Longest Common Subsequence (LCS)

(CLRS 15.4)

Laura Toma, csci2200, Bowdoin College

Biological applications work with DNA sequences. A strand of DNA consists of a string of molecules which are one of four possible bases: adenine (A), guanine (G), cytosine (C) and thymine (T). Thus DNA sequences can be expressed as arrays or strings over four symbols, A, C, G, T. Biologists want to compare how "close" are two DNA strands, and one way to model closeness is to compute the longest common subsequence of two DNA strands.

The LCS problem: 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 *subsequence* of X if there exists a strictly increasing sequence of indices $i_1, i_2, i_3, ..., i_k$ such that we have $X[i_1] = Z[1], X[i_2] = Z[2], ..., X[i_k] = Z[k]$.
- We say that Z is a *common subsequence* (of X, Y) if Z is a subsequence of both X and Y.

Given two sequences X and Y of size n and m respectively, come up with an algorithm that finds their longest common subsequence (LCS).

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

The questions below will guide you towards a solution. Before you turn to the next page, stop and try to think how you might approach this problem with dynamic programming.

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

2. Sketch a brute-force algorithm for finding $LCS(X, Y)$. For example you could enumerate all possibilities—-How many possibilities are there?

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

3. **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, called the i-prefix: $X_i = X[1..i]$.

   Let $Z[1..k]$ be the LCS of X, Y.

   - Case 1: If $X[n] == Y[m]$: Is the following True or False?

     The last element of Z must be equal to the last element of X and Y: $Z[k] = X[n] = Y[m]$

     Why?

     What can you say about Z_{k-1}? Express it recursively in terms of X_{n-1} and Y_{m-1}:

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

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

     $Z = LCS(?, ?)$

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

     $Z = LCS(?, ?)$

4. We are now ready to write the recursive algorithm for LCS. For simplicity, we'll start by computing *only* the length of the LCS of X, Y. Then we'll extend the solution to compute not only the length, but the actual LCS.

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

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

Start by writing the base case:

$c(i, 0) =?$

$c(0, j) =?$

5. If $X[n] == Y[m]$: Express $c(i, j)$ recursively.

   $c(i, j) =$

6. If $X[n] \neq Y[m]$: Express $c(i, j)$ recursively.

   $c(i, j) =$

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

8. Analyze the running time of your algorithm (assume it does NOT use dynamic programming) and argue that it is exponential.

9. Describe how you can improve the running time of your algorithm using dynamic programming. Analyze the new running time.

10. 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 $c(7, 6)$. You are encouraged to write Python code to help you with this problem.

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