Dynamic Programming Solution to the Longest Common Subsequence Problem

Cheng Li, Virgil Pavlu

[this solution follows "Introduction to Algorithms" book by Cormen et al]

Longest Common Subsequence Problem

Given two sequences $X =< x_1, x_2, \ldots, x_m >$ and $Y =< y_1, y_2, \ldots, y_n >$, find a maximum length common subsequence of X and Y.

Methodology

(1) Characterize the Structure of an Optimal Solution. The LCS problem exhibits optimal substructure in the following manner. Given a sequence $X =< x_1, x_2, \ldots, x_m >$, we define the ith prefix of X, for $i = 0, 1, \ldots, m$, as $X_i =< x_1, x_2, \ldots, x_i >$.

Claim 1 *Let $X =< x_1, x_2, \ldots, x_m >$ and $Y =< y_1, y_2, \ldots, y_n >$ be sequences, and let $Z =< z_1, z_2, \ldots, z_k >$ be any LCS of X and Y.*

1. *If $x_m = y_n$, then $z_k = x_m = y_n$ and Z_{k-1} is an LCS of X_{m-1} and Y_{n-1}.*
2. *If $x_m \neq y_n$, then $z_k \neq x_m$ implies that Z is an LCS of X_{m-1} and Y.*
3. *If $x_m \neq y_n$, then $z_k \neq y_n$ implies that Z is an LCS of X and Y_{n-1}.*

Proof: (1) By contradiction, assume $z_k \neq x_m$, then by appending $x_m = y_n$ to Z, we get a common subsequence of X and Y of length $k + 1$, contradicting the supposed optimality of Z. So $z_k = x_m = y_n$. Thus, the prefix Z_{k-1} is a common subsequence of X_{m-1} and Y_{n-1}. Next we show that it is an LCS. Suppose for the purpose of contradiction that there exists a common subsequence W of X_{m-1} and Y_{n-1} with length greater than $k - 1$. We can append $x_m = y_n$ to W and get a common subsequence of X and Y whose length is greater than k, which contradicting the supposed optimality of Z.

(2) $z_k \neq x_m$ implies that Z is a common subsequence of X_{m-1} and Y. By contradiction, suppose that there is a common subsequence W of X_{m-1} and Y with length greater than k, then W is a common subsequence of X_m and Y, contradicting the supposed optimality of Z.

(3) The proof is similar to (2). □

(2) Recursively Define the Value of the Optimal Solution. Let $C[i, j]$ be the length of an LCS of the sequences X_i and Y_j. If either $i = 0$ or $j = 0$, one of the sequences has length 0, and so the LCS has length 0. If $i, j > 0$ and $x_m = y_n$ we should first find an LCS of X_{m-1} and Y_{n-1} and then append $x_m = y_n$ to this LCS to get an LCS of X and Y. If $i, j > 0$ and $x_m \neq y_n$, then we must first find an LCS of X_{m-1} and Y and an LCS of X and Y_{n-1}, and then choose the longer one as an LCS of X and Y. We thus have the following recurrence.

Claim 2

$$C[i,j] = \begin{cases} 0 & \textit{if } i = 0 \textit{ or } j = 0, \\ C[i-1, j-1] + 1 & \textit{if } i, j > 0 \textit{ and } x_i = y_j, \\ \max(C[i, j-1], C[i-1, j]) & \textit{if } i, j > 0 \textit{ and } x_i \neq y_j. \end{cases}$$

Proof: The correctness of this recursive definition is embodied in the paragraph which proceeds it. □

(3) Compute the Value of the Optimal Solution Bottom-up. Consider the following piece of pseudocode, where $X = < x_1, x_2, \ldots, x_m >$, $Y = < y_1, y_2, \ldots, y_n >$.

```
LCS-LENGTH(X, Y)
1  m = X.length
2  n = Y.length
3  let S[1..m, 1..n] and C[0..m, 0..n] be new tables
4  for i = 1 to m
5      C[i, 0] = 0
6  for j = 0 to n
7      C[0, j] = 0
8  for i = 1 to m
9      for j = 1 to n
10         if x_i == y_j
11             C[i, j] = C[i − 1, j − 1] + 1
12             S[i, j] = "↖"
13         elseif C[i − 1, j] ≥ C[i, j − 1]
14             C[i, j] = C[i − 1, j]
15             S[i, j] = "↑"
16         else C[i, j] = C[i, j − 1]
17             S[i, j] = "←"
18 return C and S
```

Claim 3 *When the above procedure terminates, $C[i, j]$ will contain the length of an LCS of the sequences X_i and Y_j, and $S[i, j]$ will point to the table entry corresponding to the optimal subproblem solution chosen when computing $C[i, j]$.*

Proof: The correctness of the above procedure is based on the fact that it correctly implements the recursive definition given above. The base case is properly handled in Line 4-7, and the recursive case is properly handled in Lines 8 to 17. Note that since the loop defined in Line 8 goes from 1 to m and the loop defined in Line 9 goes from 1 to n, no element of C is accessed in either Line 11,13,14 or 16 before it has been computed. □

(4) Construct the Optimal Solution from the Computed Information. Consider the following piece of pseudocode, where S is the table computed above.

```
PRINT-LCS(S, X, i, j)
if i == 0 or j == 0
    return
if S[i, j] == "↖"
    PRINT-LCS(S, X, i − 1, j − 1)
    print x_i
elseif S[i, j] == "↑"
    PRINT-LCS(S, X, i − 1, j)
elsePRINT-LCS(S, X, i, j − 1)
```

Claim 4 *The above procedure prints out an LCS of X and Y.*

Proof: The above procedure traces through the table by following the arrows. When $S[i, j]$= "↖", $x_i = y_j$ is an element of the LCS, and the procedure will print it out. □

	j	0	1	2	3	4	5	6
i		y_i	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	$\uparrow$ 0	$\uparrow$ 0	$\uparrow$ 0	$\nwarrow$ 1	$\leftarrow$ 1	$\nwarrow$ 1
2	B	0	$\nwarrow$ 1	$\leftarrow$ 1	$\leftarrow$ 1	$\uparrow$ 1	$\nwarrow$ 2	$\leftarrow$ 2
3	C	0	$\uparrow$ 1	$\uparrow$ 1	$\nwarrow$ 2	$\leftarrow$ 2	$\uparrow$ 2	$\uparrow$ 2
4	B	0	$\nwarrow$ 1	$\uparrow$ 1	$\uparrow$ 2	$\uparrow$ 2	$\nwarrow$ 3	$\leftarrow$ 3
5	D	0	$\uparrow$ 1	$\nwarrow$ 2	$\uparrow$ 2	$\uparrow$ 2	$\uparrow$ 3	$\uparrow$ 3
6	A	0	$\uparrow$ 1	$\uparrow$ 2	$\uparrow$ 2	$\nwarrow$ 3	$\uparrow$ 3	$\nwarrow$ 4
7	B	0	$\nwarrow$ 1	$\uparrow$ 2	$\uparrow$ 2	$\uparrow$ 3	$\nwarrow$ 4	$\uparrow$ 4

Figure 1: The C and S tables computed by LCS-LENGTH on the sequence $X =< A, B, C, B, D, A, B >$ and $Y =< B, D, C, A, B, A >$.

(5) Running Time and Space Requirements. The LCS-Length procedure runs in $\Theta(mn)$ since each table entry takes $\Theta(1)$ time to compute, and it uses $\Theta(mn)$ additional space in the form of the tables S and C. The Print-LCS procedure runs in time $O(m + n)$ since it decrements at least one of i and j in each recursive call. It uses no additional space beyond the inputs given. Thus, the total running time is $\Theta(mn)$ and the total space requirement is $\Theta(mn)$.