```
Name: Krishna kiswabarna
                             you, we can use two pointer approach:
 Roll: 24CSGOR71
 Putorial 2: Algorithm and Design Analysis ? (must record) much Alberton
 Date: 05/09/2024.
                                     11 Initializing pointers
(1) Algorithm: 10 + (mpala)
   count Paths (graph, S, g) { 11 S = Source Node, g= bink Node.
       top Order : topological sort ( graph);
       de [v]= {0} Il Initialize de vection to 0 100 miss - house
       dp[9] = 1 11 Privial case of path from 'g' to 'g'.
       for each node in reverse (toponder):
           for each neighbour u of vive > must tremost fisses
            dp[v]+=dp[w]
                                             1-1-1
      return dp[s]
 Time complexity: TCA = O(V+E) + O(V+E)
                   ropological pp Calculation
                                                   working-
                      Jont
                                   FI= mus , [ F. E , ZI , DI ] = a.
.> working -
   5 -1 - 3 - 9 1 met ant the manual man (21101 + 18) 3 - 2 moll
                        816 2142 = mustherrow 8 = 1,001 : 194
After topological sort assume, 3,1,2,3,4,9
starling from g and process in neverse topological order.
 dp (47 = dp[g7=1 11 Edge from 4 → g
 dp[3] = dp[g] = 1 11 " " 3 - 9
  dp[2] = dp[4]=1 11 " " 2→4
  dp[1]= dp[3]=1 11 " 11→3.
 dpts) = dpti) + dpt2) = 1+1 +2 // There are edges from s -> 1 and s.
-: o. No. of paths from S -> g + dp[s]= 2.
```

Assume graph: o) Working. Digketra Working: Step1: A=0, B=00, C=00 Step2: A=0, B=5, C=2 slep3: B= min(5,2+(-10)) -> -8 (2+0,0) nin -8 Now, since it is a cycle we will again visit & and relax the vertex and it will go on . so, it fails for negative edge weight. .> Bellman-Ford Algorithm bellman-fond (graph, s) ?... 11 Initialize all nodes distance as so. - Time complexity: for each vertex V in graph & distance [v) + 0 predecessor [v] - null distance[s] ← O. 11 Relax edges V-1 times fox (i=1 → v-1) f for each edges (u, v) in graph f. if (destance [u] + weight [u,v) & distance[v]) { distance[v] = distance [u]+ weight(uv) predecesson [v]=u 11 check for -ne weight eyele. for each edge (u,v) in graph: if {distance[u] + weight (u, v) < distance[v]: networn "-ve weight"; neturn {distance, predecessos};

· > Working:

A=0, B=0, C=0 Elevation 1: Relax A - B?

A = min (0,5+5)=0

Relar A-C:

Relax A-C:

c= min (00, 0+2) = 2

A= min (0,2+2)=0

Relax B-C;

C= min (205+ (-10)) = -5 00 20 emole 1 20 , B= -8, C=-5

B = min (5, -5 + (-10)) = -8

Distance: A=0, B=-8, C=-5

Hence bellman fond works for ne weight edge cycle.

Pleration 2: 1931

Selax A-Biggs

B= min (0,0+5)=5 ((0)+6) da B= min(-8,0+5)=-8

c=min(-5,0+2)=-5

+ well go on bo, it foils for regardie elge weight. +2)=2 Relat B-C:

c=min(-51-8+-10)=-5

Distance: Aprop broknow

for each rester V in graph & distance [v) + 0

distance [5] + 0. 11 Relax edges 1- 1 times

}(1-1 (-1-i) xaj

Mane: Kreikua Lieundarma. (3) Algorithm: Rell: SYCSEORFI Hore, we can use two pointer approach: pairwithsum (arr, sum) { september of the matinopala : showing Time complexity: sont (aur); O(nlogn)+ o(n) 11 Initialiting pointers

i=0, j= kingen(arr)-1;

Sonting 2-pointer approach. 11 Initialiting pointers while (ixj) { current-sum = and arr(i) + arr(j); (1) (nlogn) if (avoient-sum == sum) as a seas lawrence neturn {arrti], arrtj]}; else if (coverent_sum < sum): 1=1+1 Recorded Applied else j=j-1 return { } 11 No pair found . v) 0 + (3+v) 0 = 151 Tepological DP Columbianion · > workingause= {10,15,3,7}, sum=17. 11 Sout: {3,7,10,15}. wrient_sum = 3+15 → 18 Step1: 1=0, 1=3 aurrent_ sum = 3+10 >> 13 btep 2: i=0, j=2 step 3: i=1, j=2 current_sum = 7+10)(i7) -> Retwen {7,10}. beh wast styll " 11 1= [8] 30 = [239

P(1) de[3] = 1 11

```
(4) Buick sort Algorithm?
   quicksont (aro, low, high):
         if (low(high):
             pi = partition (arr, low, high)
             quick sort (arr, low, pi-1);
            quiek bont (arr, pi+1, high);
  partition (arr, low, high) {.
        pivot = arr[high]
        1 = Low -1
        for (j=low to high-1){
             if arrest pivol s
                  i = i+1;
                  swap (arr(i), arr(j));
      swap (arr(i+i), arr(high))
     networn i+1;
 of proving that Quick sort sorts the element conrectly is:
 We will use Mathematical Induction on the size of the array:
 Assume Qs (Quicksont) always work for all averag of size < n i.e.
  IEKEN.
is Partioning correctly accordinges the element such that all
 elements less than pinot is on left and others are on right and
 pinot is in its perfect place.
ii) New serice the size of averay is less than 'n' for the both ends.
 Now, by our inductive hypothesis these are sorted correctly.
" Quicksorts sorts the array correctly.
TC: O(nlogn)
```