Dynamic Programming: Longest Common Sub-sequence(LCS) Problem

Course Instructor

Anwar Ghani

PhD Computer Science

anwar.phdcs70@iiu.edu.pk

DEPARTMENT OF COMPUTER SCIENCE & SOFTWARE ENGINEERING

INTERNATIONAL ISLAMIC UNIVERSITY, ISLAMABAD

1 Introduction

As we discussed in our previous lecture in order to apply Dynamic Programming DP to a problem the problem must have the following characteristics:

- Optimal Sub-structure
- Polynomial many sub-problems

Then we can solve the problem by applying DP technique which solves a problem in the following steps:

1. Define the optimal solution for a problem
2. Setup a recurrence for the given problem
3. Implement the rule
4. Extract the solution (Bottom up approach)

As an example we are going to apply these steps to well known problem known as the longest common subsequence problem in next subsection. LCS is usually applicable in the field of genetics where it can be used to find out how much two strings of genes similar or differ from each other. It can also be applied in searching an matching algorithms where we want to check how much to document are similar.

1.1 Subsequence

Given a sequence $X =< x_1, x_2, x_3, ..., x_m >$, another sequence $Z =< z_1, z_2, z_3, ..., z_k >$ is a subsequence of x if there exists a strictly increasing sequence $< i_1, i_2, i_3, ..., i_k >$ of indices of x such that for all $j = 1, 2, 3, ..., k$, we have $X_{i,j} = Z_j$.

For Example: if $X =< A, C, B, D, A, B >$ then $Z =< A, B, A, B >$ is a subsequence of x.

1.2 Longest Common Sub-sequence

Given two sequences $X =< x_1, x_2, x_3, ..., x_m >$ and $Y =< y_1, y_2, y_3, ..., y_n >$, we wish to find the longest common sub-sequence, which is common to both X and Y.

For Example: $X =< A, B, C, B, D, A, B >$ and $Y =< B, D, C, A, B, A >$ then the common sub-sequence for X and Y is

$$Z = \begin{cases} < B, D, A, B > \\ < B, C, A, B > \\ < B, C, B, A > \end{cases}$$

Which shows that the two sequences may have more than one LCS's in common. The basic question is how do we find the longest common subsequence of the given sequences. One approach that come immediately into mind is the Brute force approach while ofcourse the other one is to use Dynamic programming.

1.3 Brute force solution

The brute force approach try all possible subsequences and X against the subsequences of Y. But this approach is not feasible since it enumerates all the subsequences of X, and checks if it exists in Y. As each subsequence of X corresponds to the indices 1,2,3,.....,m, of X therefore there are 2^m subsequences, which means that this approach requires exponential time. So in case of long sequences the it will become infeasible.

1.4 Dynamic Programing Approach

1.4.1 Step-I: Characterizing a Longest Common Sub-sequence

This approach exploits the optimal sub-structure property of LCS which corresponds to pairs of "prefixes" of the two input sequences.

For example:

In the sequence $X =< x_1, x_2, x_3, ..., x_m >$ the i^{th} prefix of X, for $i = 0, 1, 2, 3, ..., m$, as $X_i =< x_1, x_2, x_3, ..., x_i >$

Let c[i,j] denotes the length of LCS of X_i, Y_j. Eventually we want c[m,n].

Basic Idea:

The basic idea is to compute the c[i,j] assuming that we know $c[i', j']$ for all $i' < i$ and $j' < j$.

Observation:

If one sequence does not exist then the length of LCS is zero. therefore $c[o, j] = c[i, o] = 0$ if either of the sequence is empty.

1.4.2 Step-II: Last Character Matches

Case-I:

If last character of the two given sequences match then we have $X_i = Y_j$. X ends in X_i and Y ends in Y_j , we claim that LCS of both sequences contain X_i (as $X_i = Y_j$). Remove X_i from both the sequences and take LCS of X_{i-1} and Y_{j-1}. Add X_i to LCS of X_{i-1} and Y_{j-1} then

$$c[i, j] = c[i - 1, j - 1] + 1 \tag{1}$$

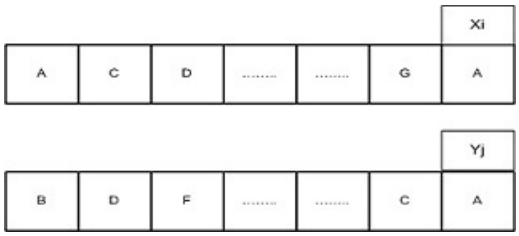

Figure 1: Last character matches

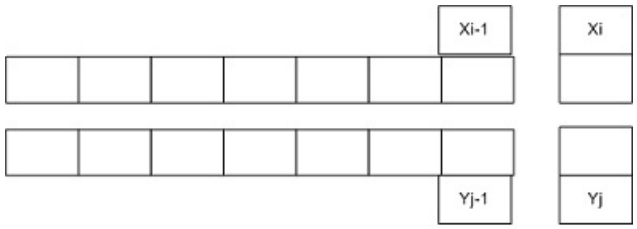

Figure 2: When Xi = Yj

Case-II: Last Character Does not Match

When the last characters of the two sequences does not match then we have $X_i \neq Y_j$, means either X_i or Y_j is not part of the LCS. Here we have two cases:

Case-A: If $X_i \notin LCS$

then the LCS of X_i and Y_j is the LCS of X_{i-1} and Y_j then

$$c[i,j] = c[i-1,j] \tag{2}$$

Case-B: If $Y_j \notin LCS$

Then the LCS of X_i and Y_j is the LCS of X_i and $Y_z j-1$ then

$$c[i,j] = c[i,j-1] \tag{3}$$

Now in case where $X_i \neq Y_j$ and $i,j > 0$ the LCS is the maximum of equations **??**. Therefore

$$c[i,j] = max(c[i,j-1], c[i-1,j]) \tag{4}$$

Now let us setup a recursion for the LCS problem from case-I and II.

$c[i,j] = c[i-1,j-1]+1$, if $i,j > 0$ and $X_i = Y_j$

and

$c[i,j] = Max(c[i,j-1], c[i-1,j])$, if $i,j > 0$ and $X_i \neq Y_j$.

Therefore the recursive formula for the LCS problem becomes

$$c[i,j] = \begin{cases} 0 & \text{if } i=0, \text{ or } j=0 \\ c[i-1,j-1]+1 & \text{if } i,j>0, \text{ and } X_i = Y_j \\ Max(c[i,j-1], c[i-1,j]) & \text{if } i,j>0, \text{ and } X_i \neq Y_j \end{cases}$$

1.4.3 Step-III: Implementing the Rule

Based on the recurrence derived in the previous section we could easily write an exponential time recursive algorithm. As we are interested in an optimal solution and there are only $\theta(mn)$ sub-problem so we can apply the dynamic programming to compute the LCS bottom up.

The following pseudo code LCS-LENGTH takes two argument X and Y which are the two input sequences and stores c[i,j] values in c[0...m,0...n] whose entries are computed in row major order. Row major mean the first row of the c[i,j] is filled in from left to right then the second row and so on. To simplify the construction of the LCS it maintains another table b[i,j] for constructing an optimal solution.

Source for the this pseudo code is (Corman et. al).

Algorithm 1 LCS-LENGTH(X,Y)

```
1: m ← length[X]
2: m ← length[Y]
3: for i = 1 → m do
4:    c[i, o] ← 0
5: end for
6: for j = 1 → n do
7:    c[0, j] ← 0
8: end for
9: for i = 1 → m do
10:   for j = 1 → n do
11:      if X_i = Y_j then
12:         c[i, j] ← c[i − 1, j − 1] + 1
13:         b[i, j] ← " ↖ "
14:      else if c[i − 1, j] ≥ c[i, j − 1] then
15:         c[i, j] ← c[i − 1, j]
16:         b[i, j] ← " ↑ "
17:      else
18:         c[i, j] ← c[i, j − 1]
19:         b[i, j] ← " ← "
20:      end if
21:   end for
22: end for
23: return c & b
```

1.4.4 Step-IV: Extracting the Sequence(Bottom Up)

Here the b table returned by the LCS-LENGTH procedure to quickly construct an LCS of $X =< x_1, x_2, x_3, ..., x_m)$ and $Y =< y_1, y_2, y_3, ..., y_n >$Source for the this pseudo code is (Corman et. al).

Algorithm 2 PRINT-LCS(b, X, i, j)

if $i = 0$ or $j = 0$ **then**
  $return$
end if
if $b[i, j] =$ " $\nwarrow$ " **then**
  $PRINT - LCS(b, X, i - 1, j - 1)$
  $printX_i$
else if $b[i, j] =$ " $\uparrow$ " **then**
  $PRINT - LCS(b, X, i - 1, j)$
else
  $PRINT - LCS(b, X, i, j - 1)$
end if

Here is an example exercise number 15.4-1 of your book page number 355.

Exercise 15.4-1. Determine an LCS of $< 1, 0, 0, 1, 0, 1, 0, 1 >$ and $< 0, 1, 0, 1, 1, 0, 1, 1, 0 >$.

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 & 3 \\ 0 & 1 & 2 & 2 & 3 & 3 & 3 & 4 & 4 & 4 \\ 0 & 1 & 2 & 3 & 3 & 3 & 4 & 4 & 4 & 5 \\ 0 & 1 & 2 & 3 & 4 & 4 & 4 & 5 & 5 & 5 \\ 0 & 1 & 2 & 3 & 4 & 4 & 5 & 5 & 5 & 6 \\ 0 & 1 & 2 & 3 & 4 & 5 & 5 & 6 & 6 & 6 \end{bmatrix}, B = \begin{bmatrix} \uparrow & \nwarrow & \leftarrow & \nwarrow & \nwarrow & \leftarrow & \nwarrow & \nwarrow & \leftarrow \\ \nwarrow & \uparrow & \nwarrow & \leftarrow & \leftarrow & \nwarrow & \leftarrow & \leftarrow & \nwarrow \\ \nwarrow & \uparrow & \nwarrow & \uparrow & \uparrow & \nwarrow & \leftarrow & \leftarrow & \nwarrow \\ \uparrow & \nwarrow & \uparrow & \nwarrow & \nwarrow & \uparrow & \nwarrow & \nwarrow & \leftarrow \\ \nwarrow & \uparrow & \nwarrow & \uparrow & \uparrow & \nwarrow & \uparrow & \uparrow & \nwarrow \\ \uparrow & \nwarrow & \uparrow & \nwarrow & \nwarrow & \uparrow & \nwarrow & \nwarrow & \uparrow \\ \nwarrow & \uparrow & \nwarrow & \uparrow & \uparrow & \nwarrow & \uparrow & \uparrow & \nwarrow \\ \uparrow & \nwarrow & \uparrow & \nwarrow & \nwarrow & \uparrow & \nwarrow & \nwarrow & \uparrow \end{bmatrix}$$

This implies that an LCS of S_1 and S_2 is the sequence $< 1, 0, 0, 1, 1, 0 >$. Another one is $< 1, 0, 1, 1, 0, 1 >$.

Figure 3: Solution to exercise 15.4-1