Longest Common Subsequence

November 23, 2020

The LCS problem

- Given two strings X and Y , the longest common subsequence of X and Y is a longest sequence Z which is both a subsequence of X and Y.
- For example, For example, if $X = ABCBDAB$ and $Y = BDCABA$, the sequence BCA is a common subsequence of both X and Y with length 3 and the sequence $BCBA$ which is also a common subsequence , has length 4 and it is the LCS.

The Optimal substructure

- Let $X = x_1, x_2 \dots x_m$ and $Y = y_1, y_2 \dots y_n$ be two sequences, and let $Z = z_1, z_2, \dots z_k$ 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}

     If the last element of both sequences are same, then it is the last element of the LCS and the sequence Z_{k-1} is the LCS of X_{m-1} and Y_{n-1}

  2. $x_m \neq y_n$ then $z_k \neq x_m$ implies that Z is an LCS of X_{m-1} and Y.

     If the Last element of X and Y are different and if $z_k \neq x_m$ then Z is a common subsequence of X_{m-1} and Y.

  3. $x_m \neq y_n$ then $z_k \neq y_n$ implies that Z is an LCS of X and Y_{n-1}

     If the Last element of X and Y are different and if $z_k \neq y_n$ then Z is a common subsequence of Y_{n-1} and X.

Recursive Formulation

- Let $c[i, j]$ be length of the LCS of the sequences X_i and Y_j.

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

- In the algorithm, we use two tables
(1) table c that helps to compute the length of LCS and
(2) table b that helps to compute the LCS.

LCS Length Algorithm

```
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
```

C and B Tables

	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

if $x_i == y_j$
$c[i, j] = c[i-1, j-1] + 1$
$b[i, j] =$ "↖"
elseif $c[i-1, j] \geq c[i, j-1]$
$c[i, j] = c[i-1, j]$
$b[i, j] =$ "↑"
else $c[i, j] = c[i, j-1]$
$b[i, j] =$ "←"

A recursive algorithm to print LCS

```
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)
```