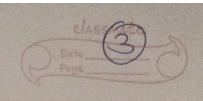
Chetna Ajay Nounani classmate 1 Rachana Vinal Grandhi Harsh Pradeep Nahar Homework-2 Coins of unlimited Supply (n, n2 - n) 6. P) OD Green: Input n., n, --- no Value to be totalled = V 6 10 100 psous of Is it possible to make changes for vusing coins as denominations n_1, n_2, n_3, n_4 Algorithm: > (indexing from 1 not 0) coinChange (coin, v) /1 method coin change Meclaring dynamic programming overally with length greater than the value DPEV+1] DP[1]=0 //for first value v we need 0 coins DPC2--00) = 05 // Setting all the rest elements to max value length = length (coin) for i = 2 to v+1 for j=1 to length

if (coincid = i) rest = DP[i-coin[i]) if (rest! = 00 AND rest+1 < DP[:]) DPCi] = rest +1 it DP[n] = 00 else setum DPCVJ

C-4,00001114 Explanation:> The final value will be contained in De the pointry principle is to verify for every coin and value.

De seconds the amount of coins needed to generate that specific value for I, say a certain is larger than a specific coin therefore we may create variations for that value i, because we wish to make I the ultimate solution will be at i I wat coin Ev I. Run-time analysis: V // Setting all elements to as V/1 for the first for loop length 11 for Second for loop 11 Total 4 lines inside the loop o Overall, Runtime = VX4XVX length -15+4 NV.



6.1902) KCoins () total [0]=0

for i=1... to value

T[i]-00

for Z=1...6(coin)

Check if (coin[Z] \(\) & total[i]

> 1+ total[i]-coin[Z]

total[i] = 1+ total[i-coin[Z])

Check if total (value) >t return false else

return true

Euplamation: >

> Here we are checking if ## of limited Coins used to make change will be be c or not:

Tit is possible to make using their their of the life to the the return their

Felse we ketwan false
Total keeps the track of min no.
of coins used to make value

Running time > 0 (nkv)

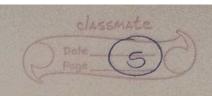
O(nv) proved in last problem, here we Consider k iterations inside.

We will create a 2 dimensional table

T, also, T[i,i) will store the

longest palindrome in the string
with, wij If symbols wi and wij are

Same, we can assume that they he longest palinblome TCi)j] = 2+TCi+1, j-1]. If the two symbols are different, then both will not be a point of any palindromic Sequence Hence, Thij] will be max & Thitij] and Thijj-1]. The best Case happens when i =



Algorithm:

for i=1 to n // Declare all diagonal

T[i,i]=1; Cells of matrix as I

for i=n-1 down to 1 // for each cell

for j=i+1 to n above diagonal

if (wi = wj)

T[iji]=2+T[i+1,j-1]

lse

T[i,i] = max (T[i+i]], T[i,i]T

Algorithm analysis

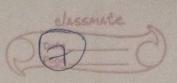
The first for loop Costs on time.

It takes on time to traveur from n-1 down to I in the outer for loop and then an additional on time to traverse from i+1 to n, t will be Constant time to check if else Conditions.

Thus time Complexity will be n(n+k)+n $-n^2+nk+n$

 $\frac{1}{1000} = \frac{n^2}{1000}$

6.13) (04) a) The following is the Sequence of rards: 7,400,4,4 When the first player will be greedy, he will choose the first card (value) Then, the second player will choose the card with value 400. The optimal Strategy for the first player is to choose the last Card with value 4 now, the Second player can choose either the value? or the remaining card with value Now the first player will choose the could with value 400. In this way, the first player can win. The time Complexity is $O(n^2)$ as the table Size is $n \times n$ which is precomputed. Also, we can consider the following sequence {1,2,10,33



first player: 3 second player: 10
flere, the first, player loses biggest raxo first player: 2 Decond player: 1 b) Algorithm:> Optimal Strategy (S) 0 < [n...0] [n...0] > 0 for i in range (0, n):

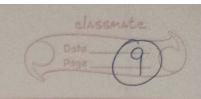
for j in rounge (0, n): d[i][i]=0 Pice

> d [] [] = max (S[i] + min (d[i-1,j-1] - S[i]) , d [i-2, i] - S[i] 8 [j] + min (d[i-1,j-1]-

S[i], d[i,i-2]-S[i-0]

return d[n][n]

Analysis: > Lot's take the assumption that player 1 attempts to marinize core and player 2 attempts to minimize it : Hence player 1 will Choose the first card or the last card and it same for players. Should opt for main the Strategy in order to manimize the We observe that d [ij] rell Values of d[n,y] where i \le n This algorithm computes: in order of increasing in take O(n²) time because)(n²) values to compute, each shich takes Constant time. U one a matrix cliji] which is Value that player Should choise. Using this choice matrix, player I can play optimally.



6.11 05) Longest Common Subsequence (String), String2)

length 1 = len (String)

length 2 = len (String)

dynamic Prog [1 -- length 1] [1--- length 2] >0

for index! in range (1, length +1)

for index2 in range (1, length 2+1)

check if String! [index!-1] == String2[index-1]

dynamic Prog [index] [index 2) =

dynamic Prog [index-1] [index 2-1]+1

else dynamic Prog Eine EXIJ [index 2] = Max (dynamic Prog Eindex 1], Cindex 2-1], dynamic Prog Cindex 1-1] [index 2])

Jeturn dynamic Prog [length 1] [length 2]

Emplanation:

Firstializing the dynamichog matrix with D Fif the tent at Lindex-II and Lindex2-II matches then we Store dynamichoglidal

then we sto else me take maximum of cinden [] [index 2-1] & Cindex [-1] [index 2]

Running Time = O(mn)