ્લી વ્યક્તિને દૂર કરી શકો છો તેથી, એક બાદબાકી -1 એ પ્રથમ શબ્દ બી 0 થી બીએન -1 બાદ બીજો શબ્દ છે અને હવે ત્યાં બે કેસો છે. પ્રથમ કેસ એ સરળ કેસ છે, ધારો કે મારી પાસે આ બન્ને વસ્તુઓ બરાબર છે, જેમ કે પહેલાં હું 1 અને b1 ની સમસ્યાને ઉકેલવા માટે અને આને ઉમેરી શકું. હું તે ઉકેલને કહીને કહી શકું છું કે 0 મેચ b છે અને પછી ગમે તે મેચો હોય. અનુગામી શું છે, પછીથી ખરેખર કોઈ પ્રકારની મેચિંગ છે, તે કહે છે કે તમે જાણો છો કે તે આ સાથે બંધબેસે છે અને પછી આ મેળ ખાય છે, આ સમાન છે અને આ મેચ આ છે અને પછી આ મેચ આ છે અને તેથી ચાલુ માત્ર વસ્તુ એ છે કે

આ રેખાઓ વાળવું ન કરી શકે, તેઓ બાથ વગર, ડાબેથી જમણી તરફ જવાનું રાખવું જોઈએ. તેથી, આ પ્રકારની સમાન અક્ષરોને અપનાવીને, મહત્તમ રીત જે ફરીથી કરવા માટે છે તે સૌથી લાંબી સામાન્ય અનુગામી છે. હવે, આપણે જે કહીએ છીએ તે છે કે જો હું પહેલા બે વસ્તુઓ સાથે મેળ કરી શકું તો મને તેમની સાથે મેળ ખાતો હોવો જોઈએ અને પછી હું આગળ જઈ શકું છું અને બાકીના સાથે મેળ ખાતો હોઈશ, અને કારણ ખૂબ જ સરળ છે, શ્રેષ્ઠ ઉકેલની ધારણા તમે દાવો કરો કે શ્રેષ્ઠ ઉકેલ માટે વાસ્તવમાં આવશ્યકતા જરૂરી છે તે 0 અને બી 1 સાથે મેળ ખાય છે. સારું, જો હું 0 અને b 1 સાથે મેળ કરી શકું તો હું તેને પૂર્વવત્ પણ કરી શકું છું અને 0 અને b 0 સાથે મેળ કરી શકું અને ત્યારબાદ ચાલુ રાખો કારણ કે 0 અને b 1 જો તેઓ મેળ ખાતા હોય અને 1 જો મેચ બરાબર છે તેથી, હું તે ઉકેલ લઈ શકું છું અને તે ઉકેલમાં ફેરફાર કરી શકું છું જ્યાં 0 થી 0 બી ધરાવે છે. પ્રથમ બે અક્ષરો સમાન છે અને તે સાથે સારી રીતે જઈ શકે છે અને કહે છે કે તે એક વત્તા છે, બાકીનામાં શ્રેષ્ઠ રીતે ઉકેલીને પરિણામ શું છે જો તેઓ એ જ નથી તો આ રસપ્રદ કેસ છે ધારો કે, તે સમાન નથી પછી શું થાય છે. પછી આપણે આગળ વધારીએ અને 0 અને b 0 ને અવગણી શકીએ, કોઈ અધિકાર નથી. તેથી, તે હોઈ શકે કે 0 વાસ્તવમાં b1 સાથે મેળ ખાય છે અથવા તે હોઈ શકે છે કે જે b 0 થી મેળ ખાય છે, આપણે જાણતા નથી, પરંતુ અમે ચોક્કસપણે જાણીએ છીએ કે 0 બી 0 સાથે મેળ ખાતો નથી. તેથી, આપણે તેમાંના એકને છોડવું પડશે. કારણ કે અમે કોઈ ઉકેલને 0 થી મેળ ખાતો નથી, પરંતુ અમને ખબર નથી કે કઈ છે. તેથી, આપણે શું કરીએ એ બે ઉપ સમસ્યા છે, આપણે કહીએ છીએ કે બી 0 એ ઉકેલનો ભાગ નથી, તો શ્રેષ્ઠ ઉકેલ 0 થી એક અને બાદ 1 અને બી 1 થી બીએન થાય છે, બી 0 એ બાકાત નથી કારણ કે હું તે 0 સાથે મેળ ખાતો નથી અને 0 મેચ ગમે તે જમણી સાથે બંધબેસતી હોવી જોઈએ. તેથી, મને તેની સાથે આગળ વધવું જ જોઈએ. પરંતુ કદાચ આ ખોટી પસંદગી છે. બીજો વિકલ્પ બી 0 રાખવો અને 0 છોડવું અને કયા કેસમાં હું 1 થી 1 ઘાત એ 1 અને બી 0 થી બ મીટર થાય. આ બે અલગ અલગ પસંદગીઓ છે જે એક પસંદ કરવા માટે છે, કારણ કે અમને ખબર નથી કે અમે બંનેને હલ કરીએ છીએ. અમે એઆઈએ 0 બી નથી તો અમે હલ કરી શકીએ છીએ કે આ બંને સમસ્યાઓનો ઉકેલ 1 થી શૂન્ય 1 બી 0 અને 0 થી લઘુત્તમ 1 બી 1 છે, તેમાંથી એકનો મહત્તમ ઉકેલ લાવો જે કોઈ પણ સારું છે તે એક છે.

સામાન્ય રીતે આ શબ્દોમાં અમને વધુ ઊંડા મળશે. તેથી, અમે કહ્યું હતું કે 0 બી 0 ને તેને 1 અને બી 0 અથવા એબીએ 0 અને બી 1 માટે હલ કરવાની જરૂર છે. તેથી, સામાન્ય રીતે આપણી પાસે એઆઈ અને બીજે જમણી બાજુ છે. ફરીથી એઈ અને બીજે છે ત્યારથી તમે એ જ તર્કનો ઉપયોગ કરો છો જો એઈ બીજે સમાન હોય તો તે એક વત્તા બાકી છે. તેથી, આ સારો કેસ છે, જો એઆઈ બીજેની બરાબર નથી, તો આપણે શું કરીએ છીએ તે જ વસ્તુને આપણે જોઈએ છીએ, આપણે બીજે છોડીએ છીએ અને તેને ઉકેલવા અને સમપ્રમાણરિતે અમે એઈ છોડીએ અને તેને ઉકેલવા અને બેમાંથી વધુ સારી રીતે લઈએ. અમે i અને સોલ્યુશનમાંથી મોટાભાગના સોલ્યુશનને j વત્તા 1 માંથી લઈએ છીએ. જો આપણે કહીએ છીએ કે અમારી પાસે ij ની lcs પહેલા i અને j થી શરૂ થતી સૌથી લાંબી સામાન્ય લંબાઈ છે, જો એઆઈ બીજે બરાબર હોય તો તે એક વત્તા હશે લંબાઈ હું તેને 1 વત્તા 1 વત્તા 1 વડે શરૂ કરી શકું છું. જો તે બરાબર ન હોય તો તે બે ઉપ સમસ્યાઓની મહત્તમ હશે જ્યાં ક્યાં તો ઇન્ડીમેન્ટ આઈ અથવા ઇન્ડીમેન્ટ જ હશે અને જ્યારે આપણે છેલ્લી પોઝિશન પર જઈશું અને n ને 0 મળશે

તેથી, અહીં ડિપેન્ડન્સી થોડી વધારે જટીલ છે કારણ કે કેસ પર આધાર રાખવો, મને ક્યાં તો i વત્તા 1 j વત્તા 1 પર નજર કરવી પડશે અથવા મને આઈ વત્તા 1 જે કે આઈજે વત્તા -1. તેથી, હું આ સ્કેવર માટે હતો, મેં તેના જમણા પડોશી, બરાબર વિકર્ણ પડોશી અને નીચલા પડોશીને જોયું હતું, પરંતુ ફરી એક વખત જે કોઈ નિર્ભરતા ન હોય તે દેખાય છે. તેથી, પહેલાં આપણે સૌથી લાંબો સામાન્ય ઉપદેશ માટે રાખ્યું હતું, અમારી પાસે ફક્ત આ નિર્ભરતાનો અર્થ હતો કે આનો કોઈ પણ અવલંબન કોઈ નિર્ભરતા નથી કારણ કે તેના તળિયે જમણે કંઈ નથી. પરંતુ હમણાં, દાખલા તરીકે, જો આપણે આ ચિત્રને જોઉં તો, કારણ કે આપણે નીચે જમણી અને ડાબી બાજુ જોઈ રહ્યા છીએ, જો હું આને જોઉં તો તેની નિર્ભરતા ત્રણ દિશાઓમાં છે; બે દિશાઓ ખાલી છે, પરંતુ આ દિશામાં પરાધીનતા છે. તેથી, હું આ સ્કેવરને સીધી રીતે ખાલી કરી શકતો નથી, માત્ર ચોરસ કરી શકું છું, હું આ સીધું જ ભરી શકું છું કારણ કે તેની ત્રાંસી દિશામાં તેની કંઈ જ નથી અને નીચે કંઈ નથી.

તેથી, હું ત્યાંથી શરૂ કરું છું અને હું 0 મુકું છું અને તે પહેલાં આપણે નીચે જઈ શકીએ છીએ, કારણ કે હવે એક વાર અમારી પાસે આ છે, અમારી પાસે બધું જ બાકી છે અને એકવાર આપણી પાસે આ છે અને કારણ કે આપણે આ શબ્દની બહાર છે જ્યાં મીટર, આ 5મી પોઝિશન, પંક્તિ અને સ્તંભ 0 ની બનેલી છે, પરંતુ યાદ રાખવા માટે મહત્વની બાબત એ છે કે પંક્તિ અને કોલમ 0 નહીં કારણ કે તેમની પાસે કોઈ નિર્ભરતા નથી, પરંતુ કારણ કે આપણે વ્યવસ્થિત રીતે, તેને ભરી બરાબર જેમ કે ગ્રીડ ભાગોમાં આપણે નીચેની પંક્તિ અને ત્યાં ડાબી સૌથી વધુ કોલમ ભરી શકીએ છીએ, અહીં સૌથી વધુ જમણા કોલમ. હવે, એકવાર આપણી પાસે આ છે, તો આપણે આ ભાગને ભરી શકીએ છીએ અને પછી ફરીથી આપણે બે અને ત્રણ કેટેગરીઓ ધરાવી શકીએ છીએ. આપણી પાસે ત્રણ એન્ટ્રીઓ છે જે આપણે ભરી શકીએ છીએ, અમારી પાસે ત્રણ એન્ટ્રીઓ છે જે આપણે ભરી શકીએ છીએ અને આ સ્તંભ ભરી શકીએ છીએ અને આપણે આ સ્તંભ કરી શકીએ છીએ અને આપણે આ સ્તંભ દ્વારા કોલમ કરી શકીએ છીએ અને આપણે તે પ્રગટ કરીએ છીએ અને છેવટે, મૂલ્ય આ પ્રચાર અહીં સૌથી લાંબી સામાન્ય પ્રગતિની સૌથી લાંબી લંબાઈ છે, અમે પંક્તિ દ્વારા આ પંક્તિ પણ કરી શકીએ છીએ.

હવે, જ્યારે ઉકેલ ઉગે છે ત્યારે આપણે વાસ્તવિક ઉકેલને કેવી રીતે શોધી શકીએ, જ્યારે પણ આપણે સંખ્યા વધારીએ? તો આપણે કહી શકીએ કે શા માટે આ 4 છે? તેથી, આપણે કહીએ છીએ કે આ 4 નથી કારણ કે આપણે વત્તા 3 કર્યું છે કારણ કે s એ b ની બરાબર નથી, આપણે 4 કર્યું કારણ કે આપણને અહીંથી મહત્તમ મૂલ્ય મળી છે, શા માટે આ 4 ફરીથી હું s ની બરાબર નથી. તેથી, આપણને અહીંથી મહત્તમ કિંમત મળી છે, આ 4 શા માટે છે, ઓ ઓ બરાબર x છે. આપણે તેને 3 વત્તા 1 વડે મેળવવું જોઈએ, આ કેમ છે કારણ કે ઇ વત્તા ઇ. તો, આપણે તે અહીંથી મેળવ્યું છે. તેથી, અમે દરેક ભાગોમાં મૂલ્ય પેદા કરવા માટે અમે પ્રાયોગિક વિધેયને અમલમાં મૂકી તે પસંદગીઓ અનુસાર પાથને અનુસરીએ છીએ. બીજા શબ્દોમાં કહીએ તો, દરેક સેલ આઈજે માટે આપણે લખીએ છીએ કે આપણે તે લખ્યું છે કે આપણે તે લખ્યું છે કારણ કે તે એક વત્તા છે જે કર્ણ પડોશી અથવા જમણે ડાબી બાજુએ મહત્તમ છે, જેમાં આપણે રેકોર્ડ કરીએ છીએ કે ડાબી કે નીચે તે મહત્તમ છે. હવે, ચિત્રમાં દરેક વખતે આપણે કર્ણ પગલું લઈએ છીએ એનો અર્થ એ કે અમારી પાસે એક મેચ છે. તેથી, આ તે મેચ છે જે અહીં પ્રથમ મેચ છે, જે મેચ માટે સમાન છે, આ મેચ ઇ સમાન છે, આ મેચ c બરાબર c છે. હવે, આ બિંદુ પછી અમે સપાટ છીએ અને પછી આ બિંદુએ ફરીથી અમારી પાસે મેચ છે. તેથી, આપણે સેકંડ અને ટી મેળવીએ છીએ. તેથી, આપણે આ પ્રકારની કર્ણના પગલાઓ વાંચી શકીએ છીએ જે આપણને સૌથી લાંબો નંબરની સૌથી મોટી સંખ્યાના સમજૂતી મળી છે અને દરેક કર્ણના પગલાં અંતિમ ઉકેલમાં ફાળો આપશે, હવે 1 થી વધુ હોઇ શકે છે, કારણ કે અમને આમાં કોઈ ઉદાહરણ મળ્યું નથી. કેસ છે, પરંતુ જો મહત્તમ ડાબી તરફ જમણી પાડોશી લે છે અને તે જ તે સમાન હોઈ શકે છે, તો ઘણીવાર મહત્તમ બંને દિશામાં એક હોઈ શકે છે મને એવું લાગે છે કે હું અહીં ઉતર્યો છું, મને ખબર નથી કે મને તે અહીંથી અથવા અહીંથી મળી છે. તેથી, મને બે અલગ અલગ એક્સટેન્શન્સ હોઈ શકે છે જે મને ઉકેલ તરફ દોરી જાય છે. તેથી, સૌથી લાંબી સામાન્ય આવશ્યકતા અનન્ય હોવાની જરૂર નથી, પરંતુ તમે આ પાથને અનુસરીને ઓછામાં ઓછો એક પુનઃપ્રાપ્ત કરી શકો છો

તો, અહીં પાચથોન કોડ પહેલાના એકથી અલગ નથી. આપણે જોઈ શકીએ છીએ કે આપણે છેલ્લા સ્તંભની છેલ્લી પંક્તિ અને નીચેની પંક્તિને પ્રારંભ કરી છે અને તે પહેલાં તમે પંક્તિ દ્વારા હરોળ, સ્તંભ દ્વારા સ્તંભ અને સમીકરણનો ઉપયોગ કરીને આગળ વધતા પહેલા અને આ કિસ્સામાં, અમારે ટ્રેક રાખવા નથી. મહત્તમ મૂલ્ય અને અપડેટ રાખવાનું કારણ કે મહત્તમ કિંમત આપમેળે 0 0 મૂલ્યનો પ્રચાર કરે છે.

સૌથી લાંબો સામાન્ય સ્યુડવર્ડની જેમ, અહીં ફરી એકવાર આપણે માપ મીટર n નો ટેબલ ભરી રહ્યા છીએ. દરેક એન્ટ્રીમાં તમારે ફક્ત ત્રણ અન્ય એન્ટ્રીઝ જોવાની જરૂર છે. તેથી, નીચે એક જમણા ખૂણે જમણી બાજુ એક તેથી, તે કામ સતત છે તેથી, એમ.એન. એન્ટ્રીમાં પ્રવેશ દીઠ સતત સંખ્યામાં કામ કરે છે, આને લીધે સમયનો એમ સમય લાગે છે.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રોફેસર. માધવન મુકુન્ડ ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ,
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી,
મદ્રાસ વીક - 08
લેકચર - 04

મેટ્રિક્સ ગુણાકાર ગતિશીલ પ્રોગ્રામિંગને સમજાવવા માટે આ અંતિમ ઉદાહરણ છે. અમે મેટ્રિક્સ ગુણાકારની સમસ્યાની તપાસ કરીએ છીએ.

જો તમને યાદ છે કે મેટ્રિક્સ ગુણાકાર કેવી રીતે કાર્ય કરે છે, તો તમારે સુસંગત પંક્તિઓ સાથે બે પંક્તિઓ, બે મેટ્રાસો લેવાની રહેશે, અને પછી અમે એક નવી મેટ્રિક્સની ગણતરી કરીએ છીએ જેમાં દરેક પ્રવેશ માટે, આપણે એક પંક્તિને ગુણાકાર કરવી પડશે અહીં એક જ લંબાઈના સ્તંભ દ્વારા અને મૂલ્યો ઉમેરો. જો આપણી પાસે મેટ્રિક્સ છે જે m દ્વારા n છે તો આ n વખત p હોવું જોઈએ, જેથી આપણી પાસે અંતિમ જવાબ છે જે m વખત p છે. અંતિમ પ્રવેશમાં, અમારે અંતિમ ઉત્પાદન મેટ્રિક્સમાં એમપી એન્ટ્રી કરવી પડશે; અને આમાંની દરેક એન્ટ્રીઝ, આ n એન્ટ્રીઝની આ રકમની ગણતરી કરવા માટે અમને જરૂરી છે, જેથી અમને ઓર્ડર એન સમય લાગી શકે. કુલ કામ સાથે, અમને ગણતરી વખત સામાન્ય રીતે સરળ એમ વખત n વખત p . તેથી, એબી આઈજે આ લાંબુ રકમ છે અને આ રકમ 1, 2, 3 એ એન એન્ટ્રીઓ સુધી છે. બે મેટ્રિક્સ માટે મેટ્રિક્સ ગુણાકારના ગણતરીમાં ખૂબ સીધી આગળ અલ્ગોરિધમનો સમાવેશ થાય છે જે ઉત્પાદન ત્રિવિધ નેસ્ટેડ લૂપ છે, જે કમના એમ વખતનો વખત લે છે, જો તમામ પરિમાણ સમાન હોય તો આ ઓર્ડર એન સમય અલ્ગોરિધમ છે. હવે તે કરવાના વધુ હોશિયાર રીત છે, પરંતુ તે આ વ્યાખ્યાનો હેતુ નથી, પરંતુ બે સાદડીઓને ગુણાકાર કરવાના નિષ્કપટ સીધાં, માર્ગ એમ ગુણાંક વખત છે. અમારી રુચિ છે જ્યારે અમારી પાસે આવું ગુણાકારનો ક્રમ છે.

ધારો કે, આપણે ત્રણ મેટ્રિક્સિસને એકસાથે ગુણાકાર કરવા માગીએ છીએ, એક વખત B વખત C, તો પછી તે તારણ બતાવે છે કે આપણે પ્રથમ એબી વડે ગુણાકાર કરીશું અને પછી C ગુણાકાર કરીશું અથવા આપણે પ્રથમ A ને ગુણાકાર કરીશું અને પછી BC ને ગુણીશું, એક વખત ઇ.સ. પૂર્વે, કારણ કે આ એસોસિએટીવ તરીકે ઓળખાતું હતું જેમાં આપણે ગુણાકારનું જૂથ છે તે ફક્ત સામાન્ય નંબરો માટે વાંધો નથી. જો આપણે 6 ગુણ્યા 3 ગુણ્યા 2 કરીએ, તો તે કોઈ વાંધો નથી કે તમે 6 વખત 3 પ્રથમ, અથવા 3 વખત 2 છેલ્લે છેલ્લે, ઉત્પાદન એ સમાન બનશે. તેથી, બ્રેકેટિંગ એ જવાબને બદલી નાંખે છે કે અંતિમ મૂલ્ય એક જ છે, પરંતુ તે તારણ કાઢે છે કે તેનો જવાબ ગણતરીની જટિલતા પર નાટ્યાત્મક અસરો હોઈ શકે છે. શા માટે આ કેસ છે

ધારો કે, આ મેટ્રીસીસ A, B અને C છે, અને A અને B પાસે આ સ્તંભો અને પંક્તિઓ છે. A ને મૂળભૂત રીતે 1 થી 100 મળ્યું છે, એમાં 1 પંક્તિ અને 100 કોલમ છે અને B માં 100 પંક્તિઓ અને 1 કોલમ છે. અને સી ફરીથી 1 પંક્તિ અને 100 કોલમ છે. આ મેટ્રીસીસ છે જે આના જેવો દેખાય છે. હવે શું થાય છે કે જ્યારે હું C વખત ગુણાકાર કરું છું તો મને કંઈક 100 માં 1 થી 100 માં મળે છે. તેથી, આપણે એક આઉટપુટ મેળવીશું જે 100 by 100 મેટ્રિક્સ છે, તેથી તે 10000 એન્ટ્રીઝ છે, તેથી તે લેશે. અમને 10,000 પગલાંઓ હવે જ્યારે હું એ દ્વારા ગુણાકાર કરું છું ત્યારે હું 1 થી 100 માં 100 લઈશ, જે બીજું 10000 પગલાં છે. જો હું એ કરું છું, હું બીસી કર્યા પછી અને હું 10000 વત્તા 10000 કરું છું, તેથી હું 20000 પગલાં કરું છું. હવે જો હું તે બીજી રીતે કરું, જો હું પહેલા A વખત B લેતો હોઉં તો આ આખી વસ્તુ 1 બાય 1 સિંગલ એન્ટ્રીમાં તૂટી જશે. તેથી, મને 1 થી 100 માં 1 થી 100 પગલાં મળે છે, હું આ પંક્તિને એક જ એન્ટ્રીમાં પતન કરું છું જે એક મેટ્રિક્સ ગુણાકારમાં એક એન્ટ્રી કોમ્પ્યુટિંગ જેવું છે, પરિણામી વસ્તુ બરાબર તે એક એન્ટ્રી છે. હવે મારી પાસે આ એક એન્ટ્રી છે, અને ફરીથી મને આ વસ્તુ દ્વારા ગુણાકાર કરવો પડશે અને તે મને દરેક કોલમ માટે લઈ જશે, મને એક એન્ટ્રી મળશે, જેથી મને 100 પગલાં લઈ જશે. આ A માં B ને એક કોષમાં તોડવા માટે મેં 100 પગલાં લીધાં છે અને તે પછી બીજું 100 પગલાં તે ઉત્પાદનને સીમાં ગણશે. તેથી 20000 પગલાંને બદલે, મેં 200 પગલાંમાં તે કર્યું છે. આ તે રીતે છે કે

જેમાં ગુણાકારનો અનુક્રમણિકા સાંધાયુક્ત છે અને તે કોઈ બાબત નથી જે તમે કરો છો તે જ જવાબ મળશે; જે અનુક્રમણિકા તમે સહયોગી પગલાઓ કરો છો તે નાટ્યાત્મક રીતે તમે જે સમય પસાર કરો છો તે નાટ્યાત્મક રીતે સુધારો અથવા વધુ ખરાબ થઈ શકે છે.

સામાન્ય રીતે, અમારી પાસે અનુક્રમમાં M_1 થી M_n છે અને તેમાંની દરેક પાસે કેટલીક પંક્તિઓ અને કોલમ્સ છે, અને અમે જે ખાતરી આપી છે તે છે કે દરેક સંલગ્ન જોડીને ગુણાકાર કરી શકાય છે. તેથી, આર 1 સી 1, આર 2 સી 2 એ પ્રથમ બે છે, પછી C 1 બરાબર આર 2 છે, પ્રથમ મેટ્રિક્સમાં કોલમોની સંખ્યા બીજા મેટ્રિક્સની સંખ્યા પંક્તિઓ જેટલી છે, તેવી જ રીતે, સી 2 બરાબર છે આર 3 અને તેથી વધુ. આ પરિમાણો મેળ ખાતાં ખાતરી છે, તેથી મેટ્રિક્સ ગુણાકાર હંમેશાં શક્ય છે. અમારું લક્ષ્ય છે તે શોધવાનું છે કે અમે તે શું કરીશું. તેથી, આપણે એક સમયે બે મેટ્રીસીસ કરી શકીએ છીએ, આપણે માત્ર એ જાણીએ છીએ કે A વખત B કેવી રીતે ગુણાકાર કરવા, આપણે ત્રણ મેટ્રીસેસ લઈ શકતા નથી અને તેમને સીધી રીતે ગુણાકાર કરી શકતા નથી. જો આપણે ત્રણ મેટ્રીસીસ કરવું હોય તો, આપણે બે તબક્કામાં A વખત B અને પછી C, અથવા B વખત C અને પછી એ જ કરવું પડશે. હવે n એ matrices સાથે સમાન રીતે, આપણે એક સમયે બે કરવું પડશે, પરંતુ તે બે સમય જટીલ બાબત બની શકે છે. હું એમ 1 2 કરી શકું છું, પછી હું તે ભેગા કરી શકું છું અને તે સંયોજન 3 સાથે અથવા હું એમ 1 2, એમ 3 4 કરી શકું છું અને પછી M 3 2 નું મિશ્રણ M 3 4 દ્વારા અને તેથી આગળ વધવું. ઉત્પાદનની ગણતરી કરવા માટે શ્રેષ્ઠ માર્ગ શું છે, અન્ય શબ્દોમાં કૌંસને મૂકવાનો શ્રેષ્ઠ માર્ગ શું છે? કૌંસ આપણે શું કહીએ છીએ, તેથી જ્યારે આપણે એમ 1, એમ 2, એમ 3, એમ 4 કહીએ છીએ ત્યારે આ જ કરવા માટે ફક્ત એમ 1, એમ 2, પછી એમ 3, એમ 4 અને બીજી રીત છે. આમ કરવું એમ એમ 1, એમ 2 કહેવું હશે અને તે પછી એમ 3 અને પછી એમ 4 અને તેથી વધારે હશે. તેથી, બ્રેકેટીંગના વિવિધ માર્ગો આ ગુણાકાર માટે જુદા જુદા મૂલ્યાંકન ઓર્ડરના અનુરૂપ છે. અને તમે શું કરવા માગો છો તે ખરેખર ગણતરી વગર જે પ્રકારની ગણતરી છે તે શ્રેષ્ઠ ક્રમ છે અને જે આ ગણતરી કરવા માટે છે, જેથી અમે તેમાં સામેલ કામગીરીને ઓપ્ટિમાઇઝ કરી શકીએ.

ઇન્ડિવિડુઅલ માળખું શું છે? છેલ્લે, જ્યારે આપણે આ યાદ કરીએ છીએ ત્યારે અમે ફક્ત એક જ સમયે બે કરીએ છીએ. અંતમાં, આપણે ગુણાકાર સાથે સમાપ્ત કરવું જોઈએ જેમાં કેટલાક જૂથ દ્વારા ગુણાકાર કરાયેલા કેટલાક જૂથને આવશ્યક લાગે છે, અને આ સમગ્ર વસ્તુને કેટલાક 1 ની પ્રાઇમની તૂટેલી હોવી જોઈએ, અને આખી વસ્તુ એમ 2 પ્રાઇમની પડી ભાંગી અને છેવટે ગુણાકાર કરીને એમ પ્રાઇમ દ્વારા 1 પ્રાઇમ અન્ય શબ્દોમાં એમ 1 પ્રાઇમ એ અમુક કી માટે છે જે M 1 થી MK સુધી બધી રીતે છે, અને M 2 પ્રાઇમ એ K વત્તા 1 થી n સુધી છે. તેથી, આ મારી M1 પ્રાઇમ છે અને આ મારી M 2 પ્રાઇમ છે, અને આ k એ 1 અને n વચ્ચે ક્યાંક છે. સૌથી ખરાબ કિસ્સામાં હું એમ 1 કરી શકે છે, અને બીજી તરફ હું આમ કરી શકું છું, હું M 2 ને આ સમગ્ર બાબત એમ એમ કરી શક્યો હોત. આ આખી વસ્તુ મારા છે, બીજો સૌથી ખરાબ કેસો હું એમ 1 2 કરી શક્યો હોત, પરંતુ આત્યંતિક કેસો M 1 માં M n ની 1, મેં પહેલેથી જ ગણતરી કરી લીધી હોત, અને હવે હું છેલ્લે એમ કરવા માંગુ છું, તેને એમ એન અથવા તે વચ્ચે ક્યાંય પણ હોઈ શકે છે જો હું માત્ર એક મનસ્વી K પસંદ કરું તો પ્રથમ r 1 પંક્તિઓ CK કોલમ છે. તેથી, આરક વત્તા એક પંક્તિઓ cn કોલમ તરીકે બીજા, પરંતુ આપણે જાણીએ છીએ કે ck બરાબર rk plus 1 છે. તેથી આ મેટ્રિક્સ કાર્ય કરશે. અંતિમ ગણતરીને સી 1 માં cn માં સી.એન. માં જ હશે, m માં n માં p - સ્તંભમાં પંક્તિઓ, કોલમની અંતિમ સંખ્યામાં કોલમ પંક્તિની સામાન્ય સંખ્યા. તો આ અંતિમ ગુણાકાર, આપણે જાણીએ છીએ કે તે કેટલો ખર્ચ કરશે. અને આ માટે, આપણે પુનરાવર્તિત બે પરિબલોને ગણતરીમાં લેવાના આગ્રહી ખર્ચને ઉમેરવાની જરૂર છે; તે અમને એમ 1 પ્રાઇમ કરવા માટે કેટલો સમય લેશે તે અમને એમ 2 પ્રાઇમ કરવા માટે કેટલો સમય લેશે.

અમારી પાસે છે કે K પર M 1 સ્પ્લિટિંગની કિંમત M 1 થી MK ની કિંમત MK વત્તા એકથી એમ n વત્તા વત્તા છેલ્લા ગુણાકાર r1 માં ck થી c ની કિંમત છે. n. તેથી, સ્પષ્ટ રીતે આ કિંમત જુદા જુદા કેસમાં બદલાશે, અને ઘણા બધા કિસ્સાઓ છે. અમે જણાવ્યું હતું કે, n ની બાદબાકી 1 પસંદગીઓ છે, 1 થી n ની 1 ઘાત અમે કંઈપણ તરીકે પસંદ કરી શકીએ છીએ. ત્યાં n બાદ 1 પેટા સમસ્યાઓ છે. તેથી, જ્યારે અમે સૌથી લાંબી સામાન્ય પેટા અનુક્રમ સમસ્યા કરી, અમારી પાસે બે પેટા સમસ્યાઓ હતી. આપણે ક્યાં તો પ્રથમ અક્ષર A_i અથવા બીજો અક્ષર બીજે છોડો, અને પછી આપણે બે સબ સમસ્યાઓને ધ્યાનમાં રાખવી પડશે. અમારે જાણવાની કોઈ રીત

નથી કે જે વધુ સાડું છે, તેથી અમે તેમને બજાએ કર્યું અને મહત્તમ લીધો. હવે અહીં આપણી પાસે n ની 1 અલગ અલગ પસંદગીઓ છે, અમારી પાસે આ જાણવાની કોઈ રીત નથી કે આ કઈ બાબતમાં વધુ સાડું છે. તેથી, ફરીથી અમે TRવાય અને તે બધા અને ઓછામાં ઓછા લેવા ત્યાં અમે મહત્તમ કરી રહ્યા હતા કારણ કે અમે સૌથી લાંબી સામાન્ય પ્રગતિ માંગીએ છીએ, અહીં અમે લઘુત્તમ ખર્ચ કરવા માંગીએ છીએ તેથી અમે કે જે પસંદ કરીએ છીએ કે જે આ વિભાજિતને ઓછું કરે છે, અને તે બજા સ્ક્રીપ્ટને ઘટાડે છે, તેથી, તે પ્રત્યક્ષ સમૂહ છે

છેલ્લે, અમે એમ કહીએ છીએ કે M 1 થી M n નો ગુણાકાર એ M 1 થી M K વત્તા ગુણાકારની કિંમતની K ની બધી પસંદગીઓ માટે ન્યૂનતમ વત્તા 1 નો વધારો કરવાની કિંમત છે. M n વત્તા માટે અલબત્ત, તે માટે કેવું પસંદ કરો, આપણે એક અંતિમ ગુણાકાર કરવાની જરૂર છે જે આ પરિણામી સબ મેટ્રિક્સ લેવાનું છે, જેથી તે આર 1 માં c n માં c n થાય. તેથી, જ્યારે આપણે આ લાગીએ છીએ, તેથી અમારી પાસે M 1 થી M n છે, તેથી અમે M 1 થી M K ની પસંદગી કરી છે. પછી શું થશે તે પહેલાં આપણે આને ક્યાંક વિભાજિત કરવું પડશે, તેથી હવે અમે કેટલાક સેગમેન્ટ સમાપ્ત કરીશું જે ન તો એમ 1 કે એમ એન છે, તે કેટલાક એમ પર શરૂ થાય છે અને એમ કે જાય છે. સામાન્ય રીતે, આપણે મનસ્વી ડાબી અને જમણી બિંદુ માટે આ જથ્થો વ્યક્ત કરવાની જરૂર છે, અમે એમ ન ધારણ કરી શકીએ કે ડાબા હાથનું પોઇન્ટ 1 છે; આપણે ધારી શકતા નથી કે જમણા હાથનો મુઠો એ સાચો છે.

સામાન્ય રીતે, જો અમારી પાસે એમ આઇ થી એમ જ સેગમેન્ટ હોય, તો આપણે i થી j ની માપદંડ 1 ની તમામ કિંમતો વચ્ચે સૌથી નાનો મૂલ્ય જોઈએ છે, આપણે લઘુત્તમ કિંમત જોઈએ છે જે એમ આઇ થી એમ કે, અને એમ કે વત્તા 1 થી એમજે, અને અંતિમ ગુણાકારની કિંમત, જે સીજે માં સી.આઇ. આ જથ્થો આપણે લખીશું ખર્ચ આઇજે કિંમત આઇજે એ એમ આઇથી એમજે સુધીની સેગમેન્ટની ગણતરી કરવાની કિંમત છે, જેમાં શ્રેષ્ઠ કે. તેથી, એમ i થી એમજેને કિંમત ij કહેવામાં આવે છે, અને અમે આ જ આવર્તક આગવી વ્યાખ્યાને શ્રેષ્ઠ પસંદ કરીએ છીએ.

બેઝ કેસ સારી રીતે જો આપણે માત્ર લંબાઈના સેગમેન્ટ પર જોઈ રહ્યા હોય, તો આપણે ધારવું કે આપણે એક મેટ્રિક્સ 1 થી 1, અથવા 3 થી 3, અથવા 7 થી 7 ને ગુણાકાર કરવા માંગીએ છીએ. થવું તે માત્ર મેટ્રિક્સ છે ત્યાં કોઈ ગુણાકાર નથી, તેથી કિંમત 0 છે. પછી અમે આ કિંમતને લખી શકીએ છીએ ij સમીકરણ જે કહે છે કે લઘુત્તમ કિલો ખર્ચે ik વત્તા કિંમત k વત્તા 1 j વત્તા રિક્ષજ જે વાસ્તવિક ગુણાકાર છે. અને અલબત્ત, હું હંમેશાં જ કરતાં ઓછું અથવા બરાબર હોઈશ, કારણ કે અમે તેને ડાબેથી જમણે કરી રહ્યા છીએ, તેથી આપણે ધારણ કરી શકીએ છીએ કે સેગમેન્ટ અમને બે અંતિમ બિંદુઓ સાથે આપવામાં આવ્યું છે જ્યાં હું ઓછું અથવા સમાન છે j

તો, હું જે કરતા ઓછું અથવા બરાબર જ છું, અને અમે ખર્ચ આઇજેને એક પ્રકારની મેટ્રિક્સ તરીકે જોઉં છું તો આ સમગ્ર વિસ્તાર જ્યાં હું j કરતા વધારે છે તે બહાર નકારી શકાય છે. તો, આપણે ફક્ત આ ત્રાંસા પર જોશું. અને કર્ણ સાથેના પ્રવેશો એ છે કે જે ફોર્મ મેં અલ્પવિરામ i છે, તેથી આ બધી એન્ટ્રીઓ શરૂઆતમાં શૂન્ય છે. આ કર્ણ સાથે કોઈ પણ ગુણાકાર કરવાથી પોઝિશન j થી પોઝિશન j કરવામાં કોઈ ખર્ચ નથી. હવે, હું અલ્પવિરામ ગણતરી કરવા માટે, મને એક પસંદ કરવાની જરૂર છે, અને મને તે ગણતરી માટે ગણતરી કરવાની જરૂર છે જેનો અર્થ હું મારી ગણતરી કરું છું IC ને ગણતરી કરાવવું પડશે, અને તેથી તે ચાલુ થાય છે તે કહે છે કે જો હું ઇચ્છું છું ચોક્કસ એન્ટ્રી i કોમમા જ ગણિત કરો, તો પછી મને એક સારા કે. અને સારા કેય પસંદ કરવા માટે, મને આની જરૂર છે જેથી હું ઘણી અલગ રીતે વ્યક્ત કરી શકું. હું પિક કહી શકું છું, ઉદાહરણ તરીકે, ધારો કે આ ચોક્કસ વસ્તુની ગણતરી કરવા માગે છે તો મને કહેવું પડશે કે આ એન્ટ્રી i ને પસંદ કરો અને પછી હું એન્ટ્રી i વત્તા 1 થી જ કરવા માંગુ છું. તેથી, મારી પાસે 1 થી 1 આ એન્ટ્રી છે. આ બે એન્ટ્રીઓને હું ટૂંકમાં વર્ણવીશ; અન્યથા, મને આ એન્ટ્રી લેવાની અને આ એન્ટ્રી સાથે સરભર કરવું પડશે. સામાન્ય રીતે, જો મારી પાસે વસ્તુ હશે તો, હું કહીશ કે જો હું આ પ્રવેશ પસંદ કરું છું તો હું તેનો પ્રવેશ મેળવવા માટે આ પ્રવેશ ઉમેરવો જોઈએ, અને આ રકમ લેવી પડશે અથવા મને આ એન્ટ્રી લેવા પડશે અને આ એન્ટ્રી ઉમેરો અને પછી આ અને તેથી પર ઉમેરો સામાન્ય રીતે, આપણી પાસે ઓર્ડર એન મુલ્ય હોઈ શકે જે મને ગણતરી માટે જરૂર છે, જે મને ગણતરી કરવાની જરૂર છે, મારે અહીં તમામ મુલ્યો જોઈએ છે અને મને અહીં તમામ મુલ્યો ની જરૂર છે.

મને ડાબી અને તળિયે કંઈક જોઈએ છે, પણ જો હું કર્ણમાં શરૂ કરું તો તે સરળ બને છે, કારણ કે હું ખરેખર કહી શકું છું કે જો મારી પાસે શરૂઆતમાં આ મૂલ્યો ભરવામાં આવે તો, આ છે આધાર કેસ પછી ડાબા અને નીચે, હું આને ભરી શકું છું કારણ કે મને ખબર છે કે તે ડાબી બાજુનું મૂલ્ય છે, હું આ મૂલ્યને ડાબી અને નીચેથી જાણું છું, તેથી હું આ કર્ણ ભરી શકું છું. આગળના પગલામાં, હું આ કર્ણ ભરી શકું છું, કારણ કે મારી પાસે નીચેનાં ડાબે બધા મૂલ્યો છે. હવે આ એન્ટ્રી પર, મારી પાસે ડાબી અને નીચેનું મૂલ્ય છે, તેથી હું આ કર્ણ ભરી શકું છું. તેથી, હું તેને વિકર્ણ દ્વારા ત્રાંસા ભરી શકું છું. આ એ ક્રમ છે કે જેમાં હું આ ટેબલને આ પ્રત્યક્ષ વ્યાખ્યા દ્વારા ભરી શકું છું કારણ કે આ તે માર્ગ છે કે જેમાં અવલંબન સંગ્રહિત છે.

આ ચોક્કસ વસ્તુ માટે આ કોડ છે તેથી, તમે તેમાંથી પસાર થઈ શકો છો અને ફક્ત તે જ વસ્તુની તપાસ કરો કે જેનો અર્થ છે કે અમે કેટલાકનો ઉપયોગ કર્યો છે.

તેથી આપણે શું કરી રહ્યા છીએ, જ્યારે આપણે એક એન્ટ્રી કહીએ છીએ ત્યારે ગણતરી કરીએ છીએ. ધારો કે, અમે આ એક એન્ટ્રી ગણતરી કરી રહ્યા છીએ, તો આપણે લઘુત્તમ ઘણાં જુદાં જુદાં ગણવા માંગીએ છીએ, આ પ્રવેશ આ એન્ટ્રીમાં અને તેથી આગળ. યાદ રાખો કે જ્યારે અમે મહત્તમ લાંબી સામાન્ય પેટા શબ્દની ગણતરી કરી ત્યારે શું કર્યું, અમે ધારીએ છીએ કે મહત્તમ શૂન્ય છે, અને દર વખતે અમે એક મોટી મૂલ્ય જોયું છે જે અમે તેને અપડેટ કર્યું છે. અહીં, અમે ન્યૂનતમ એન્ટ્રી જોઈએ છે. તેથી, આપણે શું કરીએ છીએ તે લઘુત્તમ જેટલી મોટી સંખ્યા છે અને દર વખતે જ્યારે આપણે પ્રવેશ જોએ, તો તે લઘુત્તમ કરતાં ઓછો હોય તો આપણે તેને ઘટાડીએ છીએ.

અહીં આવી રહ્યું છે, જ્યારે આપણે લૂપ શરૂ કરીએ છીએ ત્યારે આપણે ધારીએ છીએ કે આરસી માટેનું મૂલ્ય ખરેખર અનંત છે. હવે અનંત શું છે? સારી રીતે તમે અનંત લઈ શકો છો જેથી અમે દાખલા તરીકે આ સમસ્યામાં દેખાતા તમામ પરિમાણોનું ઉત્પાદન લઈ શકીએ. તમે જાણો છો કે કુલ પરિમાણ તે કરતાં વધુ નહીં હોય. તેથી, તમે તમામ પરિમાણોનો ઉત્પાદન લઈ શકો છો જે તમે ખૂબ મોટી સંખ્યામાં લઈ શકો છો, તે તમારી સમસ્યાની સાથે સંબંધિત કોઈ બાબતને વાંધો નથી. તેથી, આપણે તેને કોડમાં વ્યાખ્યાયિત કર્યું નથી. મહત્વની બાબત એ છે કે આપણે લઘુત્તમ ગણતરી કરીએ છીએ. 0 થી શરૂ કરવાને બદલે, અને જ્યારે તમે મહત્તમ કરો ત્યારે તેને અપડેટ કરો; તમે મોટા મૂલ્યથી શરૂ કરો છો અને તેને સંકોચાય તો રાખો તેથી દર વખતે, આપણે પેટા સમસ્યા શોધીએ છીએ જે વર્તમાન મૂલ્ય કરતાં નાની છે જે આપણે જોયેલી છે, તો આપણે તે મૂલ્યને ન્યૂનતમ તરીકે બદલીએ છીએ, તેથી તે અહીં મહત્વનું છે. બાકીનું બધું જ તે ખાતરી કરવા માટે એક રસ્તો છે કે આપણે તેને જઈએ, વિકર્ણ દ્વારા આ કોષ્ટક કર્ણને પસાર કરીએ, અને દરેક કર્ણ માટે આપણે પંક્તિ અને સ્તંભને સ્કેન કરીએ છીએ અને તે પંક્તિ અને સ્તંભમાં તમામ જોડીઓમાં ન્યૂનતમ ગણતરી કરીએ છીએ.

આ એલસીએસ સમસ્યાની જેમ, આપણે ઓર્ડર એન સ્કવેર્ડ માપદંડ ભરવાનું રહેશે, પરંતુ એલસીએસ વચ્ચેનો મુખ્ય તફાવત છે અને આ એ છે કે એલસીએસ વસ્તુમાં એન્ટ્રીને ભરવા માટે આપણે જોવાની જરૂર છે પ્રવેશોની માત્ર એક જ સંખ્યામાં તેથી, એમએન અથવા ઓર્ડર એન સ્કવેર્ડનો ઓર્ડર કરો, પરંતુ બિંદુ એ દરેક એન્ટ્રી સતત સમય લે છે, તેથી તેમાં સામેલ કોષ્ટકનું કદ, એમએન ટેબલ અથવા એમએન ટાઇમ લે છે. અહીં, કમનસીબે, આ કેસ યોગ્ય નથી. અમે જોયું છે કે પ્રવેશને પ્રમાણસર લાગશે, તે કર્ણમાંથી અંતર છે. સામાન્ય રીતે, તે ઓર્ડર એન ફેક્ટર ઉમેરશે. જોકે અમારી પાસે બે એન્ટ્રીઓ દ્વારા ઓર્ડર એન સ્કવેર્ડ છે, વાસ્તવમાં એન સ્કવેર્ડ એન્ટ્રીઝની ક્રમમાં ગોઠવે છે, અમારે દરેક એન્ટ્રી દીઠ એન્ટ્રી માટે ખાતું ખોલાવવું પડશે કારણ કે દરેક એન્ટ્રીને તેની પંક્તિને સ્કેન કરવી પડશે. અને તેથી આ આખી વસ્તુ ક્રમમાં ગોઠવાય છે અને ક્રમમાં n ચોરસ નથી. તેથી, આ એ ધ્યાનમાં રાખવાનું છે કે સમસ્યા હોઈ શકે છે, જ્યાં કોષ્ટક કદ કરતાં જટિલતા મોટી છે. આ ઉદાહરણ પહેલાં આપણે જે ઉદાહરણો જોયા છે તે આ ઉદાહરણનું કારણ છે. બધા ઉદાહરણોમાં આપણે પહેલાં જોયું, ફિબોનાકી અમારી પાસે એક ટેબલ છે જે n માં રેખીય છે અને તેને ભરવા માટે n માં સમય રેખીય લીધો હતો. ગ્રીડ પથ

માટે અને સૌથી લાંબી સામાન્ય પ્રગતિ માટે, આપણી પાસે કોષ્ટકો હતાં, જે n દ્વારા મીટર છે, પરંતુ તે માત્ર n એ મીટર સમય અથવા m એ n ભરવા માટે n સમય લીધો. અહીં, આપણી પાસે એક કોષ્ટક છે જે n દ્વારા n છે, પરંતુ તેને પૂર્ણ કરવા માટે n સમઘન સમય લે છે કારણ કે દરેક પ્રવેશને સતત સમય કરતાં વધુ જરૂરી છે, વાસ્તવમાં તે n નો સમય પ્રમાણસર લે છે.

પ્રોગ્રામિંગ, ડેટા સ્ટ્રક્ચર્સ એન્ડ એલ્ગોરિધમ્સ ઇન પાયથોન
પ્રો. માધવન મુકુન્દ ડિપાર્ટમેન્ટ ઓફ કમ્પ્યુટર સાયન્સ એન્ડ એન્જિનિયરિંગ
ઇન્ડિયન ઇન્સ્ટિટ્યુટ ઓફ ટેકનોલોજી, મદ્રાસ

વીક - 08

લેક્ચર - 05

સમેટો, પાયથોન વિ. બીજી ભાષાઓ

અમારી પાસે આ અભ્યાસક્રમના છેલ્લા પ્રવચનમાં આવો. તેથી, પાયથોન ની વધુ લાક્ષણિકતાઓ પર જવાને બદલે જેનું વર્ણન અમે કર્યું નથી. ચાલો આપણે પાઠાણને ભાષા તરીકે પ્રસ્તુત કરવા માટે થોડો સમય લઈએ અને અન્ય કેટલીક ભાષાઓમાં તેની સરખામણી કરીએ. તેમ છતાં તમે તેમની સાથે પરિચિત ન હોઈ શકે, હું પાયથોન કેટલાક પાસાઓ પ્રકાશિત કરવા માંગો છો, જે અન્ય ભાષાઓ કરતાં અલગ છે અને દલીલ કરે છે કે શું તેઓ વધુ સારી અથવા ખરાબ છે અમે આ કોર્સ કરવા માટે પાયથોન શા માટે પસંદ કર્યું? વેલ, ઘણા કારણોસર પ્રોગ્રામિંગ શરૂ કરવા માટે પાયથોન એ ખૂબ સારી ભાષા છે. એક કારણ એ છે કે આ વાક્યરચના ખૂબ જ સરળ છે અને સરળતાનો ભાગ એ હકીકત પરથી આવે છે કે તમારે અગાઉથી નામો જાહેર કરવાની જરૂર નથી, તેથી તમારે તમારો કોડ લખતા પહેલા ઘણી બધી વસ્તુઓ કરવાનું શરૂ કરવાની જરૂર નથી. તમે દાખલા તરીકે, પાયથોન ઇન્ટરપ્રીટરમાં બાંધી શકો છો, અને તમે શું કરી શકો છો તે વિશે ચિંતા કર્યા વગર નિવેદનો લખી રાખો, જેમ તમે જાઓ છો તેમ તમે મૂલ્યો નિર્ધારિત કરી શકો છો અને તેમને ચાલાકી કરી શકો છો. તેથી, તે ખૂબ અનુકૂળ અને પ્રોગ્રામિંગ શરૂ કરવા માટે ખૂબ જ સરસ બનાવે છે કારણ કે તમે પણ મૂળભૂત કાર્યક્રમો લખવા માટે ક્રમમાં ઘણી વસ્તુઓ શીખવા માટે નથી જે રીતે પ્યોથોન પ્રોગ્રામ્સ ઇન્ટરેક્શન સાથે મૂકવામાં આવે છે તે દર્શાવવા માટે જ્યારે લૂપ શરૂ થાય છે અને જ્યારે `if` બ્લોક પ્રારંભ થાય છે. અમે ઘણી વિરામચિહ્નો વિશે ચિંતા કરવાની જરૂર નથી: કૌંસ, કૌંસ, અર્ધવિરામ, જે વધુ પરંપરાગત ભાષાઓના ભાગ અને પાર્સલ છે. ફરી એક વાર આ ભાષા શીખવા માટે સહેલી સરળ બને છે અને નવા મુલાકાતીને જોવા માટે થોડું ઓછું પ્રયત્ન છે. બીજી બાબત જે થોડી વધુ તકનીકી છે, જે અમે કરીશું ... હું આ વ્યાખ્યાનમાં થોડુંક વિશે વાત કરું છું કે અમારે સ્ટોરેજ વિશે ચિંતા કરવાની જરૂર નથી, જે પરિવર્તનક્ષમ અને અસંબદ્ધ છે. જો તમને કોઈ નામ x વાપરવાની જરૂર હોય, તો અમે તેનો ઉપયોગ કરીએ છીએ; જો આપણને લિસ્ટની જરૂર હોય તો અમે તેનો ઉપયોગ કરીએ છીએ, જો આપણને એલ માટે કંઈક ઉમેરવાની જરૂર પડતી હોય, તો આપણે ફક્ત ઉમેરવું કહીએ છીએ. જગ્યા ક્યાંથી આવે છે અથવા ક્યાં જઈ રહી છે તે વિશે અમે ક્યારેય ચિંતા કરતા નથી. આ બધા પાયથોનનાં પ્લસ પોઇન્ટ છે. તો, ઓછા બિંદુઓ શું છે? તેથી એવી વસ્તુઓ છે કે જે પાયથોન વિશે ખરાબ છે?

ઘોષણાઓની અભાવ સાથે પ્રથમ વસ્તુ એ છે - ઘોષણાઓની અભાવ ઘણી સારી વાત છે, કારણ કે તમારે પ્રોગ્રામિંગ શરૂ કરતા પહેલાં ઘણી બધી સામગ્રી લખવાની ચિંતા કરવાની જરૂર નથી, પરંતુ તે પણ ખરાબ વસ્તુ છે કારણ કે ઘણા પ્રોગ્રામિંગ ભૂલો સરળ ટાઇપિંગ ભૂલોમાંથી આવે છે. વારંવાર તમને x લખવાની જરૂર છે અને તમે y લખો છો. હવે તમે પાયથોન માં જો લખી શકો છો અને તમે તેને ક્યાંક મૂલ્ય નિર્ધારિત કરો જ્યાં તમે x નો અર્થ હોવો જોઈએ, પાયથોન એ નથી જાણતા હશે કે તમે x નો ઉપયોગ કરવા માગતા હતા અને y નથી કારણ કે દરેક નવું નામ જે સાથે આવે છે તે ફક્ત નામના પરિવારમાં ઉમળકાભર્યું ઉમેરે છે જે તમારા પ્રોગ્રામનું સંચાલન કરે છે. આ પ્રકારના ટાઇપોઝ ખૂબ જ મુશ્કેલ થઈ શકે છે, અને કારણ કે તમારી પાસે કોઈ પ્રકારની જાહેરાતો સાથે ગતિશીલ નામ પરિચય નથી, પાયથોન તમારા માટે આ ભૂલોને શોધવામાં પ્રોગ્રામર તરીકે ખૂબ જ મુશ્કેલ બનાવે છે. બીજી બાજુ જો તમે આ નામોને અગાઉથી પ્રગટ કરો છો જે C અથવા C++ અથવા Java જેવી અન્ય ભાષાઓમાં થાય છે. પછી જો તમે કોઈ નામનો ઉપયોગ કરો છો જેને તમે જાહેર કર્યું નથી, તો કમ્પાઇલર તમને જણાવે છે કે આ નામ પહેલાં જોવામાં આવ્યું નથી અને તેથી કંઈક ખોટું છે. તેથી, જો તમારી પાસે ઘોષણાઓ છે તો એક અવિવેકી નામ તરીકે સરળતાથી ગુમ થયેલ નામને સરળતાથી અકબંધ નામ તરીકે પકડી શકાય છે. જ્યારે કે, પાયથોન માં તે ઉમળકાભેર આગળ વધશે અને તે નામ માટે એક નવું મૂલ્ય બનાવશે અને ડોળ કરવો જોઈએ કે હવે ત્યાં બે નામો છે, જ્યારે તમને લાગે છે કે ફક્ત એક જ છે અને તમારા પછીના કોડમાં તમામ પ્રકારની અનિશ્ચિત ભૂલો બનાવો. આની બીજી બાજુ ટાઇપીંગ છે. તેથી, પાયથોન માં આપણે જોયું છે કે નામોમાં એવા પ્રકારો નથી કે તેઓ માત્ર પ્રકારનાં પ્રકારને પ્રાપ્ત કરે છે જે તેમની પાસે છે. તમે અમુક બિંદુએ કહી શકો છો x બરાબર 5 અને પાછળથી સ્ટ્રિંગને x સોંપી દો અને બાદમાં સૂચિમાં x સોંપી દો અને પાયથોન ફરિયાદ નહીં કરે, તો x વેરિએબલ ઓપરેશન્સની વર્તમાન કિંમત આપેલ દરેક બિંદુએ તે મૂલ્ય પ્રમાણે વ્યાખ્યાયિત થયેલ છે. હવે આ ફરીથી સરસ અને અનુકૂળ છે પણ તે જ કારણથી તમે x ને પૂર્ણક તરીકે વિચારી

રહ્યા હો તેવી ભૂલો માટે પણ પરિણામી શકે છે, પરંતુ તમારા પ્રોગ્રામ દ્વારા તમે અડધી રીતે ભૂલી ગયા છો કે તે એક પૂર્ણાંક છે અને તેને અમુક જુદી જુદી પ્રકારની કિંમત આપવી શરૂ કરો . હવે જો તમે પાચથોન પર જાહેરાત કરી હતી કે એક્સ હંમેશા પૂર્ણાંક હોવું જોઈએ તો આ નામ ફક્ત પૂર્ણાંક મૂલ્યને સંગ્રહિત કરવું જ જોઈએ, સંભવતઃ એક કમ્પાઇલર તરીકે તે આંતરિક રીતે તેને પકડે છે ઘણી બધી ભૂલો વેરિયેબલ નામોમાં ટાઇપોઝ અથવા નામોનો એક પ્રકારથી બીજા પ્રકારનો ઉપયોગ થાય છે, આ બન્નેને કમ્પાઇલરો દ્વારા સરળતાથી પકડવામાં આવે છે જો તમારી પાસે નામોની જાહેરાત હોય, તો આ બન્ને અનંત થઈ જાય છે અથવા તેઓ પાચથોન અને તેઓ તમને ભૂલો પ્રગટ કરવા માટે પરવાનગી આપે છે અને આ ભૂલો શોધવા માટે ખૂબ જ મુશ્કેલ હોઈ શકે છે તેથી નામો ઉમેરવાની અને તેમના પ્રકારો બદલાવવાની આ સુગમતા હોવાના નીચે બાજુ છે, તમે જેટલા મૂલ્યનો સામનો કરો છો તે એ છે કે મોટા કાર્યક્રમોને ડીબગ કરવાથી તમારા ભાગ પર વધુ કાળજી લેવી જરૂરી છે, મોટા કાર્યક્રમો લખવાની કાળજી જરૂરી છે, ડિબગીંગ ખૂબ મુશ્કેલ છે

અન્ય ભાગમાં ચર્ચા સાથે શું કરવું છે કે જે વર્ગો અને ઓબ્જેક્ટ્સની દ્રષ્ટિએ યુઝર નિર્ધારિત ડેટા પ્રકારો વિશેના કોર્સના બીજા ભાગમાં છે. તેથી, એવી જાહેરાતની સીધી પરિણામની પ્રથમ વસ્તુ એ છે કે આપણે કહી શકીએ નહીં કે નામનું પ્રકાર છે. ખાસ કરીને અમે જોયું છે કે જો આપણે કંઈક યાદીમાં ઉપયોગ કરવા માગીએ છીએ તો આપણે તેને ખાલી યાદી જાહેર કરવી પડશે, તેથી આપણે કંઈક લખવું પડશે, જેમ કે `x` . એ બરાબર છે એમ કહેવું છે કે તેથી આગળ હું તેને ઉમેરી શકું છું. સૂચિમાં પ્રથમ એન્ડ અથવા પ્રથમ ઓપરેશન કાયદેસર બનવું પડશે, તે માટે મને કહેવાની જરૂર છે કે તે એક યાદી છે. આ ઘોષણા કરતાં વધુ કે ઓછું છે સિવાય કે તે ઘોષણા નથી. હું ખરેખર તે પ્રકારના ખાલી ઓબ્જેક્ટ બનાવું છું. એ જ રીતે જો હું શબ્દકોશ ઇચ્છું તો મને આના જેવું સંપૂર્ણપણે નવું નામ આપવું પડશે. મને કહેવું છે કે ડી એક ખાલી શબ્દકોષ સમાન છે. તેથી, તે પ્રકારનો ઓબ્જેક્ટ બનાવ્યાં વિના નામ આપવા માટે કોઈ રીત નથી. વાસ્તવમાં તે પ્રકારની વ્યાખ્યાયિત વપરાશકર્તા પ્રકારની સમસ્યા છે જે અમારી પાસે છે, દાખલા તરીકે, એક વૃક્ષ, નોડ બનાવવા વગર ખાલી વૃક્ષને વ્યાખ્યાયિત કરવા માટે તે ખૂબ અનુકૂળ છે હમણાં પૂરતું, જો તમારી પાસે પ્રકારો ઘોષણાઓ હોય તો તમે કહી શકો છો કે નામ ટી પ્રકારનું વૃક્ષ છે અને પછી તમે આ મૂલ્યનો ઉપયોગ કોઈની જેમ કરી શકો છો - બધી પ્રોગ્રામિંગ ભાષાઓમાં એવું મૂલ્ય છે જે અસ્તિત્વમાં નથી તેવી કોઈ વસ્તુને સૂચવે છે. તમે કહી શકો છો કે ત્યાં એક પણ નથી, પરંતુ ઘણાં નાનો અને સંદર્ભ દ્વારા આ કોઈ પ્રકારનું વૃક્ષ નથી. પાચથોન પણ તેનો ઉપયોગ કરે છે ત્યાં માત્ર એક જ મૂલ્ય નથી અને તમે કોઈ પણ વસ્તુ માટે કંઈ પણ ઉપયોગ કરી શકતા નથી, પરંતુ જ્યારે તેની પાસે કોઈ નથી, તો તેનો કોઈ પ્રકાર નથી - તે તફાવત છે. જો પાચથોન માં નામ કોઈ મૂલ્ય ધરાવતું નથી, તો તેની પાસે વ્યાખ્યા કોઈ પ્રકાર નથી, પરંતુ જો તમે જાહેરાત કરી હોય તો તમે કહી શકો છો કે આ નામનો પ્રકાર છે, તે મૂલ્ય ધરાવતી નથી. અને આ સામાન્ય રીતે તમે ખાલી નોડ અથવા એક ખાલી વૃક્ષ માટે શું કરવા માંગો છો તે છે. અમને યાદ છે જો તમને યાદ કરાવવું હોય તો આ સાથે વ્યવહાર કરવા માટે ખૂબ જ કષ્ટદાયક રસ્તો યાદગાર વૃક્ષ સંશોધનની કામગીરી વધુ સારી બને છે, અમે વાસ્તવમાં સરહદ પર ખાલી ગાંઠો એક સ્તર ઉમેર્યા છે અને અમારા વૃક્ષને એક સ્તરથી વિસ્તૃત કર્યો છે જેથી અમે સરળતાથી શોધી શકીએ કે જ્યારે અમે પાથ ઓવરને અંતે પહોંચે છે. હવે આ વાસ્તવમાં અવગણી શકાય છે, જો તમારી પાસે પ્રકારો ઘોષણાઓ છે, તો આ એક બીજું લક્ષણ છે જે વાસ્તવમાં બનાવે છે ... ઘોષણાના અભાવથી વસ્તુઓ પાચથનમાં થોડી વધુ જટીલ બને છે જે સામાન્ય રીતે શરૂઆત પ્રોગ્રામિંગમાં આવતી નથી. બીજી વસ્તુ વધુ ગંભીર છે. તેથી, આ ખાલી વસ્તુઓની સગવડ અને પ્રતિનિધિત્વ કરવા માટે વધુ છે, પરંતુ જાહેરાત વિના તમે ખરેખર કોઈ પણ વસ્તુમાં જાહેર અને ખાનગી વિચ્છેદને અમલમાં મૂકી શકતા નથી. એક અમૂર્ત ડેટા પ્રકારમાં અમારો ધ્યેય યાદ રાખો કે જાહેર ઇન્ટરફેસ અથવા વિધેયોનો સમૂહ છે જે ડેટા પ્રકાર માટે કાનૂની છે અને ખાનગી અમલીકરણ ધરાવે છે. હમણાં પૂરતું, અમારી પાસે એક સ્ટેક હશે, અમે એક સૂચિ તરીકે અમલ કરી શકીએ છીએ, પણ અમે ફક્ત પોપ અને દબાવણને પસંદ કરીશું. એવી જ રીતે કતારમાં પણ સૂચિ હોઈ શકે છે, પરંતુ અમે ફક્ત વિરામના અંતમાં કતાર ઉમેરો અને દૂર કરવા માંગીએ છીએ અને અમે આ સૂચિ ઓપરેશન્સ સાથે મૂંઝવણ કરવા નથી માગતા; અમે વસ્તુઓને ઉમેરવાની જરૂર નથી અને અણધારી રીતે ઉપયોગમાં લેવા માટે વિસ્તૃત નથી. તેનો બીજો ભાગ એ છે કે આપણે માહિતીને પોતાને સુલભ ન થવા જોઈએ, આપણે એમ કહેવા નથી માગતા કે જો કોઈ બિંદુ p આપવામાં આવે, તો હું p ડોટ એક્સ અને પી ડોટ y નો ઉપયોગ બહારથી કરી શકું છું અને કિંમતોને સીધી અપડેટ કરી શકું છું. કારણ કે, આ હકીકતને સંવેદનશીલ છે કે આવતીકાલે હું x ડોટ y ... x અને y પ્રતિનિધિત્વને આર અને થીટામાં બદલી શકીશ, અમે કહ્યું છે કે અમારી પાસે એવા પરિસ્થિતિઓ હોઈ શકે છે કે જ્યાં અમે r અને θ નો

ઉપયોગ કરીને અમારી બિંદુ પ્રતિનિધિત્વ કરવાનું પસંદ કરી શકીએ. હવે જો કેટલાક પ્રોગ્રામરે દિવસોનો ઉપયોગ કરવાનું શરૂ કર્યું છે જ્યારે આપણે x અને y નો ઉપયોગ કરી રહ્યા છીએ અને કોડની બહાર p ડોટ એક્સ અને પી ડોટ y નો ઉપયોગ શરૂ કર્યો છે. જુઓ કે તે ક્લાસ કોડની અંદર છે, પછી વર્ગના જાળવનારા તરીકે આપણે ખાતરી કરીશું કે જ્યારે આપણે $p \cdot x$ અને $p \cdot y$ નો ઉપયોગ કરીએ છીએ, તો આપણે હવે $p \cdot r$ અને p ડોટ થીટા નો ઉપયોગ કરીએ છીએ. તેથી, તે આંતરિક વસ્તુ છે જે આપણે x , y થી r , θ થી બદલીએ છીએ, આપણે આંતરિક વર્ગના તમામ કોડને આર, થિટા અને નહીં x , y ની દ્રષ્ટિએ અદ્યતન રીતે અપડેટ કરીએ છીએ. પરંતુ જો કોઈ બહારના એક્સ અને વાયનો ઉપયોગ કરી રહ્યો છે અને આ મૂલ્યો લાંબા સમય સુધી અસ્તિત્વમાં નથી તો પછી શું થાય છે તે બહારના વ્યક્તિ માટે તેમનું કોડ કામ કરવાનું બંધ કરે છે, કારણ કે અમે પોઈન્ટનું આંતરિક અમલીકરણ બદલ્યું છે. અને આ એક ખૂબ જ ખતરનાક પરિસ્થિતિ છે જે ઘણીવાર થાય છે, અને આ તે છે જ્યાં ખાનગીને ખાનગીમાંથી અલગ કરવા માટે મહત્વપૂર્ણ છે જો તેમની પાસે બિંદુના ખાનગી અમલીકરણનો કોઈ પ્રવેશ ન હોય તો તેઓ પી ડોટ એક્સ અને પી ડોટ વાયનો ઉપયોગ કરી શકતા નથી. આ સમસ્યાની બહાર ટાળવામાં આવે છે. તેથી માત્ર પુનરુક્તિ કરવી, ધારો કે અમારી પાસે સૂચિ તરીકે સ્ટેક લાગુ છે અને અમે ફક્ત જાહેર પદ્ધતિઓને પુશ અને પોપ કરવાની મંજૂરી આપીએ છીએ, જે વ્યક્તિ આ હકીકતનો શોષણ કરે છે કે તે એક સૂચિ છે જે અંતમાં કંઈક સીધી જ ઉમેરી શકે છે અને ઉદાહરણ તરીકે ઢગલો મિલકતનું ઉલ્લંઘન કરી શકે છે. અથવા સ્ટેક મિલકત. તેથી, અમે એવા પરિસ્થિતિઓ મેળવી શકીએ છીએ જ્યાં ડેટા માળખું સમાધાન થયું છે કારણ કે વ્યક્તિ કાર્યરત છે જે ખાનગી અમલીકરણ માટે કાનૂની છે, પરંતુ અમૂર્ત ડેટા પ્રકાર માટે ગેરકાયદેસર છે જે અમે વપરાશકર્તાને રજૂ કરવાનો પ્રયાસ કરી રહ્યા છીએ. તેથી આ માટે, આપણે આની આસપાસ જ એકમાત્ર રસ્તો મેળવી શકીએ છીએ તે વાસ્તવમાં કહેતા હોય છે કે આ નામો ખાનગી છે અને આ નામો જાહેર છે. હવે તે એ નથી કે પાયથોન પાસે ઘોષણા નથી. જો તમને યાદ છે કે પાયથોન પાસે આ વૈશ્વિક ઘોષણા છે જે તમને ફંક્શનની અંદર અસક્ષમ મૂલ્ય લેવાની પરવાનગી આપે છે અને કહે છે કે તે જ અસમર્થ મૂલ્યની બહાર છે. એવી પરિસ્થિતિઓમાં કે જેમાં પાયથોન ઘોષણાઓ કરવાની પરવાનગી આપે છે, પરંતુ ત્યાં ઘણી અન્ય વસ્તુઓ છે કે જ્યાં તે જાહેરાતોની પરવાનગી આપી શકે છે અને વસ્તુઓને વધુ ઉપયોગી બનાવી શકે છે, પરંતુ તે એક નથી અને આ એક ઉદાહરણ છે. જાવા અથવા જેવી ભાષાઓમાં ... તમને જાહેર અને જાહેર જનતાને ઘોષણા ઘોષણા મળશે, અને આ ખૂબ જ અમલદારશાહી જુએ છે પરંતુ ઈન્ટરફેસમાંથી અમલીકરણને બહાર કાઢવા માટે તે જરૂરી છે. વાસ્તવમાં, આદર્શ વિશ્વમાં, અમલીકરણ સંપૂર્ણપણે ખાનગી હોવું જોઈએ. તેથી, તમે ક્યારેય સી અને વાય સીને સીધી નજર કરી શકશો નહીં. દાખલા તરીકે, જો તમે એક્સ ઈચ્છતા હોવ, તો તમને x મળે તે ફંક્શન હોવો જોઈએ, જે x મૂલ્ય આપે છે, સેટ x નામનું ફંક્શન છે જે x મૂલ્ય સુયોજિત કરે છે. આ ઉપરીક્ષરીતે કહી શકે છે કે p ડોટ x બરાબર v છે, પરંતુ તફાવત એ છે કે આપણે ખરેખર જાણતા નથી કે x છે, x એ આપણા માટે અમૂર્ત ખ્યાલ છે. તો, એક્સ સંકલન માત્ર છે ... તે બિંદુની મિલકત છે જે આપણે સેટ કરી શકીએ છીએ. હવે આપણે આ ફંક્શન દ્વારા કેવી રીતે સેટ કરીએ છીએ. તેથી, જો કાર્યોની અંદર આપણે આર એટીએને રજૂઆત તરીકે શરૂ કરીએ, તો પછી x મૂલ્યને બદલીને અનુલક્ષીને આર અને થિટા પરોક્ષ રીતે ફેરફાર થશે. તેથી, જ્યારે આપણે કહીએ x ને મળે, ઉદાહરણ તરીકે, તે વાસ્તવમાં x મૂલ્યને વાંચતું નથી, તે આર કોસ થીટા મળે છે અને આપણે કહીએ છીએ કે તે નવા એક્સ માટે આરને બદલશે અને થિટાને ફરી કમ્પટ કરશે. આ રીતે જો અમારી પાસે આ કાર્યોને એક્સેસ કરવા માટે ફક્ત આ વિધેયો છે તો આપણી અંદર માત્ર એક જ કલ્પનાત્મક મૂલ્યો છે અને આ વિભાવનાઓ દ્વારા આ કાલ્પનિક મૂલ્યોનો ઉપયોગ થાય છે અને અમને વાસ્તવિક પ્રતિનિધિત્વ ખબર નથી. તેથી આ આદર્શ વિશ્વ છે, બધું છુપાયેલું છે અને અમારી પાસે ફક્ત આ કાર્યો છે, પરંતુ ડેટા પ્રકારનાં દરેક ભાગ માટે આ બોજારૂપ છે કે આ વિધેયોનો ઉપયોગ કરવો પડશે. અને આંશિક કારણ કે આ ખાનગી જાહેર ઘોષણા હોવી જોઈએ, એ છે કે પ્રોગ્રામરો હંમેશાં કોઈ કાર્યનો ઉપયોગ કરવાથી ખુશ નથી, ક્યારેક તેઓ સીધી રીતે સોંપી શકે છે. તેઓ જાહેર કરવા માટે ડેટા પ્રકારનાં કેટલાક ભાગોને પસંદ કરવા માગે છે. તેથી, તેઓ p ડોટ x ઇકવલ ટુ 7 કહી શકે છે. પરંતુ સામાન્ય રીતે આ શૈલી એવી છે કે જે કોઈ ઓબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગ માટે આદર્શ રીતે વકીલ કરે છે - બધી આંતરિક નામો અને વેરિએબલ્સ ખાનગી બનાવો અને માત્ર પ્રતિબંધિત એક્સેસની પરવાનગી આપે છે, જેથી તેનો ઉપયોગ કરવામાં આવે. યોગ્ય રસ્તો અને આંતરિક પ્રતિનિધિત્વમાં ફેરફાર થાય તો તેનો સમાધાન થતો નથી. જો આપણે x , y થી r , થીટા ખસેડીએ તો x મળશે અને સેટ x હજુ પણ કામ કરશે, જ્યારે p ડોટ x હવે અર્થપૂર્ણ ન હોઈ શકે. મૂલ્યોનો ઉપયોગ કરવાની આ શૈલીનું બીજું કારણ ક્યારેક તમે વ્યક્તિગત મૂલ્યોને વ્યક્તિગત રૂપે એક્સેસ કરવા માંગતા નથી. ધારો કે, અમારી પાસે એક તારીખ છે, તે તારીખ સામાન્ય રીતે એક ત્રણ ઘટક ક્ષેત્ર છે જે દિવસ, મહિનો અને વર્ષ ધરાવે છે. અને આની પાસે તેમની પોતાની શ્રેણીઓ છે: 1 થી 31 સુધીની મહિનાની માન્ય દિવસો અને મહિના 1 થી 12 સુધીની છે, પરંતુ દર મહિને 31 દિવસ નથી. એક દિવસ અને મહિનો માટે માન્ય મિશ્રણ બંને આ જથ્થા પર નિર્ભર કરે છે

અને હકીકતમાં તે વર્ષ પર પણ આધાર રાખે છે કારણ કે અમારી પાસે ફેબ્રુઆરી 29 ન હોઈ શકે સિવાય કે તે લીપ વર્ષ નથી. તેથી, જો આપણે દિવસ કે મહિનો અપડેટ કરીએ તો અમને સાવચેત રહેવાની જરૂર છે કે અમે તેને કાયદેસર રીતે અપડેટ કરી રહ્યા છીએ, અમે ફેબ્રુઆરી મહિનાથી 15 તારીખની કાનૂની તારીખથી શરૂ કરી શકીએ છીએ અને ત્યારબાદ તારીખ 15 થી 31 સુધી બદલી શકીએ અને હવે ગેરકાયદેસર તારીખ 31 ફેબ્રુઆરી છે. તેથી આપણે જે વાસ્તવમાં જરૂર છે તે એક સંયુક્ત સુધારા ઓપરેશન હોય છે જે ત્રણ ક્ષેત્રોને અલગથી અપડેટ કરવા માટે ત્રણ અલગ અલગ ઓપરેશનો પૂરી પાડવાને બદલે તમામ ત્રણ મૂલ્યો પ્રદાન કરીને તારીખને સુયોજિત કરે છે. જો તમારી પાસે આ ફંક્શન્સ હોય તો તમે હંમેશા આ વિધેયોને વ્યક્તિગત રીતે આપવા માંગતા નથી, કારણ કે મૂલ્યો વચ્ચે અમુક અવરોધો હોઈ શકે છે જેને સાચવી શકાય છે અને તમે તેને નિયંત્રણમાં રાખી શકો છો. તેમને `ess` તેથી તમે કહી શકો છો, તમે દિવસને અલગથી અને મહિનાને અલગથી સેટ કરી શકતા નથી, તમારે તેમને એક સાથે સેટ કરવું પડશે. આ રીતે બધા અમલીકરણને ખાનગી રાખવું તે એકદમ અન્ય કારણ છે, સીધા અપડેટને મંજૂરી આપતા નથી અને પછી તમે મૂલ્યોને અપડેટ કરો છો તે રીતે નિયંત્રિત કરો.

હવે ચાલો આપણે સ્ટોરેજ ફાળવણીમાં આવીએ. કેવી રીતે અમારા પ્રોગ્રામમાં ઉપયોગમાં લેવાતી નામો અને મૂલ્યો ખરેખર મેમરીમાં સંગ્રહિત થાય છે અને સંગ્રહિત થાય છે જેથી આપણે તેને જોઈ શકીએ. કારણ કે પાયથોન માં અમે નામો સાથે નામોનો ઉપયોગ કરીએ છીએ અને કિંમતો બદલાતા રહે છે, પાયથોન અગાઉથી નક્કી કરી શકતા નથી કે તેને એક પૂર્ણાંક માટે `x` માટે સ્થાનની જરૂર છે, કારણ કે આવતીકાલે આ એક્સ એક યાદી હોઈ શકે છે અને તે પણ નથી જાણો કે કયા નામો આવે છે. તેથી પાયથોનને ગતિશીલ રીતે હંમેશા સ્થાન ફાળવવું પડે છે, જેમ કે તમે કોઈ નામનો ઉપયોગ કરો છો, તેને તેના માટે જગ્યા શોધવી પડશે અને જો મૂલ્ય તેના પ્રકારને બદલે છે તો જગ્યા જરૂરિયાત બદલાઈ શકે છે. બીજી બાજુ જો મારી પાસે સ્થિર ઘોષણા હોય તો, જો હું `k` હું `k` પૂર્ણાંક છું અને `j` એ પૂર્ણાંક છે અને `k` એ એક પૂર્ણાંક છે તો કમ્પાઇલર સીધી રીતે મેગામમાં જગ્યા ખાલી કરવા માટે `i`, `j` અને `k`. હવે આ ખાસ કરીને એરે માટે ઉપયોગી છે, અમે ઉલ્લેખ કર્યો છે કે અગાઉની પ્રવચનમાં, એરે અને યાદીઓ વચ્ચે તફાવત. સ્થાયીકૃત જાહેર પરિસ્થિતિમાં, સો કદની ઝાટકો વાસ્તવમાં અવકાશ વિના સો સંલગ્ન મૂલ્યોના બ્લોક તરીકે ફાળવવામાં આવશે. આનો અર્થ એ કે હું જોઈને એરેમાં કોઈપણ એન્ટ્રી મેળવી શકું છું ... પ્રથમ મૂલ્યને જાણવું અને પછી કેટલી અવગણો છે. તો, જો હું 75 મા મુલ્ય ઈચ્છું તો, હું પ્રથમ મુલ્ય પર જઈ શકું છું અને ગણતરી કરું છું કે 75 મા મુલ્ય હશે જો હું તે ઘણા મૂલ્યો ઉપર ફ્રી જઈશ અને તે સીધી જ મળે. આ આપણને રેન્ડમ એક્સેસ એરે કહેવાતું છે. છેલ્લા ઘટક કરતાં પ્રથમ તત્વ મેળવવા માટે તે વધુ સમય લેતો નથી. સૂચિમાં જ્યારે આપણે આપણા ઓબ્જેક્ટ આધારિત અમલીકરણમાં `list` તત્વ મેળવવા માટે જોયું ત્યારે આપણે માથાથી શરૂ કરવું અને પહેલા, સેકન્ડ, તૃતીય પર જવું પડે છે, તેથી તે પોઝિશન માટે પ્રમાણમાં સમય લે છે. જો અમારી પાસે સ્થિર ફાળવણી છે તો આપણે એ હકીકતનો ફાયદો ઉઠાવી શકીએ છીએ કે આપણે ચોક્કસ રેન્ડમ એક્સેસ એરેઝ મેળવી શકીએ છીએ જે પાયથોન ખરેખર નથી. હમણાં જ કારણ કે અમારી પાસે ઘોષણાનો અર્થ એ નથી કે બધું જ સ્થાયીરૂપે જાહેર કરવામાં આવ્યું છે. પણ જાવા અને C ++ જેવી ભાષાઓમાં, અમે આ ઓબ્જેક્ટ ઓરિએન્ટલ શૈલી કરીશું જ્યાં અમારી પાસે એક ક્લાસ માટે નમૂનો હશે અને પછી આપણે આ ક્લાસનો ઓબ્જેક્ટ ગતિશીલ બનાવીશું કારણ કે વૃક્ષ અથવા સૂચિ વધે છે અને ઘટતી જાય છે, આપણે વધુ ઓબ્જેક્ટ્સ બનાવીશું અથવા દૂર કરીશું તેમને તેથી એક ગતિશીલ ભાગ પણ છે, પરંતુ તે આ પ્રકારના યુઝર ડિફોલ્ટ ડેટા પ્રકારો માટે જ છે અને સ્ટેટિક ભાગ તમામ પ્રમાણભૂત ડેટા પ્રકારોની કાળજી લે છે જે તમે ગણતરી માટે અને વિવિધ પ્રમાણભૂત બાબતો અને ચોક્કસ એરે ઘણી વખત ઉપયોગમાં લે છે. જો તમને અગાઉથી ખબર હોય, તો તમને કોઈ ચોક્કસ કદના ડેટાના બ્લોકની જરૂર છે, એરે એ યાદીઓ કરતાં વધુ કાર્યક્ષમ છે

સ્ટોરેજ ફાળવણીનો એક ભાગ તેને મળી રહ્યો છે, અન્ય ભાગ ફાળવણી છે, તેને આપીને. તેથી, આપણે જોયું કે પાયથોન માં તમે ડેલ (એલ [0]) કહીને સૂચિમાંથી એક ઘટક દૂર કરી શકો છો. સામાન્ય રીતે તમે ડેલ (x) કહીને કોઈ પણ નામનું મૂલ્ય ફાળવી શકો છો. તો શું થાય જ્યારે આપણે ... આ સ્ટોરેજ છોડીએ? જ્યારે આપણે ડેલ (x) કહીએ છીએ ત્યારે કહીએ છીએ કે જે જગ્યા હાલમાં એક્સ માટે ઉપયોગ કરી રહી છે તે હવે મૂલ્યની જરૂર નથી.

તો તે જગ્યા શું થાય છે? તે કોણ ફાળવે છે? તેવી જ રીતે જ્યારે આપની પાસે એક સૂચિ હોય, જો તમને યાદ હોય, જ્યારે આપણે સૂચિને કાઢી નાખવા માગીએ છીએ, જો અમારી પાસે વસ્તુઓ કમ હોય અને અમે એક યાદી કાઢી નાખવા માગીએ, તો અમે ખરેખર તેને કાઢી નાખી ન હતી, આપણે તેને અવગણ્યો, અમે કહ્યું કે પ્રથમ તત્વ ત્રીજા નિર્દેશ કરે છે. હવે, તાર્કિક રીતે આ વસ્તુ મારી સૂચિનો ભાગ નથી પરંતુ તે ક્યાં ગયો છે? તે ગમે ત્યાં ગયો છે અથવા તે હજી પણ મારી યાદમાં છે, અવરોધિત જગ્યા? તેથી મૃત મૂલ્યો સાથે શું થાય છે કે જે અમે ડેલનો ઉપયોગ કરીને અનસેટ કર્યા છે અથવા આપણે ઓબ્જેક્ટની અંદર લિંક્સને હેરફેર કરીને અને તેથી બાયપાસ કર્યું છે.

તેથી તે તારણ આપે છે કે પાયથોન જેવી ભાષાઓ, જાવા જેવી બીજી ભાષાઓ પણ ... આને ચલો જાહેર કરવાથી કંઈ કરવાનું નથી, તેની પાસે જગ્યા કેવી રીતે ફાળવવામાં આવે છે તેની સાથે કરવાનું છે. તેઓ કયરો સંગ્રહ કહેવાય કંઈક ઉપયોગ કરે છે. પ્રોગ્રામિંગ ભાષાના અર્થમાં કયરો મેમરીને એક વખત ફાળવવામાં આવે છે, પરંતુ તે હવે ઉપયોગમાં લેવાતું નથી અને તેથી હવે તે ઉપયોગી નથી, કારણ કે અમે તેને અમુક રીતે મેળવી શકતા નથી. તેથી, તે તે યાદીની જેમ છે: આપણી સૂચિમાં એક તત્વ હતું જે આપણે માથા પરથી મેળવી શકીએ. અમુક બિંદુએ અમે તેને કાઢી નાખ્યો, હવે આપણે તે મૂલ્ય સુધી પહોંચી શકતા નથી અને અમે તેને ફરીથી વાપરી શકતા નથી કારણ કે તેને ફાળવવામાં આવે તેવું જાહેર કરવામાં આવ્યું છે, તેથી તે કયરો છે. તેથી મોટાભાગે કયરાપેટીના સંગ્રહનું કામ કેવી રીતે થાય છે? ઠીક છે, તમે શું કરો છો તે તમે કલ્પના કરો કે તમારી પાસે તમારી મેમરી છે અને પછી તમે તમારા પ્રોગ્રામમાં ક્યાંક કેટલાક નામો, એન, એમ, એક્સ, એલ અને તેથી છે, અને તમે જાણો છો કે આ વસ્તુઓ ક્યાં છે, આ ક્યાંક અહીં છે, આ ક્યાંક છે અહીં, આ ક્યાંક અહીં છે, આ ક્યાંક અહીં છે તેથી, તમે જે નામો સાથે પ્રારંભ કરો છો અને તમે આગળ વધો છો ... તમે આ વસ્તુને ચિહ્નિત કરો છો, તમે કહો કે આ મારું છે, આ મારું છે, આ મારું છે, અને આ મારું છે. હવે આ એક સૂચિ હોઈ શકે છે, એક સૂચિ હોઈ શકે છે, તે હોઈ શકે કે તે બદલામાં યાદીમાં આગલા ઘટક સુધી નિર્દેશ કરે છે, તેથી તે અહીં જાય છે તેથી તમે આ તરફ નિર્દેશ કરો છો અને કહે છે કે આ મારું પણ છે. તે જેનું નિર્દેશન કરે છે તે નામ છે જે તેમની પોઇન્ટર મૂલ્ય ને નિર્દેશ કરે છે, તમે આને અનુસરતા રહો ત્યાં સુધી તમે યાદ રાખો કે તમે ખરેખર તમારા નામથી પહોંચી શકો છો, જેથી તમારા પ્રોગ્રામમાં નામો અને તે બધી મેમરી કે જે તેઓ પરોક્ષ રીતે પહોંચે છે. હવે હું જઈ શકું છું અને કહી શકું છું કે જે ચિહ્નિત કરવામાં આવ્યો નથી તે બધું ઉપયોગમાં નથી, તેથી હું જઈ શકું અને તેને સ્પષ્ટ રીતે મુક્ત કરી શકું. આ બીજો તબક્કો છે. તમે બધા અચિહ્નિત મેમરી સ્થાનો એકત્રિત કરો અને તેમને મફત અને આગળ વધો. હવે, આ પ્રોસેસ છે જે તમારા પ્રોગ્રામ સાથે સમાંતર ચાલે છે, અમુક સમયે તાર્કિક રીતે કહીએ તો તમારે તમારા પ્રોગ્રામને રોકવું પડશે અને મેમરીને ચિહ્નિત કરવું પડશે, ખાલી જગ્યા છોડો અને પછી તમારા પ્રોગ્રામ ફરી શરૂ કરો. તેથી ઓવરહેડ છે. સી જેવી કેટલીક ભાષાઓમાં આ કયરો સંગ્રહમાં સમાયેલ નથી. તેથી સીમાં જો તમને ગતિશીલ મેમરીની જરૂર હોય, તો યાદ રાખો કે જે બધી વસ્તુઓ તમે અગાઉથી જાહેર કરો છો તે સ્થાયી રૂપે ફાળવવામાં આવે છે, તમે તેને પૂર્વવત્ કરી શકતા નથી, તમે આગળ કહી શકતા નથી કે હું નથી તેમને જરૂર છે પરંતુ જો તમારી પાસે કોઈ સૂચિ અથવા એક વૃક્ષ હોય કે જ્યાં તમે વધતી હોય અને સંકોચાયા હોય, તો તમે સ્મરણશક્તિ માટે પૂછશો, હું એક નોડની વર્ચસ્વ મેમરી જોઈએ, પછી તમે તેને વસ્તી કરશો. જ્યારે તમે કોઈ વસ્તુને કાઢી નાખો ત્યારે તે તમારી જવાબદારી છે કે તે પ્રોગ્રામર તરીકે કહે છે કે આ મને કેટલીકવાર પાછા આપવામાં આવ્યું હતું, કૃપા કરીને તેને પાછો લો. તમારે, પ્રોગ્રામર તરીકે, ગતિશીલ મેમરી માટે પૂછવું અને વધુ અગત્યનું છે જ્યારે તમે તેનો ઉપયોગ કરી રહ્યાં ન હોવ, તે પાછો પાછા કરો. તેથી, સી આ પ્રકારના પ્રોગ્રામરે જાતે મેમરી ફાળવણીને સંચાલિત કરે છે.

આ તદ્દન ભૂલ સંભાવના છે. જેમ તમે કલ્પના કરી શકો છો ત્યાં એક સરળ કેસ હોઈ શકે છે જ્યાં એક પ્રોગ્રામર એક વૃક્ષ અથવા સૂચિમાં કાઢી નાંખો ઓપરેશન લખે છે અને મેમરીને પાછા આપવાનું ભૂલી જાય છે. તેથી, આનો અર્થ એ થાય કે દર વખતે કંઈક સૂચિ અથવા વૃક્ષમાંથી કંઈક ઉમેરવામાં આવે છે અને દૂર કરવામાં આવે છે, આવા પ્રોગ્રામમાં, તે વસ્તુ મેમરીમાં રહેતી હશે જ્યારે તેનો વાસ્તવમાં ઉપયોગ થતો નથી ત્યારે તેનો ઉપયોગ કરવો. તેથી સમયના સમયગાળામાં આ વસ્તુ ભરવાનું ચાલુ રાખશે. જો તમે ફિલિપ બાજુ લો છો અને તમે ફી મેમરી જુઓ તો તમને લાગે છે કે ફી મેમરીને પ્રવાહી ગણવામાં આવે છે પછી મુક્ત મેમરી સંકોચાઈ રહી છે. તેથી આને મેમરી લીક કહેવામાં આવે છે, મેમરી તમારી સિસ્ટમમાંથી બહાર નીકળી રહી છે, તે ખરેખર બહાર

નીકળી નથી, તે ભરવામાં આવે છે. આ પરિભાષા છે તેથી તમે કદાચ અમુક ટેક્સ્ટમાં ક્યાંક જોઈ શકો છો, શબ્દ મેમરી લીક. તે માત્ર તે જ હકીકત એ છે કે એક કાર્યક્રમમાં ફાળવવામાં આવેલી મેમરીને યોગ્ય રીતે ફાળવવામાં આવી રહી નથી જ્યારે તે ઉપયોગમાં નથી, અને આ લક્ષણ એ છે કે જે કાર્યક્રમ લાંબા સમય સુધી ચાલે છે અને તે લાંબા સમય સુધી ચાલે છે, તે વધુ અને વધુ મેમરી અપ અને સમગ્ર કાર્યક્રમ ખૂબ વધુ આગસનો બનાવે છે તેથી, પ્રદર્શન સંકોચાઈ જશે, સમય જતા રહેશે કારણ કે જગ્યા ઘટતી જાય છે. તેથી, તમામ આધુનિક ભાષાઓમાં વર્ચ્યુઅલ રીતે કચરો સંગ્રહ ઉપયોગ કરે છે કારણ કે તે ખૂબ સરળ છે. તેમ છતાં આ માર્ક સાથે રનટાઈમ ઓવરહેડ છે અને પ્રક્રિયાની પ્રકારની પ્રક્રિયાની રીત છે, લાભો કે જેને તમે ચિંતા ન કરો અને પ્રોગ્રામરને વિશ્વાસ ન કરો, જેમ કે, તમે ખાતરી કરો કે તમારી કચરો સંગ્રહ બદલે કામ કરે છે તેની ખાતરી કરીને તમે મેમરી લિક ટાળી શકો છો. પ્રોગ્રામરની સારી સમજણ પર આધાર રાખતા નથી તેની ખાતરી કરવા માટે કે જે બધી મેમરી ફાળવવામાં આવેલ છે તે વાસ્તવમાં ડિ-ફાળવેલ છે.

આ છેલ્લી લેક્ચરના અંતિમ બિંદુ માટે, ચાલો પ્રોગ્રામિંગની એક સંપૂર્ણપણે અલગ શૈલી જોઈએ જે કાર્યાત્મક પ્રોગ્રામિંગ કહેવાય છે. તેથી, પાયથોન અને અન્ય ભાષાઓ જે અમે વિશે વાત કરી છે તે છે જેને આવશ્યક પ્રોગ્રામિંગ ભાષાઓ કહેવામાં આવે છે. તેથી આવશ્યક પ્રોગ્રામિંગ ભાષા માં તમે વાસ્તવમાં વસ્તુઓ ગણતરી માટે પગલું પ્રક્રિયા દ્વારા એક પગલું આપો છો. તમે મધ્યવર્તી મૂલ્યોનો નજર રાખવા માટે નામો આપો છો, તમે વસ્તુઓને યાદીમાં મૂકો છો અને પછી તમારે મૂળભૂત રીતે મેકેનિકલ ગણતરી કરવાની જરૂર છે: આપણે તેને વધુ કે ઓછું અનુકરણ કરવું પડશે, જેથી સૂચનોનો ક્રમ. તેથી તમારે તેને અમલ કરી શકે તે પહેલાં તમારે કંઈક કે વધુ કંપ્યુટ કરવાની યાંત્રિક પ્રક્રિયા જાણવી પડશે. કાર્યાત્મક પ્રોગ્રામિંગ એ એવી વસ્તુ છે જે અમે પ્રત્યક્ષ પ્રમાણમાં વ્યાખ્યાયિત કરવાના પ્રયાસો કરે છે જે આપણે હૃદય પર સીધી જોયા છે અને તેનો ઉપયોગ ગણતરી માટેના આધાર તરીકે કરીએ છીએ. તેથી, તમે પ્રત્યક્ષ રીતે વિધેયાત્મક વિધેયોને સ્પષ્ટ રીતે ઘોષણાત્મક રીતે રજૂ કરો છો, તેથી આ એક સામાન્ય ઘોષણા છે. ઉદાહરણ તરીકે, આ એક હોસ્કેલ શૈલી ઘોષણા છે. તેથી, આ ડબલ કોલન સાથેની પ્રથમ લીટી કહે છે કે ફેક્ટોરિયલ એક ફંક્શન છે અને આ તે પ્રકાર છે, તે ઇનપુટ તરીકે પૂર્ણાંક લે છે અને આઉટપુટ તરીકે પૂર્ણાંક પેદા કરે છે. આ એવું કહી રહ્યું છે કે ફેક્ટોરિયલ એક બોક્સ છે જે આના જેવું લાગે છે. તે ints લે છે અને ints ઉત્પન્ન કરે છે, તેથી આ તમારી પાસે જે વસ્તુ છે. અને પછી તે તમને ગણતરી કરવા માટેનો નિયમ આપે છે. તે કહે છે કે 0 નો ફેક્ટોરિયલ 1 છે અને તે પછી નિયમોને ક્રમમાં સંસારિત કરવામાં આવે છે, તેથી જો તે બીજા નિયમની વાત કરે છે, તો તેનો અર્થ એ છે કે n એ 0 નથી. જો n એ 0 નથી, તો n વખત ફેક્ટોરિયલ n બાદ 1 છે જવાબ. હાસેલની વાસ્તવિક રીત ફરીથી લખવાથી છે. અમે તેમાં પ્રવેશ નહીં કરીએ, પરંતુ મુખ્ય મુદ્દો એ છે કે મધ્યવર્તી નામો વિશે અહીં કોઈ ઉલ્લેખ નથી, તે ફક્ત વિધેય અને પ્રત્યક્ષ પ્રમાણમાં અથવા ફરી યાદ આવતાં કોલ્સને પસાર કરવામાં આવેલા પરિમાણો અને તેમને કેવી રીતે એકસાથે ભેગા કરવું તે લઈ રહ્યું છે. તો અહીં બીજું ઉદાહરણ છે. જો તમે સૂચિમાં એલિમેન્ટ્સ એડ કરવા માંગતા હો, તો આ પૂર્ણાંકની સૂચિ માટે આ હોસ્કેલનો પ્રકાર છે, તેથી તે પૂર્ણાંકની સૂચિ લે છે અને પૂર્ણાંક પેદા કરે છે. તેથી, તમે કહી શકો છો કે જે સૂચિ માટે ખાલી છે, સરવાળો 0 છે અને ફરી અહીં આવે છે રકમ નથી ... તે ખાલી નથી જો તે અહીં આવે છે કારણ કે આપણે ક્રમમાં જઈએ છીએ જો પ્રથમ યાદી મેળ ખાતી ન હોય તો, બીજી સૂચિ ખાલી હોવી જોઈએ નહીં, પછી તેમાં પ્રથમ ઘટક અથવા છેલ્લો તત્વ લેવા માટે હેડ અથવા પૂંછડી જેવા કાર્યો છે. તેથી, જો તમારી પાસે બિન ખાલી સૂચિ હોય, તો પ્રથમ તત્વ લઈને રકમ આપવામાં આવે છે અને તે પછી તેને બીજા ભાગની ગણતરી કરવાના આડંબરી પરિણામ સાથે ઉમેરો. આ પ્રોગ્રામિંગની એક સંપૂર્ણપણે અલગ શૈલી છે જેને ડેક્લેરેટીવ પ્રોગ્રામિંગ કહેવાય છે અને તમે તેને જોઈ શકો છો. તે ખૂબ જ ભવ્ય છે અને તેની પાસે તેના ઉપયોગો અને સુવિધાઓ છે. પાયથોન ખરેખર કાર્યાત્મક પ્રોગ્રામિંગથી તેમાંની કેટલીક સરસ લાક્ષણિકતાઓ ઉછીના લીધાં છે. અને ખાસ કરીને તેમાંથી એક કે જે આપણે જોયું છે તે નકશા અને ફિલ્ટરનો ઉપયોગ છે અને સામાન્ય રીતે એવો વિચાર છે કે તમે વિધેયોને અન્ય કાર્યોમાં પસાર કરી શકો છો. તેથી, આ બધા કાર્યાત્મક પ્રોગ્રામિંગ અને નકશા અને ફિલ્ટરમાંથી આવતા હોય છે જે એક કાર્ય કરવાની પરવાનગી આપે છે અને સૂચિના દરેક ઘટક પર લાગુ થાય છે. અને ત્યારબાદ આની પરિણામે, એક વાક્યમાં સૂચિના સૂચિનો ઉપયોગ કરીને નકશા અને ફિલ્ટરનું મિશ્રણ લખવાના અત્યંત સઘન રસ્તો. આ કાર્યાત્મક પ્રોગ્રામિંગની સુવિધાઓ છે જે પાયથોનમાં સંકલિત છે.

પૂર્ણ કરવા માટે, કોઈ એવું ક્યારેય કદી ન કહી શકે કે પ્રોગ્રામિંગ ભાષા એ શ્રેષ્ઠ પ્રોગ્રામિંગ ભાષા છે, કારણ કે જો ત્યાં શ્રેષ્ઠ પ્રોગ્રામિંગ ભાષા હતી, તો દેખીતી રીતે બધા જ તે ભાષાનો ઉપયોગ કરશે. ખૂબ જ હકીકત એ છે કે ઘણી પ્રોગ્રામિંગ ભાષાઓ આસપાસ છે, તે સ્પષ્ટ છે કે કોઈ પ્રોગ્રામિંગ ભાષા સાર્વત્રિક રીતે દરેક અન્ય કરતા વધુ સારી નથી. તો શું થાય છે તે, તમે એક ભાષા પસંદ કરો જે તમારા માટે શ્રેષ્ઠ છે. ખાસ કરીને અહીં અમે પ્રોગ્રામિંગ અને પ્રોગ્રામિંગ ડેટા સ્ટ્રક્ચર્સ, એલ્ગોરિધમ્સના વિવિધ પાસાંઓ શીખવાનો પ્રયાસ કરી રહ્યા હતા. અને પાયથોન એ સરળ ભાષા છે જેની સાથે કામ કરવા માટે અને ઉપયોગ કરવા માટે અને તે સરસ અને બિલ્ટ-ઇન વસ્તુઓ જેવી કે શબ્દકોશો અને સામગ્રી જેવી, અપવાદ હેન્ડલિંગ, ખૂબ સરસ રીતે કરવામાં આવે છે, તેથી તે જાણવા માટે તે ખૂબ જ આકર્ષક બનાવે છે પરંતુ જેમ આપણે જોયું તેમ, એ જ વસ્તુઓ જે તે શીખવા માટે આકર્ષક બનાવે છે, જાહેરાતોની અભાવ પણ તેની સ્પષ્ટતાને મર્યાદિત કરે છે અમે અમલીકરણો અથવા ઓબ્જેક્ટસના સંદર્ભમાં ગોપનીયતા વિશે વાત કરી શકતા નથી. તેથી વાર્તાની નૈતિકતા એ છે કે જ્યારે તમારી પાસે પ્રોગ્રામિંગ ટાસ્ક છે, ત્યારે તમે જાણો છો કે તમે શું કરવા માગો છો, તમારા કાર્યને શ્રેષ્ઠ રીતે અનુરૂપ ભાષા માટે જુઓ. પ્રોગ્રામિંગ ભાષા શીખવા માટે ડરશો નહીં કાર્ય, કારણ કે એકવાર તમે એક પ્રોગ્રામિંગ ભાષા શીખ્યા, વાસ્તવમાં તે એક અને બીજા વચ્ચેનો એટલો તફાવત નથી, તે ફક્ત નાના તફાવતો છે. અલબત્ત, ત્યાં વિધેયાત્મક પ્રોગ્રામિંગ જેવી વિવિધ શૈલીઓ છે જે ખૂબ જુદી જુદી શૈલીઓ ધરાવે છે, પરંતુ જો તમારી પાસે થોડીક પૃષ્ઠભૂમિ હોય, તો તમે ફક્ત સંદર્ભ મેન્યુઅલને જોઈને અને તેની સાથે કામ કરીને પ્રોગ્રામ્સને એક ભાષામાંથી બીજા પર સ્વિચ કરી શકો છો. અલબત્ત, જો તમે લાંબા સમય સુધી પ્રોગ્રામિંગ ભાષા નો ઉપયોગ કરો છો, તો તમારે તેને સારી રીતે શીખવા માટે સાવચેત રહેવું જોઈએ, પરંતુ ઘણી વસ્તુઓ માટે તમે ફક્ત એક ભાષાને બીજી ભાષામાં અનુવાદ કરીને ફ્લાય પર મેળવી શકો છો તેથી, મુખ્ય સંદેશ એ છે કે તમારે પ્રોગ્રામિંગ શીખવા પર ધ્યાન આપવું જોઈએ. સારા એલ્ગોરિધમ્સ સાથે કેવી રીતે અનુવાદ કરવો તે શીખવું, સૌ પ્રથમ, એલ્ગોરિધમ્સનું કેવી રીતે અસરકારક અમલીકરણમાં ભાષાંતર કરવું, સારા ડેટા માળખાં શું છે, તેથી તમારું ધ્યાન એલ્ગોરિધમ્સ, ડેટા માળખાં, સૌથી વધુ ભવ્ય રીતે હોવું જોઈએ જેમાં તમે શબ્દસમૂહ કરી શકો છો સૂચનો અને પછી પ્રોગ્રામિંગ ભાષા વિશે ચિંતા કરો. પ્રોગ્રામિંગ ભાષા બેસવાનો અને શીખવાની ભૂલ છે; કોઈ પ્રોગ્રામિંગ ભાષા શીખે છે, તમે પ્રોગ્રામિંગ ભાષા ની લાક્ષણિકતાઓ શીખી શકો છો અને સારા પ્રોગ્રામ્સ સાથે આવતાં તરીકે વાપરવા માટે તેમને મૂકો. તેથી, આ સાથે હું તમને સર્વશ્રેષ્ઠ ઈચ્છું છું અને મને આશા છે કે તમારી પાસે પ્રોગ્રામિંગમાં આગળ એક ફળદાયી કારકિર્દી છે. આભાર.