Date / /

NAME :- KRUNAL RANK.

ROLL NO :- U18CO081

CLASS :- B TECH 3RD YEAR,
COMPUTER ENGINEERING.

U18CO081

5/11/20

Camlin Page

Date / /

CAA
Tutorial 8

Ans 1: Given, dimensions sequence of matrices =
{5, 10, 3, 12, 5, 50, 6}

The required dp table can be formed as:-
dp[i][j] where matrix ~~is~~ i to j are already multiplied.

	1	2	3	4	5	6
1	0	150	330	405	1655	2010
2	–	0	360	330	2430	2070
3	–	–	0	180	930	1770
4	–	–	–	0	3000	1860
5	–	–	–	–	0	1500
6	–	–	–	–	–	0

Hence, the required parenthesization is,

~~(5, 50, 6)~~

~~((((5, 10, 3) 12, 5)) (5, 50, 6))~~

((5×10)(10×3)) (((3×12)(12×5)) ((5×50)(50×6)))

Ans 2: The vertices of the subproblem are ordered pair $V_{i,j}$ where $i \le j$.

- If $i = j$, then vertex V_{ij} has no output edge.
- If $i < j$, for each k, s.t. $i \le k < j$, the subproblem graph contains edges (V_{ij}, V_{ik}) and $(V_{ij}, V_{k+1 j})$ and these edges indicate that to solve the subproblem of optimally parenthesizing the product $A_i, \ldots, A_j$, we need to solve subproblems of optimally parenthesizing the product $A_i, \ldots, A_k$ and $A_{k+1}, \ldots, A_j$

Scanned with CamScanner

The number of vertices is:-

$$\sum_{i=1}^{n}\sum_{j=1}^{n} j = \frac{n(n+1)}{2}$$

The number of edges is:-

$$\sum_{i=1}^{n}\sum_{j=1}^{n} (j-1) = \frac{(n-1)(n)(n+1)}{6}$$

Ans 3:- Given $P = \{1,0,0,1,0,1,0,1\}$

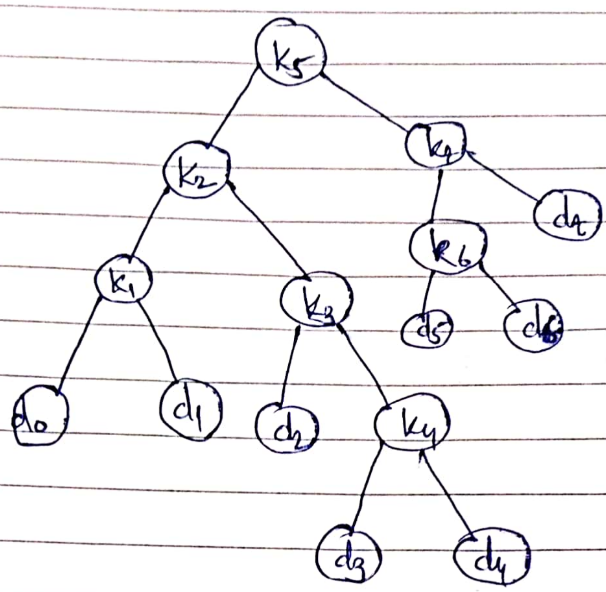
$Q = \{0,1,0,1,1,0,1,1,0\}$

dp	1	2	3	4	5	6	7	8
1	0	1	1	1	1	1	1	1
2	0	1	1	2	2	2	2	2
3	0	1	1	2	3	3	3	3
4	0	1	1	2	3	4	4	4
5	0	1	1	2	3	4	4	5
6	0	2	2	3	3	4	5	5
7	1	2	3	4	4	4	5	6
8	1	2	3	4	4	5	5	6
9	1	2	3	4	5	5	6	6

Hence, the LCS is:-

0101101

Ans 4:



Here, the cost is 3.12.

Ans 5:

Greedy Algorithm	Dynamic Algorithm
• The best option available is chosen in the hope that it will lead to optimal solution.	• At each step, we make a decision based on previously solved subproblems to reach the optimal solution.
• No guarantee of reaching the optimal solution.	• Always gives optimal solution.
• It is more memory efficient.	• It is less memory efficient.

For example, Dijkstra's shortest path solution is a greedy approach to ~~reach~~ find minimum distance between two ~~nodes~~ nodes in a graph that doesn't have negative edges.
We make a choice which ~~is~~ has the minimum cost and stop when we reach the final node.

On the other hand, Least Common Subsequence is a DP problem with

$$LCS[i][j] = \begin{cases} LCS[i-1][j-1]+1 & ; \text{if } P[i] = Q[j] \\ \max(LCS[i-1][j], LCS[i][j-1]) & ; P[i] \,!= Q[j] \end{cases}$$

Scanned with CamScanner