Vansh Sakora
23107007 4

Page No.
Date

DAA - LAB 6

TASK - 1

★ ALGORITHM : findLCS (string S1, string S2) :

// Computes the longest common subsequence between two strings
// Input : 2 strings S1 & S2
// Output : LCS of both strings

Let m = S1.length
n = S2.length
Create table with m+1 rows and n+1 cols. of empty strings

```
for i = 1 to m :
    for j = 1 to n :
        if S1 [i-1] == S2[j-1]:
            table [i][j] = table (i-1] [j-1] + 1
        else :
            table [i] [j] = larger of table [i-1] [j]
                            and table [i] [j-1]
return table [m][n]
```

Time Complexity of algorithm

i) Let "c" be the constant time taken for the innermost operation in the loops
ii) The first loop runs from 1 to m
iii) The second loop runs from 1 to n

$$T.C = \sum_{i=1}^{m} \sum_{j=1}^{n} c$$

$$= (m)(n) * c$$

$\therefore$ T.C = $O(m * n)$

∴ The T.C. is $O(mn)$ where m and n are the lengths of strings s1 & s2 respectively.

★ Algorithm: Find LCS _ Multiple (string s):

```
   // Computes the LCS among n strings
   // Input: Array of n strings
   // Output: LCS of n strings

   SET max_seq = " "
   for i = 0 to n-1:
       LCS = s[i]
       for j = 0 to n-1
           if i ≠ j
              LCS = find LCS (LCS, s[j])
              if LCS == " ":
                  break
           max_seq = max (max-seq, LCS, key = length)

   return max_seq
```

TIME COMPLEXITY:

i) There are 2 loops, one from 0 to n-1, and the other from 0 to n-1 as well.

ii) Innermost operations of computing LCS is $O(n^2)$ for the worst case possible.

iii) With an avg. string length of L, total complexity:

$$TC = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} cL^2 = (n)(n)cL^2$$

$\therefore$ T.C. = $O(n^2L^2)$

Date

★ ALGORTHM MATRIXChainMul (N, arr)

// input: An array containing 'N' matrix dimensions
// output: Smallest number of multiplications & optimal order

Set dp[1....N][1.....N] as a 2D array to store
min multiplication costs.
Set S[1.....N][1.....N] as the SPLIT points

```
for L=2 to N-1 :                          // Chain segment's length
    for i=1 to N-L :                      // Start ind.
        j = i+L-1                         // End ind
        dp[i][j] = ∞
        for k=i to j-1 :                  // Possible split points
            q = dp[i][k] + dp[k+1][j] + ~~arr[i-1]~~
                + arr[i-1] * arr[k] * arr[j]
            if q < dp[i][j]:
                dp[i][j] = q
                s[i][j] = k

optimal_order = GetOptimalOrder (1, N-1)
return [1][N-1] , optimal_order
```

```
ALGO   GetOptimalOrder (i, j):
if  i == j
    return "M" + i
k = split[i][j]   **
left = GetOptimalOrder (i, k)
right = GetOptimalOrder (k+1, j)
return (left x right)
```

TIME COMPLEXITY:

i) There are 3 nested loops in our algorithm:

- Outer loop : $L = 2$ to $N-1$
- Middle loop : $i = 1$ to $N-1$
- Inner loop : $K = i$ to $i+L-2$

ii) Summing up the loop operations

$$T.C = \sum_{L=2}^{N-1} \sum_{i=1}^{N-L} \sum_{K=i}^{i+L-2} O(1)$$

$$= \sum_{L=2}^{N-1} \sum_{i=1}^{N-L} \cancel{i}+L-2-\cancel{i}+1$$

$$=$$

$$\Rightarrow \sum_{L=2}^{N-2} (N-L)(L)$$

$$= O\left(N \cdot \sum_{L=2}^{N-1} L - \sum_{L=2}^{N-2} L^2\right)$$

$$= O\left(\frac{N \cdot N^2}{2} - \frac{N^3}{6}\right)$$

$$\therefore T.C \approx O(N^3)$$

CONCLUSION:

Here, we're using the LCS algorithm to compute the longest common subsequence b/w 2 sequences with a Time Complexity of $O(m*n)$ making it a reliable approach with Dynamic Programming. The MCM algorithm with a Time Complexity of $O(n^3)$ finds the optimal order for multiplying matrices which solves our "meteorological" problem. This algo is more computationally intensive for larger datasets.