L9

-①: Structure of the optimal solution:

$X = [x_1, x_2, \ldots, x_m]$ $Y = [y_1, y_2, \ldots, y_n]$

Suppose $Z = [z_1, z_2, \ldots, z_k] = LCS(X, Y)$.

Case 1: $x_m = y_n$

Then $z_k = x_m = y_n$ and

$[z_1, z_2, \ldots, z_{k-1}] = LCS([x_1, \ldots, x_{m-1}], [y_1, \ldots, y_{n-1}])$

Case 2: $x_m \neq y_n$

Then $z_k \neq x_m$ or $z_k \neq y_n$ (or both)

Case 2a: $x_m \neq y_n$ and $z_k \neq x_m$

Then $Z = LCS(X[1..m-1], Y[1..n])$

Case 2b: $x_m \neq y_n$ and $z_k \neq y_n$

Then $Z = LCS(X[1..m], Y[1..n-1])$

But we don't know Z, so we don't know which case we're in.

$\Rightarrow$ Pick the best of the two. (try both!)

② Recurrence:

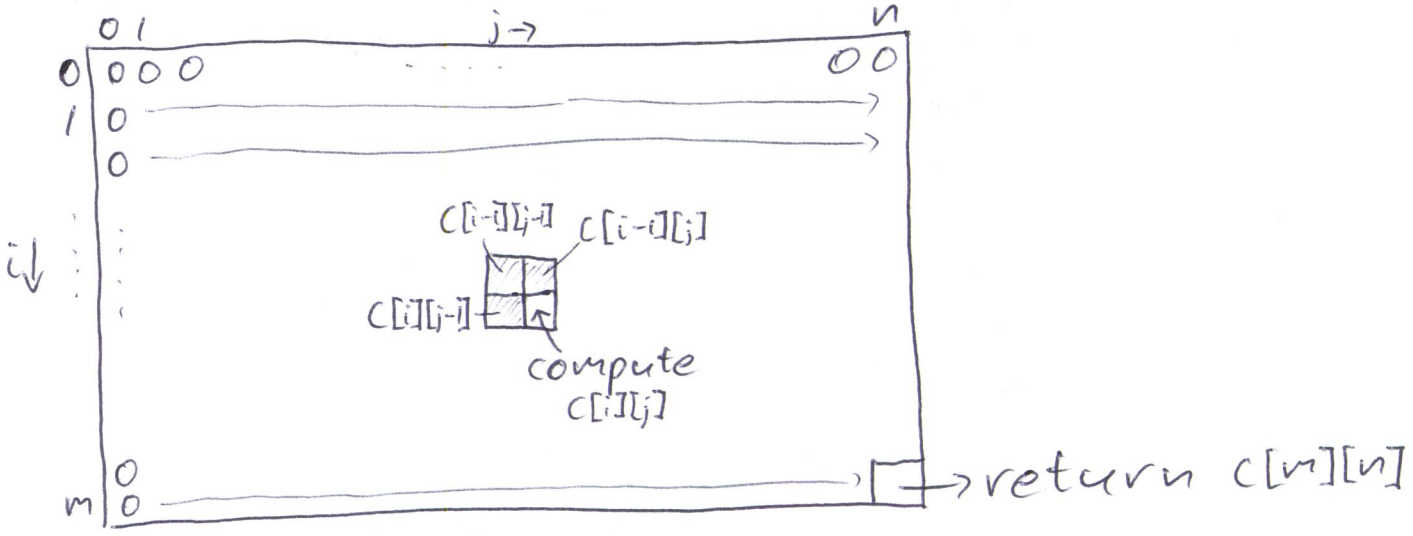
Define $c(i,j)$ = length of $LCS(X[1..i], Y[1..j])$

We want to compute $c(m,n)$.

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

③ Solve bottom-up:

Fill in $m \times n$ matrix $c[i][j]$:



Algorithm LCS(X, Y):

```
for i ← 0 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 X[i] = Y[j] then c[i,j] ← 1 + c[i-1,j-1]
    else c[i,j] ← max(c[i-1,j], c[i,j-1])
return c[m,n]
```

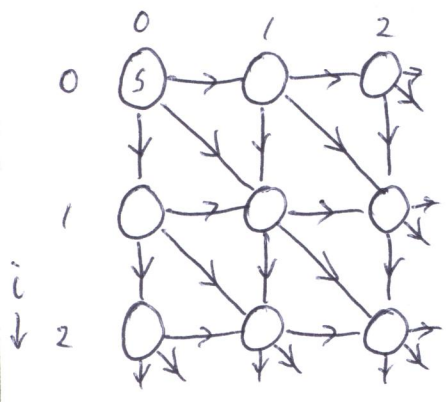
(Example!)

- Correct? Yes, by steps ① and ②.
- Terminates! Yes
- Efficient? $O(mn)$ time and space.

  Can you think of a way to use less space?

Reduction to paths in DAGs

Consider this weighted DAG G:



- Diagonal edge to node (i,j) has weight 1 if $X[i] = Y[j]$
- All other edges have weight 0.

The longest s-t path in G corresponds to the longest common subsequence of X and Y.

We can find the longest path with a small modification of the shortest path algorithm.

Thus, we can solve longest common subsequence by ~~finding~~ solving longest path in a specific DAG.

$\Rightarrow$ We have <u>reduced</u> LCS to longest path.