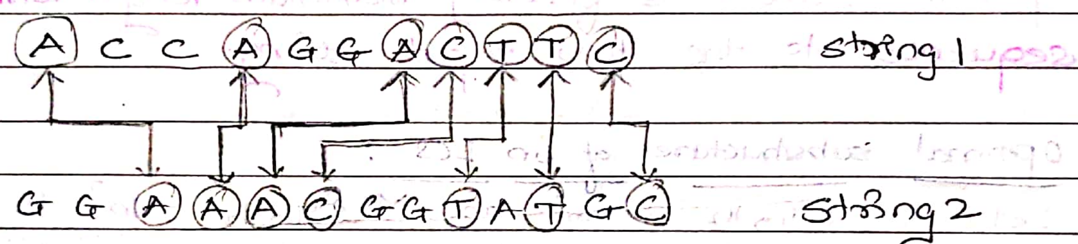
Dynamic Programming

LONGEST COMMON SUBSEQUENCE (LCS)

Application of LCS :

- Comparing DNA string to see how similar they are.
- Given, 2 DNA string, there is a need to find a third string in which the bases (symbols) appear in both original string in same order but not necessarily consecutively.
- As shown in diagram, it has 2 strings & the LCS string.



A A A C T T C . string 3 - LCS

Problem definition :

- Given the first sequence which contains m symbols i.e $X = (x_1, x_2, x_3 \dots x_m)$
- Given the second sequence which contains n symbols i.e $Y = (y_1, y_2, y_3 \dots y_n)$
- Problem is to find longest common subsequence (LCS) i.e Z between X and Y i.e $Z = (z_1, z_2, z_3 \dots z_k)$.

- Example, Let
  $X = \{A, B, C, A, D, B\}$ $Y = \{C, B, E, C, A\}$
  Subsequence {B, C} is common in X&Y but length is 2 whereas {B, C, A} is also a common subsequence with length 3
  ∴ LCS is {B, C, A}

* Notations

1. x_i means symbol at position 'i' in sequence X.
   eg if $X = \{A, B, C, D\}$ then $x_2 = B$.
2. X_i means the subsequence in X of all symbols starting at x_1 ending at x_i.

   eg: If $X = \{A, B, C, D\}$ then $X_2 = \{A, B\}$
3. Same points are applied for second sequence Y and LCS sequence Z.

* I] Characterize the structure of optimal solution

Optimal solution is finding maximum length common subsequence to the two input string.

Optimal substructure of an LCS :

- Let $X = \{x_1, x_2 \dots x_m\}$ and $Y = \{y_1, y_2 \dots y_n\}$ be 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}

   i.e, (i) If last symbol of both the sequence is same then it is the last symbol of LCS too.

   (ii) String resulting from removing last symbol from LCS will also be a LCS for both strings after removing the last symbol.

   eg: $X = \{A, B, C, D\}$ $\therefore x_4 = y_4 = z_3 = D$.

   $Y = \{B, A, B, D\}$

   $Z = \{A, B, D\}$

   Now, $Z_{k-1} = \{A, B\}$ which is LCS for $X_{m-1} = \{A, B, C\}$ & $Y_{m-1} = \{B, A, B\}$

* LCS of 2 sequences contains within it an LCS of prefixes of two sequences → optimality

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.

i.e if last symbol of X and Y are different and even if last symbol of Z and X are also different the it implies that Z is an LCS of Y and X after removing last symbol x_m from X.

eg: $X = \{A\ B\ C\ D\}$; $m=4$

$Y = \{B\ A\ B\ B\}$; $n=4$

$Z = \{A\ B\}$; $k=2$

here, $x_4 \neq y_4$ & $x_4 \neq z_2$

∴ Z is also an LCS for Y and

$X_3 = \{A, B, C\}$; $m=3$.

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

[II] Recursively define optimal solution

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

- Recursive formula for optimal substructure of the LCS is

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

Scanned by CamScanner

III] Compute the length of an LCS (optimal solution)

- Algorithm takes two sequences $X = \{x_1, x_2 \dots x_m\}$ and $Y = \{y_1, y_2, \dots y_n\}$ as inputs.
- Stores $c[i,j]$ value in a table $c[0 \dots m, 0 \dots n]$ and computes values in row major order fashion.
- For constructing the optimal solution: table $b[1 \dots m, 1 \dots n]$ is used.
- $b[i,j]$ → corresponds to the table entry of optimal subproblem solution chosen when computing $c[i,j]$.
- At end, $c[m,n]$ → length of an LCS of X and Y.

ALGORITHM:

```
LCS_Length (X, Y)
        m = X.length
        n = Y.length
        let b[1..m, 1..n] and c[0..m, 0..n] be a new tables.
Θ(m)    for i = 0 to m
            c[i, 0] = 0
Θ(n)    for j = 0 to n
            c[0, j] = 0

Θ(m)    for i = 1 to m
Θ(n*m)      for j = 1 to n
                if xi == yj
                    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 b and c
```

Analysis :-

Running time of this procedure is $O(m \times n)$, since each entry takes $O(1)$ time to compute.

[IV] Constructing an optimal solution ie LCS

Inputs are :- (i) back pointer / table $b[1 \ldots m, 1 \ldots n]$
(ii) One of the i/p sequence i.e X or Y
(iii) length of both sequence ie X.length and Y.length.

Procedure :-

X.length
Y.length

```
Print_LCS (b, X, i, j)
1.  if i == 0 or j == 0
2.      return

3.  if b[i, j] == "↖"
4.      Print_LCS (b, X, i-1, j-1)
5.      print xi

6.  else if b[i, j] == "↑"
        Print_LCS (b, X, i-1, j)

7.  else PrintLCS (b, X, i, j-1)
```

This procedure takes $O(m+n)$ time, since it decrement atleast one of i and j in each recursive call.

Problem :-

Determine an LCS of $X = \{A, B, C, B, D, A, B\}$ and $Y = \{B, D, C, A, B, A\}$

$\Rightarrow$ Solution : $m = 7$ $n = 6$

1] LCS Length

		0	1	2	3	4	5	6
		y_i	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

So, length of LCS $= c[m, n] = c[7, 6] = 4$

2] Print LCS :

$\boxed{B\ C\ B\ A}$

Recursive call

print_LCS (b, X, 7, 6)

↓

print_LCS (b, X, 6, 6) - print x_i

↓

print_LCS (b, X, 5, 5)

↓

print_LCS (b, X, 4, 5) - print x_i

↓

print_LCS (b, X, 3, 4)

↓

print_LCS (b, X, 3, 3) - print x_i

↓

print_LCS (b, X, 2, 2)

↓

print_LCS (b, X, 2, 1) - print x_i

↓

print_LCS (b, X, 1, 0)

Scanned by CamScanner

Problem 2 :-

Determine an LCS of $X = \{1,0,0,1,0,1,0,1\}$ and $Y = \{0,1,0,1,1,0,1,1,0\}$

$m = 8$

$n = 9$

			1	2	3	4	5	6	7	8	9
	y_j		0	1	0	1	1	0	1	1	0
	x_i	0	0	0	0	0	0	0	0	0	0
1	1	0	↑ 1	↖ $\boxed{1}$	↑ 1	↖ 1	↖ 1	← 1	↖ 1	↖ 1	← 1
2	0	0	↖ 1	↑ 1	↖ $\boxed{2}$	← $\boxed{2}$	← $\boxed{2}$	↖ 2	← 2	← 2	↖ 2
3	0	0	↖ 1	↑ 1	↖ 2	↑ 2	↑ 2	↖ $\boxed{3}$	← 3	← 3	↖ 3
4	1	0	↑ 1	↖ 2	↑ 2	↖ 3	↖ 3	↑ 3	↖ $\boxed{4}$	↖ 4	← 4
5	0	0	↖ 1	↑ 2	↖ 3	↑ 3	↑ 3	↖ 4	↑ 4	↑ 4	↖ 5
6	1	0	↑ 1	↖ 2	↑ 3	↖ 4	↖ 4	↑ 4	↖ 5	↖ $\boxed{5}$	↑ 5
7	0	0	↖ 1	↑ 2	↖ 3	↑ 4	↑ 4	↖ 5	↑ 5	↑ 5	↖ $\boxed{6}$
8	1	0	↑ 1	↖ 2	↑ 3	↖ 4	↖ 5	↑ 5	↖ 6	↖ 6	↑ 6

So, length of LCS = $c[m,n] = c[8,9] = 6$

0.

[1 0 0 1 1 0]

Step 2 printing LCS

Recursive calls

print_LCS (b, X, 8, 9) → print x_i 1

↓

print_LCS (b, X, 7, 9) → print x_i 0

↓

print_LCS (b, X, 6, 8) → print x_i 1

↓

print_LCS (b, X, 5, 7)

↓

print_LCS (b, X, 4, 7) — print x_i 1

↓

print_LCS (b, X, 3, 6) → print x_i 0

↓

print LCS (b, X, 2, 5)

↓

print LCS (b, X, 2, 4)

↓

print LCS (b, X, 2, 3) → print x_i 0

↓

print LCS (b, X, 1, 1) → print x_i 1

↓

print LCS (b, X, 0, 0)
return