Vivekanand Education Society's
Institute of Technology
Approved by AICTE & Affiliated to University of Mumbai

Artificial Intelligence and Data Science Department.
AOA / Even Sem 2021-22 / Experiment 8.

YASH SARANG.
47 / D6AD.
EXPERIMENT - 8.

Aim: Write a program for finding the longest common subsequence (LCS) using the Dynamic Programming approach.

Theory:

Given two sequences, find the length of the longest subsequence present in both of them. A subsequence is a sequence that appears in the same relative order but is not necessarily contiguous. For example, “abc”, “abg”, “bdf”, “aeg”, ‘”acefg”, .. etc are subsequences of “abcdefg”.

In order to find out the complexity of the brute force approach, we need to first know the number of possible different subsequences of a string with length n, i.e., find the number of subsequences with lengths ranging from 1,2,..n-1.

Recall from the theory of permutation and combination that a number of combinations with 1 element is nC1. A number of combinations with 2 elements are nC2 and so forth and so on.

We know that nC0 + nC1 + nC2 + … nCn = 2n.
So a string of length n has 2n-1 different possible subsequences since we do not consider the subsequence with length 0. This implies that the time complexity of the brute force approach will be O(n * 2n).

Note that it takes O(n) time to check if a subsequence is common to both the strings. This time complexity can be improved using dynamic programming.

Time Complexity: O(m*n)

CODE:

Code is in the LCS.c file attached along with this doc.

INPUT:

```
char X[] = "AGGTAB";
char Y[] = "GXTXAYB";
```

OUTPUT:

```
Length of LCS is 4
```

CONCLUSION:

The time complexity of the naive recursive approach is O(2^n) in the worst case and the worst case happens when all characters of X and Y mismatch i.e., length of LCS is 0.

The Time Complexity of the above implementation (Dynamic Programming Approach) is O(mn) which is much better than the worst-case time complexity of Naive Recursive implementation.