Algo

DP:

Longest common subsequence (LCS)

A = {1, 2, 3, 4}, B = {1, 2, 3}

Sub: {1}, {2}, {3}, {4}
{1,2}, {1,3}, {1,4}, {2,3} ....
{1,2,4}, {1,2,3} .....

maintain order sequence

{1, 4, 3} → नहीं

step:

lcs ({1, 2, 3, 4}, {1, 2, 3})

⇒ 1 + lcs ({2, 3, 4}, {2, 3})

⇒ 2 + lcs ({3, 4}, {3})

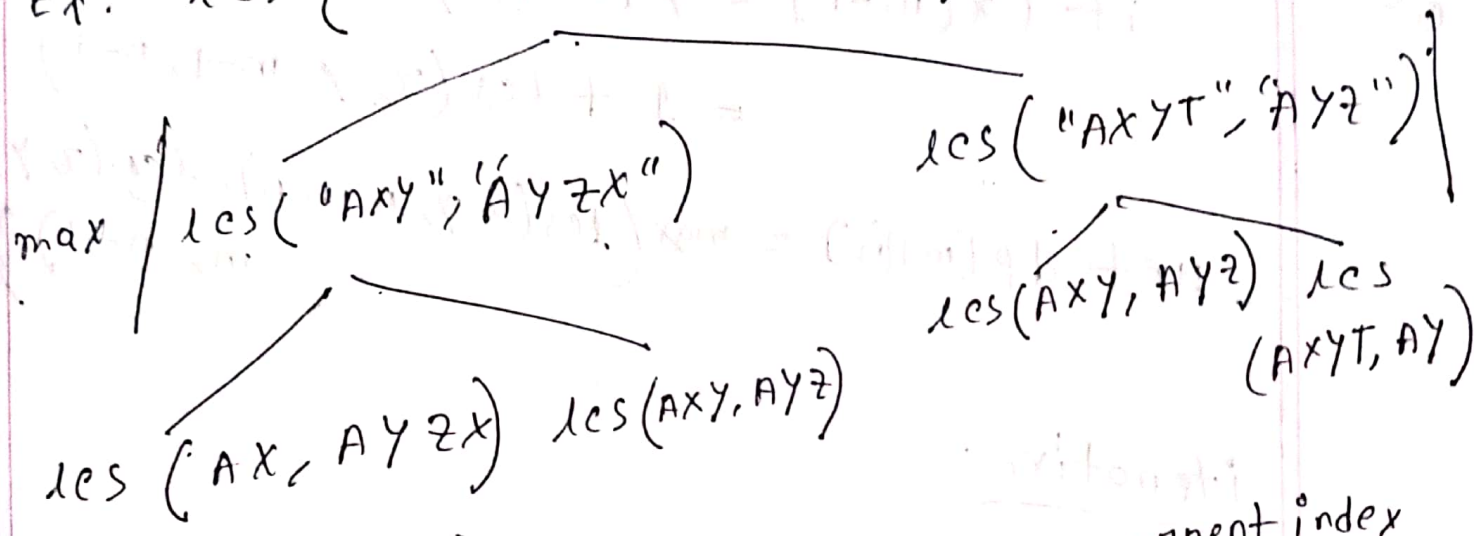
→ Sort
→ Fond
→ DP

Algo TT
8 dec

=> 3 + lcs({4}, {4})

=> 3 + 0 ∴ length = 3

Ex: lcs("AXYT", "AYZX")



lcs(x, y, m, n) { m → x string current index
n → y " "

code:

```
if (m = 0
|| n = 0) return 0;
if (x[m] == y[n]) return 1 + lcs(x, y, m-1, n-1)
return max{lcs(x, y, m-1, n),
           lcs(x, y, m, n-1)}; }
```

Time : O(2^n)

2D vector [m] [n]

~~Top-down~~ Bottom-up (recursive)

improvised code: if (m==0 || n==0) return 0;

if (dp[m][n] != -1)

return dp[m][n];

if (x[m-1] == y[n-1]) return dp[m][n]

= 1 + lcs(x, y, m-1, n-1)

ret dp[m][n] = max(lcs(x, y, m-1, n), lcs(x, y, m, n-1));

Time: O(m*n)

iterative:

	0	0	0	0	0
A	0	0	0	0	1
C	0	0	0	1	2
A	0	1	1	2	2
D	0	1	1	2	2
B	0	1	2	2	2