What is the recurrence relation for the longest common subsequence problem?

The longest common subsequence (LCS) problem is a dynamic programming problem that involves finding the longest sequence of characters that appears in two given strings in the same order, but not necessarily consecutively. The recurrence relation for this problem can be defined as:

LCS(i, j) = 0 if i = 0 or j = 0 LCS(i, j) = LCS(i-1, j-1) + 1 if xi = yj LCS(i, j) = max(LCS(i-1, j), LCS(i, j-1)) if xi != yj

Where LCS(i, j) is the length of the LCS for the first i characters of string x and the first j characters of string y, xi is the ith character of string x, yj is the jth character of string y, and i and j are indices of the characters in the two strings. The recurrence relation is defined for i = 1 to m and j = 1 to n, where m and n are the lengths of the two strings.

The base case for this recurrence relation is LCS(0, j) = 0 and LCS(i, 0) = 0, which represents that if one of the strings is empty, then there is no common subsequence between the two strings.

In this recurrence relation, LCS(i, j) represents the length of the LCS for the first i characters of string x and the first j characters of string y. The length of the LCS can be obtained by either adding the last character of both strings if they are equal or by taking the maximum of the LCS for the first i-1 characters of string x and the first j characters of string y or the LCS for the first i characters of string x and the first j-1 characters of string y.

This recurrence relation can be solved using a bottom-up approach by filling a table and then returning the value in the last cell of the table which is the length of the LCS. The time complexity for this solution is O(m*n) where m and n are the lengths of the two strings.

The LCS problem can also be solved using a space-efficient version of the dynamic programming approach called the Space-Optimized LCS, which reduces the space complexity from O(m*n) to O(min(m,n))