Longest Common Subsequence (LCS)[*]

Introduction: Biological applications need to compare DNA sequences. A strand of DNA consists of a string of four possible bases: adenine, guanine, cytosine and thymine. Thus DNA strands can be expressed as arrays, or strings, composed of four symbols, A, C, G, T. It is sometimes important to quantify the similarity between two different strands of DNA. One way to do this is to find the Longest Common Subsequence (LCS).

Definitions and Notation:

- Suppose we have two sequences (arrays) $X[1..n]$ and $Y[1..m]$, where each of $X[i], Y[i]$ are one of the four bases A, C, G, T.
- We say that another sequence $Z[1..k]$ is a sub-sequence if X if there exists a strictly increasing sequence of indices $i_1, i_2, i_3, ..., i_k$ such that we have $X[i_1] = Z[1], X[i_2] = Z[2], ..., X[i_k] = Z[k]$. In other words, a subsequence of X is a sequence of characters from X, not necessarily contiguous, that appear in the same order as they appear in X.
- We say that Z is a *common subsequence* of X and Y if Z is a subsequence of both X and Y.

The LCS Problem: Given two sequences X and Y of size n and m, respectively, find $LCS(X, Y)$, their longest common subsequence of X and Y.

Example: Let $X = [A, B, C, B, D, A, B]$, $Y = [B, D, C, A, B, A]$.

1. List some common subsequences of length 2.
2. List some common subsequences of length 3.
3. Can you find a common subsequence of length 4? 5?

[*]Laura Toma, Algorithms (CSCI 2200), Bowdoin College

4. A brute-force algorithm for finding $LCS(X, Y)$ would enumerate all possibilities. How long would that take? How many possibilities are there?

   You don't need to write this in detail, just sketch the idea and find it's running time.

5. Argue that the problem exhibits subproblem optimality.

6. **Towards a Recursive Formulation:**

   More Notation: For any sequence $X[1..n]$ let X_i denote the sequence consisting of the first i elements of X (usually called the i-prefix): $X_i = X[1..i]$.

   Suppose $LCS(X, Y)$ returns the LCS of X and Y, and that $Z[1..k] = LCS(X, Y)$.

   - Case 1: If $X[n] == Y[m]$: what can you say about $Z[k]$, the last symbol of Z?

And, what can you say about Z_{k-1}? Express it recursively in terms of X_{n-1} and Y_{m-1}.

$Z_{k-1} = LCS(\quad , \quad)$

- Case 2 (a): If $X[n] \neq Y[m]$ and $Z[k] \neq X[n]$, express Z recursively.

  $Z = LCS(\quad , \quad)$

- Case 2 (b): If $X[n] \neq Y[m]$ and $Z[k] \neq Y[m]$,express Z recursively.

  $Z = LCS(\quad , \quad)$

7. Case 2 above assumed you know the last symbol of Z. But, you don't know Z, that's precisely what you are trying to compute. Still, we made progress: What does Case 2(a) and Case 2(b) suggest about computing $LCS(X, Y)$ (when $X[n] \neq Y[m]$)?

   $LCS(X, Y) =$

8. We are now ready to write a recursive algorithm for finding the longest common subsequence. For simplicity, we'll start by computing the *length* of the longest common subsequence of X and Y (unlike $LCS(X, Y)$, which computed the actual longest common subsequence). Then we'll extend the solution to compute the actual LCS as well.

   Notation: Let c(i,j) denote the length of the LCS of X_i and Y_j.

   (Remember that X_i denotes the sequence consisting of the first i elements of X.)

   Given this notation, we want to compute $c(n, m)$.

   Start by writing the Base Cases:

   $c(i, 0) =$

   $c(0, j) =$

Now express $c(i, j)$ recursively, if $X[n] == Y[m]$:

$c(i, j) =$

Next express $c(i, j)$ recursively, if $X[n] \neq Y[m]$:

$c(i, j) =$

Write pseudo-code for a recursive algorithm that computes the length of LCS(X,Y) (that is, $c(i, j)$ as per notation above).

9. Analyze the running time of your algorithm and argue that it is exponential.

10. Argue that we have subproblem overlap in the recursive algorithm.

11. How would you change your pseudocode so as to improve the running time of your algorithm using dynamic programming? Analyze the new running time.

12. How would you extend your algorithm above to compute the LCS not just the length?

13. Consider the following example:

$$X = [A, B, C, B, D, A, B], \quad Y = [B, D, C, A, B, A]$$

Draw the table and show how it's filled when calling your dynamic programming function to compute $c(7, 6)$.