**Sardar Patel Institute of Technology**

Bhavan’s Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India

(Autonomous College Affiliated to University of Mumbai)

|  |  |
| --- | --- |
| **PRACTICAL NO. 4** | **DESIGN AND ANALYSIS OF ALGORITHMS** |
| **NAME** | **VEDANTI ANIL WADATKAR** |
| **UID** | **2021700072** |
| **BATCH** | **D4** |
| PROBLEM STATEMENT | Let us consider two matrices X and Y. We want to calculate the resultant matrix Z by multiplying X and Y. |
| EXPLANATION | Let the input sequences be X and Y of lengths m and n respectively. And let dp[n][m] be the length of LCS of the two sequences X and Y.  We iterate through a two dimentional loops of lengths n and m and use the following algorithm to update the table dp[][]:-   * If any of the loop variable i or j is 0 , then dp[i][j] = 0. * if X[i-1] = Y[j-1] ,i.e., when the characters at ith and jth index matches, dp[i][j] = 1 + dp[i-1][j-1]. * Otherwise, store the maximum value we get after considering either the charater X[i] or the character Y[j],i.e.,dp[i][j] = max(dp[i][j-1],dp[i-1][j]). |
| TIME  COMPLEXITY | Since we are using two for loops for both the strings ,therefore the time complexity of finding the longest common subsequence using dynamic programming approach is **O(n \* m)**where n and m are the lengths of the strings |
| function f(i, j):  if i == 0 or j == 0:  return 0  else if A[i] == B[j]:  return 1 + f(i-1, j-1)  else:  return max(f(i-1, j), f(i, j-1))  // the answer is now f(n, m) |
|  |  |
|  |