Problem statement:

Input: Sequences

$$X_m = \langle x_1, x_2, x_3, \dots, x_m \rangle$$

 $Y_n = \langle y_1, y_2, \dots, y_n \rangle$

Output: longest common subsequence (LCS) of X_m and Y_n

Brute-force solution:

lacktriangle For every subsequence of X_m , check if it is a subsequence of Y_n .

▶ Running time: $\Theta(n \cdot 2^m)$

► Intractable!

DP-Step 1: characterize the structure of an optimal solution

Let $Z_k = \langle z_1, z_2, \dots, z_k \rangle$ be any LCS of

$$X_m = \langle x_1, x_2, \dots, \frac{x_m}{n} \rangle$$
 and $Y_n = \langle y_1, \dots, \frac{y_n}{n} \rangle$

Then

- 1. If $x_m = y_n$, then
 - (a) $z_k = x_m = y_n$
 - (b) $Z_{k-1} = LCS(X_{m-1}, Y_{n-1})$
- 2. If $x_m \neq y_n$, then
 - (a) $z_k \neq x_m \Longrightarrow Z_k = \mathsf{LCS}(X_{m-1}, Y_n)$
 - (b) $z_k \neq y_n \Longrightarrow Z_k = \mathsf{LCS}(X_m, Y_{n-1})$

In words, the optimal solution to the (whole) problem contains within it the otpimal solutions to subproblems = the optimal substructure property

Sketch of the proof: by contradiction!

DP-Step 2: recursively define the value of an optimal solution

Define

$$c[i,j] = \text{length of LCS}(X_i, Y_j)$$

By the optimal structure property

$$c[i,j] = \left\{ \begin{array}{ll} 0 & \text{if } i = 0 \text{ or } j = 0 \\ c[i-1,j-1] + 1 & \text{if } x[i] = y[j] \\ \max\{c[i,j-1],c[i-1,j]\} & \text{otherwise} \end{array} \right.$$

Meanwhile, create b[i,j] to record the optimal subproblem solution chosen when computing c[i,j]

• $c[m,n] = \text{length of LCS}(X_m, Y_n)$

DP-Step 3: compute c[i,j] in a bottom-up approach

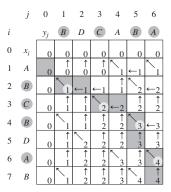
- lacktriangle Compute c[i,j] in a bottom-up approach.
- \blacktriangleright Create b[i,j] to record the optimal subproblem solution chosen when computing c[i,j]
- c[i,j] is the length of $LCS(X_i,Y_j)$ $b[i,j] \text{ shows how to construct the corresponding } LCS(X_i,Y_j)$
- ► Pseudocode, next slide
- ► Cost: Running time: $\Theta(mn)$ Space: $\Theta(mn)$

Pseudocode

```
LCS-length(X,Y)
set c[i,0] = 0 and c[0,j] = 0
for i = 1 to m // Row-major order to compute c and b arrays
   for j = 1 to n
       if X(i) = Y(j)
           c[i,j] = c[i-1,j-1] + 1
          b[i,i] = 'Diag' // go to up diagonal
        elseif c[i-1,j] >= c[i,j-1]
           c[i,j] = c[i-1,j]
          b[i,j] = 'Up' // go up
        else
           c[i,j] = c[i,j-1]
          b[i,j] = 'Left' // go left
        endif
    endfor
endfor
return c and b
```

DP-Step 4: construct an optimal solution from computed information

Example: $X_6 = \langle A, B, C, B, D, A, B \rangle$ and $Y_6 = \langle B, D, C, A, B, A \rangle$



- (1) Length of LCS = c[7,6] = 4
- (2) By the b-table (" \uparrow , \leftarrow , \nwarrow "), the LCS is BCBA