Design and Analysis of Algorithms

Lecture-34

Dharmendra Kumar (Associate Professor)
Department of Computer Science and Engineering
United College of Engineering and Research,
Prayagraj

Longest Common Subsequence Problem (LCS problem)

In the ***longest-common-subsequence problem***, we are given two sequences $X = < x_1, x_2, \ldots\ldots\ldots, x_m >$ and $Y = < y_1, y_2, \ldots\ldots\ldots, y_n >$ and wish to find a maximum length common subsequence of X and Y.

Longest Common Subsequence Problem (LCS problem)

Example:

- If X = < A, B, C, B, D, A, B > and Y= < B, D, C, A, B, A >, the sequence < B, C, A > is a common subsequence of both X and Y .
- The sequence < B, C, A > is not a *longest* common subsequence (LCS) of X and Y , however, since it has length 3 and the sequence < B, C, B, A >, which is also common to both X and Y , has length 4.
- The sequence < B, C, B, A > is an LCS of X and Y , as is the sequence < B, D, A, B >, since X and Y have no common subsequence of length 5 or greater.

Solving the LCS problem using dynamic programming

Step-1: Characterizing a longest common subsequence

***Prefix of* a sequence:**

Given a sequence $X = < x_1, x_2,\ldots\ldots\ldots, x_m >$, we define the i^{th} ***prefix*** of X, for i = 0, 1, 2, ..., m, as $X_i = < x_1, x_2,\ldots\ldots\ldots, x_i >$.

For example, if X = < A, B, C, B, D, A, B >, then X4= <A, B, C, B > and X_0 is the empty sequence.

Solving the LCS problem using dynamic programming

Optimal substructure of an LCS

Let $X = < x_1, x_2, \ldots\ldots\ldots, x_m >$ and $Y = < y_1, y_2, \ldots\ldots\ldots, y_n >$ be sequences, and let $Z = < z_1, z_2, \ldots\ldots\ldots, z_k >$ 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}.

Solving the LCS problem using dynamic programming

Step 2: A recursive 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. The optimal substructure of the LCS problem gives the recursive formula:-

$c[i,j] = 0$ if $i = 0$ or $j = 0$

$= c[i-1,j-1]+1,$ if $i, j > 0$ and $x_i = y_j$

$= \max\{ c[i-1, j], c[i,j-1] \},$ if $i, j > 0$ and $x_i \neq y_j$

Let b is the matrix which stores one of the following three arrows, ←, ↑ and ↖.

b[i,j] = ← , if c[i,j] = c[i,j-1]

b[i,j] = ↑ , if c[i,j] = c[i-1,j]

b[i,j] = ← , if c[i,j] = c[i-1,j-1]

LCS problem using dynamic programming

Step-3: Computing the length of an LCS

Following procedure is used to compute the table b and c.

```
LCS-LENGTH(X, Y)
m = X.length
n = Y.length
let b[1..m, 1..n] and c[0..m, 0..n] be new tables
for i = 1 to m
    c[i, 0] = 0
for j = 0 to n
    c[0, j] = 0
for i = 1 to m
    for j = 1 to n
        if x_i == y_j
            c[i, j] = c[i - 1, j - 1] + 1
            b[i, j] = "↖"
        elseif c[i - 1, j] ≥ c[i, j - 1]
            c[i, j] = c[i - 1, j]
            b[i, j] = "↑"
        else c[i, j] = c[i, j - 1]
            b[i, j] = "←"
return c and b
```

LCS problem using dynamic programming

Step-3: Computing the length of an LCS.

Example: Find LCS of the following sequences:-

X = < A, B, C, B, D, A, B > and Y = < B, D, C, A, B, A >.

Solution: We compute the table corresponding to b and c as following:-

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

LCS problem using dynamic programming

Step-4: Constructing an LCS

- The following recursive procedure prints out an LCS of X and Y in the proper, forward order. The initial call is PRINT-LCS(b, X, m, n).

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

LCS problem using dynamic programming

Step-4:

Time complexity of this algorithm = O(m+n).

The LCS of sequences taken in previous example = BCBA.