**DESCRIPTION OF LCS**

Finding the longest subsequence shared by two given sequences is known as the Longest Common Subsequence (LCS) issue. A subsequence is a sequence that doesn't always occur in sequential order, but rather in the same relative order. Applications like text comparison, DNA analysis, and version control systems frequently employ the LCS problem.

Z is the longest common subsequence, and it must be found if we have two sequences, s1 and s2. Z is a subsequence of s1 and s2, and in the original sequences, the elements of Z do not have to appear one after the other. Z also needs to consist of a strictly ascending series of indices in s1 and s2. This means that the indices of the elements chosen from the original sequences must be in ascending order in Z.

**Simple Recursion Solution**:

**Approach:**

A simple recursive solution involves considering all possibilities of including or excluding elements at each index of the sequences and calculating the length of the common subsequence.

**Asymptotic Upper Bound:**

The time complexity of this solution is exponential, specifically O(2^(L1+L2)), where L1 and L2 are the lengths of the input sequences s1 and s2. This is because, at each index, we have two choices: include or exclude the element.

**Dynamic Programming Solution:**

**Approach:**

The dynamic programming solution involves building a 2D table to store the lengths of the LCS for different subproblems. The table is filled in a bottom-up manner, and the final result is obtained from the bottom-right cell of the table.

**Asymptotic Upper Bound:**

The time complexity of the dynamic programming solution is O(L1 \* L2), where m and n are the lengths of the input sequences s1 and s2. This is because we fill in a table of size (L1+1) x (L2+1) in a bottom-up manner, considering all subproblems.