DP formulation

we need to break the problem into smaller pieces.

All pairs of prefixes will suffice for us.

A prefix of a sequence is just an initial string of values, $X_i = \{x_1 - - x_i\}$

X_0 is the empty sequence.

The idea is compute the longest common subsequences for every possible pair of prefixes.

Let $lcs(i, J)$ denote the length of the longest common subsequence of X_i and Y_j.

For example for the string

X = A B R A C A D A B R A

$\geq$ LCS = $\boxed{\text{ABADABA}}$

Y = Y A B B A D A B B A D O O

X_5 = A B R A C

Y_6 = Y A B B A D

LCS (5, 6) = 3 = {A B A}

<u>Ex 2</u>

$$
\begin{array}{llllll}
 & & & i-1 & i & \\
X_i: & \boxed{A \quad B \quad C} & & & D & - - - - \\
 & & & j-1 & & \\
Y_j: & \boxed{A \quad C \quad B} & & & D &
\end{array}
$$

To apply D.P we have to assume that we have found the LCS for sub problem.

Solution/s to the sub-problem

$\boxed{A \quad C}$ or $\boxed{A \quad B}$ is given

→ what would be solution of (i, j)

$$
LCS[i, j] = \begin{cases} LCS[i-1][j-1] + 1 & \text{if } X[i] == Y[j] \\ \boxed{LCS[i-1][j-1] \text{ if } X[i] \neq Y[j]} \times & \\ \max(LCS[i-1][j],\ LCS[i][j-1]) & \text{if } X[i] \neq Y[j] \\ 0 & \text{if } i = 0 \text{ or } j = 0 \end{cases}
$$

$$X_i = \boxed{A \quad B \quad C}_{\,i-1} \quad D_{\,i}$$

$$Y_j = \boxed{B \quad A \quad D}_{\,j-1} \quad A_{\,j}$$

$LCS[i-1][j-1] = 1$

$LCS[i][j] = 2$

So we have to check

$$\boxed{\begin{array}{c} LCS[i][j-1] = 2 \\ \text{and} \\ LCS[i-1][j] = 1 \end{array}}$$

$\downarrow X[i]$ $\quad Y[j] \longrightarrow$

	None	B	D	C	A	B
None	0	0	0	0	0	0
A	0	0	0	0	1	1
B	0	1	1	1	1	2
C	0	1	1	2	2	2
B	0	1	1	2	2	3
D	0	1	2	2	2	3

Ans - 3

$f[i][j] = LCS(X_i, Y_j)$

$$LCS[i][j] \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ LCS[i-1][j-1] + 1 & \text{if } X[i] == Y[j] \\ \max(LCS[i-1][j], LCS[i][j-1]) & \text{if } X[i] \# Y[i] \end{cases}$$

for character follow the "Arrow".

Algo

```
LCS (X, Y)
{
  m = X.length ,  n = Y.length

  create a 2D array t with m+1 row
  & n+1 column. (Take anyone as X and Y)

  fill the first row and first column with
  0s.

  for (i = 1 to m)
  {
    for (j = 1 to n)
    { if X[i] == Y[j]
         t[i][j] = t[i-1][j-1] + 1

      else
         t[i][j] = max (t[i-1][j], t[i][j-1])
    }
  }
```

$\Theta(n.m)$ (Because input size and number of iteration is linear so the relation between this is polynomial)

Construct LCS
```
{
    i=m , j=n

    while (i>=1 & j>=1)
    {
        if   x[i] == y[j]
             Add (x[i] to the LCS)
             i-- , j--
        else
             if  f[i-1][j] > f[i][j-1]
                 i--;
             else
                 j--;
    }
}
```

~~O(m+n)~~ O(m+n)