Name: Sindhuja Yerramalla
UUID: U00839259.
email: Syrrmila@memphis.edu

Home Work-4, Algorithms and Problem Solving

Note: I have referred textbook, lecture notes and so many random websites like chegg & stack overflow and also worked with few of my classmates.

```
Contain & spin stands of the 3. I = [1][1] of a country ha stand soon
 Problem 1: Problem 6.10 from DPV
                                 ne shoogs (c.) Crigo s recress six
```

Let P[i][i] be the probability of obtaining i heads in i coins Basecase for recursion formulation is P[0][0]=1 (probability of getting o heads in ocoins)

Recursion can be given as P[i][i]=P[i-i][i](1-P[i])+P[i-i][i-i]Hi] where probabilities of n coins be in P[1,2,---- m] This represents the probabilities of both getting head in ith coin and reducing the j and not obtaining head & not reducing the j

Algorithm: 1 to D- M+12 Call Bets deliberty of 1 1 2 20000216 21 let P be a n x (x+1) array of zeros Englinges to "a", Live combeased mass pe out 3 cold

for i from 1 to n:

P[i][0] = P[i-i][0](1-P[i])

for ; from 1 to man(i, k):

P[1][8] = P[i-1][8] (1-P[i]) + P[i-1][3-1] P[i]

return P[n](x)

And the time complexity for the above algorithm is o(n2) 1= 03[1]96

Problem 2: Problem 6.6 from DPV

This can be solved by DP. Let dP[i][i] be true of it is possible to parathesis the substrang from index ; to index ; (in clusive) in such a way that the value of the resulting 1= [7][7] 96 expression is "a" D-MITOJO A MANDER

Here of is the multiplication table given in the guestion of

function can parenthe sizes?

for 1 from 0 to not 1 reg

: mrttrioo/A

Base Case by recurssion; dp[i][i] = 1 of this a at i else o Recursion: dp[i][i] depends on the values of dp[i][i-1], dp[i+][i] 2 most in 2 board and dp [+1] [-6] [1-6] bons boards in i come

The values of dp for substrengs of length 2 and higher are then filled in by the function using a nested loop. It considers all possible ways to devide each substring of length K anto two non-empty substrings for each substring in order to find a way to parenthesize the two substrings so that their product evaluates to "a", and then et uses the multiplication table to combine the two products to create a product that evaluates to "a" for the entire substring. If a solution is discovered, the function sets dp[i][i+k-1] to 1, indicating that the substrang can be parenthesized an order for it to evaluate to "a". Time complexity will be o(m3)

Alaprethm:

6 6.000 1 p (E.Ja-1)[0][1-1]3 = [0][1]3 function can-parenthesize(s)

n = length(s)

dp = array of size nxn, instialized to o

for 1 from 0 to not do:

of scil== "a" then on one of printing soit soit soit

1= [1][1]9b

for K from a to n do a leading more and mardens a complete

for 9 from 0 to n-k do

3=1+K-1

The source solves by op need apply for P from 1 to 3-1 do if dpliJCP] == | and dplp+D[8] == 1 and M(S[P], S[8]) == 20-then (include up) in such a

1=[8767395

return aprojen-i]

Here M is the muti plication table given in the question of

for i from I to me

[A][U][WXM] BA

D 5. NOUSSOIDE

Problem 3: problem 6-12 from DPV-90 months of molders

Let us assume a DP array dp[n][n] where n is the number of Vertices on the polygon.

We assume the subproblem as A(1,8) where ?,8 represent the vertex numbers in the given polygon and finds out the cost of triangulation in the sub polygon formed by those vertices.

Base Case : dp [1] [1+1] =0 for all 1

Recursion Case: dp[i][i] = men(dp[i][i], destance(ffi], P[i])+ dp[i](x)+ 96[k][9]]

where distance gives the distance between the vertices and kranges from it to 3-1. so this gives the distance based on the cost of the previously calculated vertices and the newly calculated ost

Algorithm:

n = length (P)

dp = array of size nxn, enitalized to infinity

Base Case: for & from 1 to n-2 do in of 1 2

dp[ma+1]=0 [-Mina = Grima

Algor:thm

[EB]CB]x-F]A TO EDICTIA = CODETIA for 1 from 1 to n-2:00

dp[1][1+1]=0

afor ten: erominas to, no son generation to sou as a soud rely 4

not forms from 1 to 10 -len +1;

through the me take other possibilities legislating for k from P+1 to f-1! cost = distance (P[i], P[i]) + dp[i][k] + dp[x][j]

deliges (forade) usu = [escides]

return dp (UC1)

for you to no

ALOJA = CING

1[6] x = < ; }9

[n][v] A muter

```
Problem 4: Problem 6.17 from DPV.70 mon and maldare : & maldare
 Let us assume a DP array A[v][n] where v is the change requested
 and n is the demomination of the coins i.e x11x2, -- xn
Base Case: A [O][i] = true because we can make o change with
                 any coins given
    pounded by these westers.
 Recursion s
  (8x=2: 2) [8][84-1]A to [1-8][1)A = [8][1]A
  ACIJCIJ = ACIJCIJA (if 12 xi) (if 13 CIJCIJA = CIJCIJA
distance gives the distance between the vertices and sough
 Function make-change (xD, -- n], v);
    of the previously calculated vertices [+m][iti] replood over = A
    for 920. to n:
                                                   ALGORITHM.
      A.COJCi) = true.
                                                1) = length (P)
     for 1=1 to V: Potinger of Lagellating from ago go para ago
       for 3=1 to n: 00 8-10 01 1 0019 7 109 : 3200 3208
         [-1][1]A = [8][1]A
                            00 [1+1][7] 9b
         of 1>=x[8]:
          [[3][3] x - 1] A TO [[3][1]A = [3][1]A
    return A[v][n]
                                            0=[1+1761/96
```

After base case, if we can generate the change with i roins then we can do the same change with its coins [induction theory]. Else we take other possibilities by considering change for 9-x[i] coins

Time complexity is o(n2)