Registration Number :

VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

Continuous Assessment Test II - October 2024

Programme	B.Tech.(CSE)	Semester	Fall 2024-25
Course	Design and Analysis of Algorithms	Code	BCSE 204L
Faculty	Dr.L.Jeganathan, Dr M Janaki Meena, Dr M Raja, Dr R Sivakami, Dr B Indra , Dr G Kavipriya, Dr N.Jeiprathap	Slot/Class No.	A1/CH2024250101354 /CH2024250101360 /CH2024250102306 /CH2024250100952 /CH2024250100957 /CH2024250100961 /CH2024250100543
Time	90 Minutes	Max. Marks	50

Instructions:

- Answer all the FIVE questions.
- If any assumptions are required, assume the same and mention those assumptions in the answer script.
- Use of intelligence is highly appreciated.
- In this open-book exam, you are expected have a maximum-thinking and answer the question with just the necessary required information.
- You are requested to answer the five questions in the first five pages of your answer book, with one question each in a page.
- You are requested to do all the rough works from the page six. Contents from page six onwards, will not be evaluated.

---

1. You are given two sequences which $X =< A_1A_2, ...A_m >$, $Y =< B_1B_2, ...B_n >$ where A_i's and B_j's are matrices, $0 < i \leq m$, $0 < j \leq n$. The order of the matrix A_i, is represented as a 2-tuple (rA_i, cA_i), where rA_i is the number of rows in A_i and cA_i represents the number of columns of A_i. Similarly, the order of the matrices B_i, $i = 0$ to n is represented as a 2-tuple (rB_i, cB_i). The task is to compute the Longest-Common-Subsequence of X and Y, in the sense of the order of the matrices, denoted by $LCS(X,Y)$. For example, if $X =< A_1A_2A_3A_4, A_5 >$, with sizes $(5,10), (7,3), (3,12), (12,5), (5,7)$ respectively, $Y =< B_1B_2B_3B_4 >$ with sizes $(7,3), (3,12), (5,7), (4,7)$, LCS(X,Y) is $A_2A_3A_5$. Here $A_2 \cong B_1$, $A_3 \cong B_2$, $A_5 \cong B_3$. Here $A_i \cong B_j$, for any i, j, means that the matrix A_i and B_j are of the same order. We can also write $LCS(X,Y)$ as $B_1B_2B_3$. For the purpose of this problem, $LCS(X,Y)$ should be written in terms of A's alone. You are required to use the dynamic programming strategy for this problem. The sequence $< X, Y >$ is represented as $[m, n]$, where the sequence X has m matrices and Y has n matrices. Length of the $LCS(X,Y)$ is represeneted as $L[m,n]$,

   (a) Write the recurrence relation used to compute the $L[m,n]$ [2 marks]

   (b) What will be the entry in the $[3,2]^{th}$ cell of the bottom-up memoization table used for the dynamic programming strategy in the computation of $L[m,n]$, if the order of matrices of X are : $(2,3), (7,9), (9,6), (6,8), (8,18)$ and the order of the matrices of Y are: $(9,6), (6,8)(7,8), (8,18)$. [2 marks]

   (c) Mention the direction (left or top or diagonal) in the $(2,3)^{th}$ cell for the input given in question 1(b). [2 marks]

```
Algorithm 1 LCS(X,Y)
m=X.length
n=Y.length
Let b[0,...m:0,...,n] and c[0,...m:0,...,n] be two new tables, intialised with zero and 'space respectively
.........
..........
for i = 0 → m do
  for j = 0 → n do
    if ____(1)____ then
      c[i,j] = ____(2)____
      b[i,j] = 'diagonal'
    else if ____(3)____ then
      c[i,j] = ____(4)____
      b[i,j] = 'top'
    else
      c[i,j] = ____(5)____
      b[i,j] = 'left'
    end if
  end for
end for
```

d) There are four 'fill-in the blanks' in the Algorithm 1, which solves the given problem. Please write the response for 1.d.1 (1 mark), 1.d.2 (1 mark), 1.d.3 (1 mark), 1.d.4 (0.5 mark), 1.d.5 (0.5 mark) respectively, in separate lines. [4 marks]

2. Consider the 'Maximum Event Participation ' (MEP problem) described as follows : You are organizing a series of community events : $E_1, E_2, ..., E_n$, n is a positive integer. Each event runs for specified duration. Let the $d_1, d_2, ..., d_n$ be the duration (in minutes) of $E_1, E_2, ..., E_n$ respectively. A Participant can only attend events in the order they are listed and a participant can attend the events for a maximum duration D minutes. Given, $E_1, E_2, ..., E_n$, $d_1, d_2, ..., d_n$ and D, Task is to determine the maximum number of events that can be attended by a participant without exceeding D. Algorithm 2 is designed to solve the MEP problem.

```
Algorithm 2 MEP(E,d,D)
E = [E_1, E_2, ..., E_n]
d = [d_1, d_2, ..., d_n]
SE = {} // {Set of Selected events}
CT = 0 // {Total duration of the selected events}
for ____(1)____ do
  ____(2)____
  ____(3)____
  ____(4)____
end for
return SE
```

a) There are four 'fill-in the blanks' in the Algorithm 2. Please write the response for 2.a.1, 2.a.2, 2.a.3, 2.a.4 respectively, in separate lines. [4 marks]

(b) What will be returned by the MEP pseudocde if d=[5,7,4,3,6,1] and D=17. [2 marks]

(c) Given D=15, compute n and $d = (d_1, d_2, ..., d_n)$ such that the following conditions are satisfied. [2 Marks]

- n is maximum (in the sense that d cannot be of size greater than n that satisfies the other required conditions)
- d is in decreasing order of d_i's and all d_i's are distinct integers.

(d) Describe the time-complexity of the above MEP pseudocode. [2 marks]

3. Given a text T and a pattern P, Compute the length of the Longest Common Subsequence of T and P such that LCS(T, P) occurs in T from the leftmost character to the rightmost and the LCS(T,P) Occurs in P from the rightmost to the leftmost. You are provided with two algorithms : Algorithm 3, Algorithm 4 to compute LCS(T,P).

Algorithm 3 $LCS(T, P)$

```
m = length(T)
n = length(P)
PR=_____(1)_____
Let L[0,...m:0,...,n] initialized to 0
for i= 1 to m do
  for j=1 to n do
    if _____(2)_____ then
      _____(3)_____
    else
      _____(4)_____
    end if
  end for
end for
return L[m][n]
```

There are four 'fill-in the blanks' in the Algorithm 3. Please write the response for 3.a.1, 3.a.2, 3.a.3, 3.a.4 respectively, in separate lines.

[4 marks]

b) **Algorithm 4** BruteForce-LCS

```
Input: Text T, Pattern P
Output: Length of Longest Common Subsequence
m ← length(T)
n ← length(P)
RP ← _____(1)_____
max_LCS_length ← 0
AST ← GenerateSubsequences(T)
ASP ← _____(2)_____
for _____(3)_____ do
  for _____(4)_____ do
    if A = B then
      _____(5)_____
    end if
  end for
end for
return max_LCS_length
```

Assusme that the Algorithm 4, is supported by a function GenerateSubsequences(). If you want any other functions, you can assume the same. There are six 'fill-in the blanks' in the Algorithm 4. Please write the response for 3.b.1(1 mark), 3.b.2(1 mark) 3.b.3 (1 mark), 3.b.4 (1 mark), 3.b.5(2 marks) respectively in separate lines.

[6 marks]

4. Given $S = \{a_1, a_2, ..., a_n\}$ where a_i's are activities with the respective start-time and finish-time as (s_i, f_i), and a pattern $P = \{(s_\alpha, f_\alpha)\}$, Task is to compute the maximum subset S' of S which has mutually compatible activities such that S' has at the least one activity which has the time interval described in $P = (s_\alpha, f_\alpha)$ within itself. For example, the activity $(2, 7)$ has the time interval $(3, 5)$, within itself. If $S = \{(2,7), (3,5), (6,8), (8,12), (7,17)\}$, $P = \{(3,5)\}$, then $S' = \{(3,5), (6,8), (8,12)\}$. One

approach to solve this problem is as follows: From S and P, we first compute a set F which will have the activities that has the time interval of P, with in itself. If F is non-empty, that means that S has actvities which have the time interval of P with in itself. In the above example, F=$\{(3,5),(2,7)\}$. In that case, we can find the maximum compatible subset of S, which will ensure that S' has atleast one activity which will have the time interval of P with in itself. Based on this method, Algorithm 5 is designed to compute S'.

Algorithm 5 Maximum Compatible Subset with Overlapping Activity

a)

```
Input: Set of activities S = {(s_i, f_i)}, Pattern P = (s_α, f_α)
Output: Maximum subset S' of mutually compatible activities that overlap with P
F ← {}
counter ← 0
for each activity (s_i, f_i) in S do
    if ______(1)and______ then
        F ← ______(2)______
        counter ← counter + 1
    end if
end for
if counter > 0 then
    F______(3)______
else
    return F
end if
if F is not empty then
    Sort the activities in F by finish time
    S' ← {}
    A ← 0
    for ______(4)______ do
        if s_i ≥ A then
            S' ← ______(5)______
            ______(6)______
        end if
    end for
end if
return S'
```

There are six 'fill-in the blanks' in the Algorithm 5 . Please write the response for 4.a.1, 4.a.2, 4.a.3, 4.a.4, 4.a.5, 4.5.6 respectively in separate lines. [6 marks]

(b) Compute the time-complexity of Algorithm 5. [2 Marks]

(c) For the input $S = \{(2,7),(3,5),(8,12),(1,4),(7,17)\}$ and the pattern $P = \{(3,5)\}$, compute the output returned by the Algorithm 5. [2 marks]

5. Understand the Algorithm 6 for the N-queen's problem and answer the following.

(a) In the Algorithm 6, first queen is placed in which row? [2 marks]

(b) If the board is of size $n \times m$, n >m, how many queens can be placed in the board satisfying all the constraints. [2 marks]

(c) How many diagonals are checked before placing a queen? [2 marks]

(d) If n=4, how many recursion calls are made? [2 marks]

(e) Compute the time-complexity of the recursive function alone in Algorithm 6. [2 marks]

Algorithm 6 N-Queens Problem: Finding All Solutions

```
Input: Integer n (size of the board, number of queens)
Output: All possible solutions where queens are non-attacking
board ← empty board of size n × n
solutions ← [ ]
PlaceQueen(board, n, n − 1)
return solutions
Procedure PlaceQueen(board, n, row)
if row < 0 then
    Add board to solutions
    return
end if
for col = 0 to n − 1 do
    if IsSafe(board, row, col) then
        Place queen at (row, col)
        PlaceQueen(board, n, row − 1)
        Remove queen from (row, col)
    end if
end for
return
Function IsSafe(board, row, col)
for i = row + 1 to n do
    j = col − 1
    if j < 0 then
        Continue
    end if
    if board[i][j]=true then
        return false
    end if
    j = j − 1
end for
for i = row − 1 to 1 step −1 do
    j = col − 1
    if j < 0 then
        Continue
    end if
    if board[i][j]=true then
        return false
    end if
    j = j − 1
end for
return true
```