Analysis and Design of Algorithms Rubrics
Quiz-2
Set-3

Date: 23.02.2023

—---

Question 1)

You are given an n x n grid, where cell (i,j) is the cell of the i-th row and the j-th column, with (1,1) representing the top-left corner, (n,n) denoting the bottom right corner of the grid. Each cell (i,j) has an integer value a_{ij}. You want to map out a path from top-left to bottom right such that the values of the cells you traverse have the maximum sum possible. However, you can only move down or right one cell at every step. Which of the following recurrences represents a feasible way of doing the same?

(Where dp[i, j] is a certain dynamic programming function for the same)

a) dp[i, j] $= a_{11}$ if i = j = 1
$= dp[i-1,j] + a_{ij}$ if i > 1, j = 1
$= dp[i, j-1] + a_{ij}$ if i = 1, j > 1
$= \max\{dp[i-1,j], dp[i, j-1]\} + a_{ij}$ otherwise
b) dp[i, j] $= a_{nn}$ if i = j = n
$= dp[i+1,j] + a_{ij}$ if i < n, j = n
$= dp[i, j+1] + a_{ij}$ if i = n, j < n
$= \max\{dp[i+1,j], dp[i, j+1]\} + a_{ij}$ otherwise
c) Both are feasible recurrences
d) Neither is a feasible recurrence

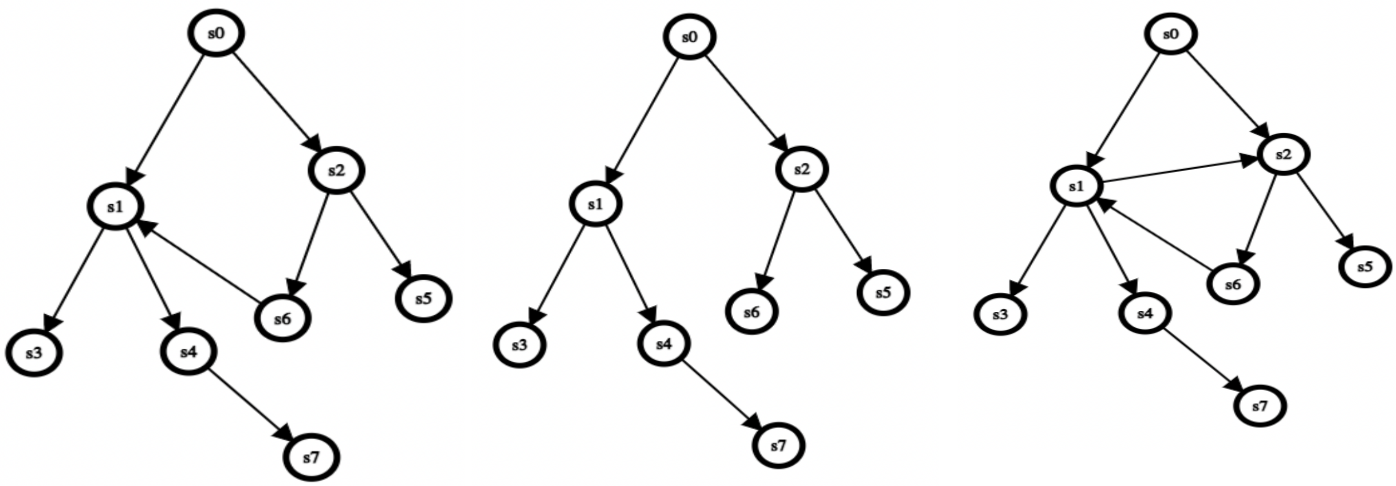
Question 2)

Given an array of *n* weights $W = [w_1, w_2, \ldots, w_n]$, you need to select a subset W' of weights from this such that the sum of weights in W' is maximum, but no two adjacent from W are in W' (that is, both w_i and w_{i+1} are not in W' for any i ≥ 1). An intuitive approach is to check each weight w from W, starting from the heaviest, and checking them in non-increasing order of weights. If the adjacent weights of w are not in W', w is added to W', else it is not. Is this approach correct? If not, select an input from which this will fail.

a) W = [1, 2, 3, 4, 5]
b) W = [2, 5, 6, 4, 3, 5]
c) W = [1, 5, 2, 4, 3, 6]
d) The proposed algorithm is correct and works correctly for all inputs

Question 3)

Recurrence 1 | Recurrence 2 | Recurrence 3



Given 3 Recurrences, to evaluate the s0 for each of the cases **optimally** which of the following is correct

a) 1->Recursion, 2->Dynamic Programming, 3->Recursion
b) 1->Can’t be Evaluated using Rec or DP, 2->DP, 3->DP
c) 1->DP, 2->Recursion, 3->Recursion
d) 1->DP, 2->Recursion, 3->Can’t be Evaluated using Rec or DP

Question 4)

Consider a country whose coins are minted with denominations of {d1, . . . , dk } units. We want to count how many distinct ways C(n) there are to make change of n units. For example, in a country whose denominations are {1, 6, 10}, C(5) = 1, C(6) to C(9) = 2, C(10) = 3, and C(12) = 4. Now, How many ways are there to change 20 units from {1, 6, 10}?

A. 20 B. 1 C. 7 D. 11 E. 10

Question 5)

Given an integer valued array of size N, you were asked to find if there exists a subsequence having a sum S. You were aware of two algorithms,
1) Write down a DP solution, DP(n,s), which signifies that by using the first n elements of the array, can you generate a sum s.
2) generating all subsequences of the array using brute recursion, and individually checking if any of them sum to S.

Which of the following is/are true regarding the two algorithms you were aware of

a) Algorithm 1 is a more better version of Algorithm 2, and in no case would one prefer to use Algorithm 2 over algorithm 1
b) For a setup with N being constrained to a small value, Algorithm 2 may perform better than Algorithm 1
c) For a setup with S being constrained to a small value, there can arise a situation where Algorithm 2 performs better than Algorithm 1
d) For a setup with S being constrained to a small value, Algorithm 1 always performs better than Algorithm 2

Question 6)
Which of the following is the recurrence relation to get the minimum number of scalar multiplications to multiply the matrices give, where M[i-1] * M[i] gives the dimension of the ith matrix?

a) dp[i,j] = 1 if i=j
dp[i,j] = min{dp[i,k] + dp[k+1,j]}

b) dp[i,j] = 0 if i=j
dp[i,j] = min{dp[i,k] + dp[k+1,j]} for all k in i to j

c) dp[i,j] = 1 if i=j
dp[i,j] = min{dp[i,k] + dp[k+1,j]} + M[i-1]*M[k]*M[j].

d) dp[i,j] = 0 if i=j
dp[i,j] = min{dp[i,k] + dp[k+1,j]} + M[i-1]*M[k]*M[j].

Question 7)

The King of St.Petersburg has come across a problem that he is confused about and now requires your help. St.Petersburg has a treasure that consists of N coins. The king now wants to find the largest subset of coins, say S, such that if coins A, B ϵ S then either A divides B or B divides A. You being his smartest ADA student are now required to find the right approaches to solve this task.

A. We can apply DP to solve the problem in O(N^4) by running 2 loops that denote the first and last indices respectively of the subset in the array of coins and then run 2 more loops that just checks whether each coin in this subset divides every other coin.
B. We can apply DP in O(N) by finding the longest subsequence S such that S[i] divides S[i+1].
C. We can apply DP to solve the problem in O(N^2) by sorting the numbers and then we need to find the longest subsequence S such that S[i] divides S[i+1].
D. DP is possible by requiring O(N * (2 ^ X)) time where X is the total number of subsets possible.
E. We cannot apply DP to this problem.

Question 8)

Returning ADA legend, Mr Kong the monkey is looking at a tree (tree here refers to trees in CS and not the Plant Kingdom). Each vertex of the tree contains some amount of bananas. Mr Kong being lazy wants you to find a single simple path in the tree which contains the most amount of bananas. A simple path in a tree is a path from vertex u to vertex v, such that no vertex in the path from u to v is repeated. Which of the following is/are true.

a) The vertex with the largest amount of bananas will always be in the optimal simple path
b) This can be solved using the following strategy: start from the vertex containing the maximum number of bananas, next go to the children of that vertex which has the maximum number of bananas, continue till you reach a leaf node.
c) If a dynamic programming approach was to be taken, defined as: opt[v] = bananas[v] + max_over_all(opt[child[v]]), where opt[v] represents the optimal path including vertex v, is a correct recurrence, final answer will be in max(opt[v]) for all v in the tree.
d) Keeping track of the optimal at a vertex (opt[v]) is not enough in the recurrence given in option c and more information is needed.

Question 9)

Given an optimal substructure for a Dynamic Programming problem as OPT[a][b][c] = F(a,b,c) where F is some recurrence function in a,b,c. Further it is given to you that OPT[a][b][c] can always be evaluated. Which of the following is/are true about this Dynamic Programming problem.

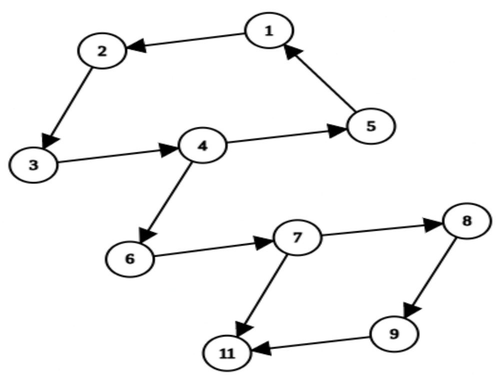
a) Evaluating OPT[a][b][c] will take O(abc) time to be evaluated
b) Evaluating OPT[a][b][c] may or may not take O(abc) time to be evaluated

c) Given OPT[a][b][c] takes O(abc) time to be evaluated. You were somehow able to prove that c is a linear combination of a and b, then OPT[a][b][c] will take O(ab) to be evaluated.

d) Given OPT[a][b][c] takes O(abc) time to be evaluated. You were somehow able to prove that c is a linear combination of a and b, then OPT[a][b][c] may or may not take O(ab) to be evaluated.

Question 10)

Consider the following graph where each node represents a DP state. An edge from node X to node Y (written as (X, Y)) denotes the recurrence relation that node Y depends upon node X's result for computation.



Changing direction of which edges would allow node 4 to use node 1's computation ?

A. (3, 4)
B. (2, 3)
C. (4, 5)
D. (1, 2)
E.(6, 7)

—--END—--

ROUGH