LONGEST COMMON SUBSEQUENCE (LCS)

→ A subsequence is any subset of the elements of a sequence that maintains the same relative order.

→ If A is a subsequence of B, this is denoted by $A \subset B$

→ Example: If $A = a_1\ a_2\ a_3\ a_4\ a_5$, the sequence $A' = a_2 a_4 a_5$ is a subsequence of A.

$A'' = a_2\ a_1\ a_5$ is not a subsequence of A bcoz a_2 and a_1 are reversed.

→ The longest common subsequence (LCS) of two sequences A and B is a sequence C such that $C \subset A$ and $C \subset B$ and $|C|$ is maximum.

→ Example:

A = abacdac   B = cadcddc

a b a c d a c

c a d c d d c

The LCS is c = acdc

Applications of LCS:

1. Molecular biology.
2. File comparison.
3. Screen display.

→ In LCS problem, two sequences

$X = \langle x_1, x_2, \cdots x_m \rangle$ and

$Y = \langle y_1, y_2, \cdots y_n \rangle$ are given. we wish to find a maximum length common subsequence of X and Y.

→ Now we are applying dynamic programming to solve the longest common subsequence problem.

Step 1: Characterizing a longest common subsequence

The LCS problem has an optimal substructure property.

Theorem: Optimal substructure for LCS

Let $X = \langle x_1, x_2, \cdots, x_m \rangle$ and $Y = \langle y_1, y_2, \cdots, y_n \rangle$ be the and $Z = \langle z_1, z_2, \cdots, z_k \rangle$ 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}.

Step 2: A recursive solution

→ There are either one or two subproblems to study when finding an LCS of $X = \langle x_1, x_2, \ldots, x_m \rangle$ and $Y = \langle y_1, y_2, \ldots, y_n \rangle$

→ If $x_m = y_n$ then find the LCS of X_{m-1} and Y_{n-1} and append $x_m = y_n$ to this to get the LCS of X and Y.

→ If $x_m \neq y_n$,

(a) find the LCS of X_{m-1} and Y

(b) find the LCS of X and Y_{n-1}.

whichever of these two LCS's is longer is an LCS of X and Y.

→ 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. So the LCS has length 0.

→ The optimal substructure of the LCS is:

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

Step-3: Computing the length of an LCS:

Here two sequences $X = \langle x_1, x_2, \ldots, x_m \rangle$ and $Y = \langle y_1, y_2, \ldots, y_n \rangle$ are inputs.

It store $c[i,j]$ values in a table $c[0..m, 0..n]$ whose entries are computed in row-major order.

It also maintains the table $b[1..m, 1..n]$ whose $b[i,j]$ points to the table entry corresponding to the optimal subproblem solution

The procedure returns the b and c tables. $c[m,n]$ contains the length of an LCS of X and Y.

```
LCS-LENGTH(X, Y)
m ← length[X]
n ← length[Y]
for i ← 1 to m
    do c[i,0] ← 0
for j ← 0 to n
    do c[0,j] ← 0
for i ← 1 to m.
    do for j ← 1 to n
        do if xi = yj
            then c[i,j] ← c[i-1, j-1]+1
                 b[i,j] ← "↖"
        else if c[i-1, j] ≥ c[i, j-1]
            then 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.
```

$j \rightarrow$

i ↓		0	1	2	3	4	5	6
		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	$\boxed{0}$	↑0	↑0	↑0	↖1	←1	↖1
2	B	0	$\boxed{↖1}$	$\boxed{←1}$	←1	↑1	↖2	←2
3	C	0	↑1	↑1	$\boxed{↖2}$	$\boxed{←2}$	↑2	↑2
4	B	0	↖1	↑1	↑2	↑2	$\boxed{↖3}$	←3
5	D	0	↑1	↖2	↑2	↑2	$\boxed{↑3}$	↑3
6	A	0	↑1	↑2	↑2	↖3	↑3	$\boxed{↖4}$
7	B	0	↖1	↑2	↑2	↑3	↖4	$\boxed{↑4}$

→ The LCS is ⟨B, C, B, A⟩ of X and Y.

→ Since each entry of table takes O(1) time to compute
The running time of this procedure is O(mn).

Step-4: Constructing an LCS.

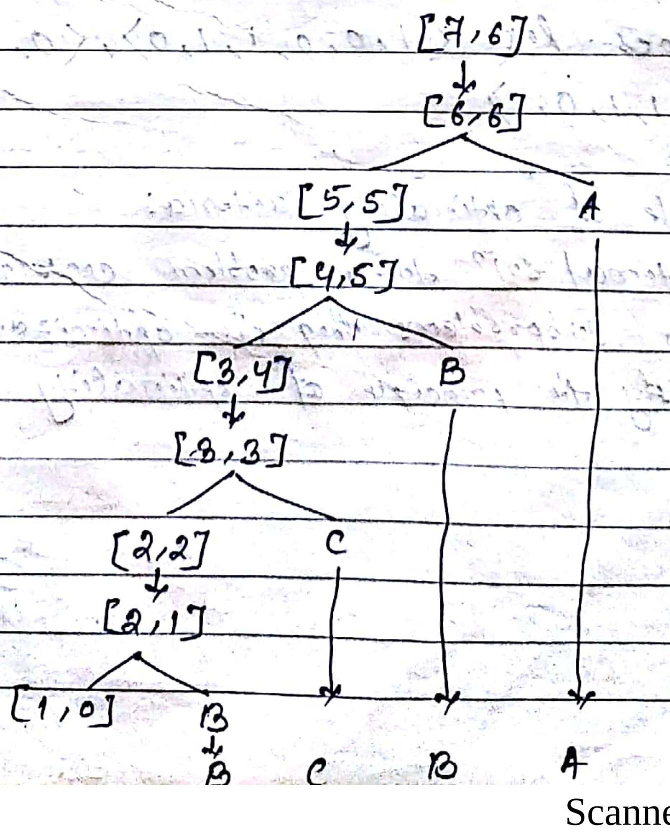
→ The b table is used to construct an LCS of $X = \langle x_1, x_2, \cdots, x_m \rangle$ and $Y = \langle y_1, y_2, \cdots, y_n \rangle$

→ we begin at b[m,n] and trace through the table followed by arrows.

→ when we encounter a "↖" in entry b[i,j], it implies that $x_i = y_j$ is an element of the LCS. The elements of the LCS are encountered in reverse order. But we prints out an LCS of X and Y in forward order.

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

This procedure takes time $O(m+n)$, since at least one of i and j is decremented in each stage of the recursion.



∴ LCS is ⟨B, C, B, A⟩

Example: Determine the LCS of ⟨1,0,0,1,0,1,0,1⟩ and ⟨0,1,0,1,1,0,1,1,0⟩

solution:

Let X = ⟨1,0,0,1,0,1,0,1⟩ and Y = ⟨0,1,0,1,1,0,1,1,0⟩

we know that,

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

		0	1	2	3	4	5	6	7	8	9
	y_i		0	1	0	1	1	0	1	1	0
0	x_i	0	0	0	0	0	0	0	0	0	0
1	1	0	↑ 0	↖ 1	← 1	↖ 1	↖ 1	← 1	↖ 0	↖ 1	← 1
2	0	0	↖ 1	↑ 1	↖ 2	← 2	← 2	↖ 2	← 2	← 2	↖ 2
3	0	0	↖ 1	↑ 1	↖ 2	↑ 2	↑ 2	↖ 3	← 3	← 3	↖ 3
4	1	0	↑ 1	↖ 2	↖ 2	↖ 3	↖ 3	↑ 3	↖ 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	↖ 5	↑ 5
7	0	0	↖ 1	↑ 2	↖ 3	↑ 4	↑ 4	↖ 5	↑ 5	↑ 5	↖ 6
8	1	0	↑ 1	↖ 2	↑ 3	↖ 4	↖ 5	↑ 5	↖ 6	↖ 6	↑ 6

we can deduce that LCS = 6. There are several such sequences like ⟨1,0,0,1,1,0⟩, ⟨0,1,0,1,0,1⟩ and ⟨0,0,1,1,0,1⟩

principle of optimality criteria:

optimal solⁿ to a problem contains within it optimal solⁿ to subproblems then the optimization problem is said to satisfy the principle of optimality criteria.