CSE 2012- Design and Analysis of Algorithms
In-lab Practice Sheet (Dynamic Prograaming)
Longest Common Subsequence Problem

Practice makes you Perfect

Longest Common Subsequence Problem :

Given two sequences $X =< x_1, x_2, ..., x_m >$ and $Y =< y_1, y_2, ..., y_n >$, task is to compute the maximum-length common subsequence of X and Y.

1. Based on the following logic, Given two sequences $X =< x_1, x_2, ..., x_m >$ and $Y =< y_1, y_2, ..., y_n >$, design a pseudocode that complies with the following logic, to compute the maximum-length common subsequence of X and Y.

    - Generate all the subsequences of X
    - Generate all the subsequences of Y
    - Compute all the common subsequences of X and Y
    - Identify the common subsequence of maximum length.

    we call such a pseudocode as 'non-recursive brute-force algorithm.

2. Given two sequences $X =< x_1, x_2, ..., x_m >$ and $Y =< y_1, y_2, ..., y_n >$, design a brute-force recursive algorithm to compute the maximum-length common subsequence of X and Y. Analyse your algorithm with all the required components .

3. Given two sequences $X =< x_1, x_2, ..., x_m >$ and $Y =< y_1, y_2, ..., y_n >$, design a bottom-up dynamic programming pseudocode to compute the maximum-length common subsequence of X and Y. Your code should print the two tables, the maximum-length of the common subsequence and the common subsequence. Analyse your pseudocode with all the required components .

4. Given two sequences $X =< x_1, x_2, ..., x_m >$ and $Y =< y_1, y_2, ..., y_n >$, design a top-down memoized dynamic programming pseudocode to compute the maximum-length common subsequence of X and Y. Your pseudocode should print a table (used for the memoization process) and the maximum-length of the common subsequence.

5. Modify the bottom-up dynamic programming algorithm to compute the longest common subsequence of the two sequences X and Y such that the modified algorithm uses only one table (i.e., the c-table) and out put the maximum-length common subsequence along with the subsequence.

6. Given a sequence of n numbers, $X =< x_1, x_2, ..., x_n >$, write a pseudocode and an appropriate algorithm to compute the longest monotnically increasing subsequence of X. Given $X =< 10, 22, 9, 33, 21, 50, 41, 60, 80 >$, the longest monotonically increasing subsequence of X is 6 and the longest subsequence is $< 10, 22, 33, 50, 60, 80 >$.

7. Usually the subsequence Z of a sequence X is defined as follows: Given a sequence $X =< x_1, x_2, ..., x_m >$, we say that a sequence $Z =< z_1, z_2, ..., z_k >$ is a subsequence of X if there exists a strictly increasing sequence $< i_1, i_2, ..., i_k >$ of indices of X such that for all $j = 1, 2, 3, ..., k$, we have $x_{ij} = z_j$. For example, $Z =< B, C, D, B >$ is a sub sequence of $< A, B, C, B, D, A, B >$ with a index sequence of $< 2, 3, 5, 7 >$. In the same way, we define Z as an alternative sequence of X if there exists a strictly decreasing sequence $< i_1, i_2, ..., i_k >$. Write a pseudocode and an appropriate code to compute the maximum-length common alternative sequence of X and Y.

8. The bottom-up dynamic programming pseudocode for LCS problem uses two tables, namely, c-table and a b-table. Modify the pseudocode in such a way that the modified algorithm computes only the c-table and does not compute the b-table. Note that the functionality of both the pseudocode and the modified pseudocode should be same. Compare the time-complexity of both the pseudocode and the modified pseudocode.

9. The recursive function for solving LCS is based on the fact: first we check whether the last symbols in X and Y are same or not and accordingly construct the subproblems to proceed further. we call this algorithm as 'last-position-check recursive ' algorithm Instead of checking whether the last symbol of X and Y are same or not. develop a recursive function by checking whether the first symbols of X and Y are same or not and x and construct the subproblems to proceed further. Based on the recursive function developed by you, design a recursive algorithm for LCS. We call this algorithm as ' first-position-checkrecursive ' algorithm. Compare the time-complexity of both the algorithm.

10. Two sequences X and Y are said to be k-similar if k-number of changes are needed in X to make X, same as that of Y. For eg, If X=$ATTGC$, Y=$AGGGC$, then X and Y are 2-similar since there are two chnages are required to make x as Y. Given two equal-length sequences X and Y of symbols from $\{A, T, G, C\}$ and a number k, design a dynamic programming pseudocode to check whether X and Y are 2-similar or not.