Dynamic Programming: Example 2
Longest Common Subsequence

Problem: Let $x_1x_2...x_m$ and $y_1y_2...y_n$ be two sequences over some alphabet. (We assume they are strings of characters.) Find a longest common subsequence (LCS) of $x_1x_2...x_m$ and $y_1y_2...y_n$.

Example: $x_1x_2x_3x_4x_5x_6x_7x_8$ = **b a c b f f c b**
$y_1y_2y_3y_4y_5y_6y_7y_8y_9$ = **d a b e a b f b c**
$z_1z_2z_3z_4z_5$ = **b a b f c** is an LCS (shown below).

Subproblems: Find an LCS of $x_1x_2...x_i$ and $y_1y_2...y_j$ ($0 \le i \le m$, $0 \le j \le m$).

Optimal substructure: If $z = z_1z_2...z_p$ is a LCS of $x_1x_2...x_m$ and $y_1y_2...y_n$, then at least one of these most hold.

i) $x_m = y_n$, and $z_1z_2...z_{p-1}$ is an LCS of $x_1x_2...x_{m-1}$ and $y_1y_2...y_{n-1}$,
ii) $x_m \ne y_n$, and $z_1z_2...z_p$ is an LCS of $x_1x_2...x_{m-1}$ and $y_1y_2...y_n$,
iii) $x_m \ne y_n$, and $z_1z_2...z_p$ is an LCS of $x_1x_2...x_m$ and $y_1y_2...y_{n-1}$.

Let c_{ij} = length of LCS of $x_1x_2...x_i$ and $y = y_1y_2...y_j$.

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

$$b[i,j] = \begin{cases} \text{"↖"} & \text{if } x_i = y_j, \\ \text{"↑"} & \text{if } x_i \ne y_j \text{ and } c[i-1,j] \ge c[i,j-1], \\ \text{"←"} & \text{if } x_i \ne y_j \text{ and } c[i-1,j] < c[i,j-1]. \end{cases}$$

We compute the $c[i,j]$ and $b[i,j]$ in order of increasing $i+j$, or alternatively in order of increasing i, and for a fixed i, in order of increasing j.

Example: $x_1x_2x_3x_4x_5x_6x_7x_8$ = **b a c b f f c b**
$y_1y_2y_3y_4y_5y_6y_7y_8y_9$ = **d a b e a b f b c**

Row i, column j of the table below contains the value of $c[i,j]$ followed (except when $i = 0$ or $j = 0$) by that of $b[i,j]$

	0	**1 d**	**2 a**	**3 b**	**4 e**	**5 a**	**6 b**	**7 f**	**8 b**	**9 c**
0	0	0	0	0	0	0	0	0	0	0
1 b	0	0 ↑	0 ↑	1 ↖	1 ←	1 ↑	1 ↖	1 ←	1 ↖	1 ←
2 a	0	0 ↑	1 ↖	1 ↑	1 ↑	2 ↖	2 ←	2 ←	2 ←	2 ←
3 c	0	0 ↑	1 ↑	1 ↑	1 ↑	2 ↑	2 ↑	2 ↑	2 ↑	3 ↖
4 b	0	0 ↑	1 ↑	2 ↖	2 ←	2 ↑	3 ↖	3 ←	3 ↖	3 ↑
5 f	0	0 ↑	1 ↑	2 ↑	2 ↑	2 ↑	3 ↑	4 ↖	4 ←	4 ←
6 f	0	0 ↑	1 ↑	2 ↑	2 ↑	2 ↑	3 ↑	4 ↖	4 ↑	4 ↑
7 c	0	0 ↑	1 ↑	2 ↑	2 ↑	2 ↑	3 ↑	4 ↑	4 ↑	5 ↖
8 b	0	0 ↑	1 ↑	2 ↖	2 ↑	2 ↑	3 ↖	4 ↑	5 ↖	5 ↑

We can compute each table element in constant time, so the entire table takes $\Theta(mn)$ time.

We can write down an LCS by starting in the lower right corner and following the arrows backward.

Whenever we reach a square containing a "↖", say in row i and column j, we insert the character $x_i = y_j$ at the beginning of the subsequence.

	0	**1 d**	**2 a**	**3 b**	**4 e**	**5 a**	**6 b**	**7 f**	**8 b**	**9 c**
0	0	0	**0**	0	0	0	0	0	0	0
1 b	0	0 ↑	0 ↑	**1 ↖**	**1 ←**	1 ↑	1 ↖	1 ←	1 ↖	1 ←
2 a	0	0 ↑	1 ↖	1 ↑	1 ↑	**2 ↖**	2 ←	2 ←	2 ←	2 ←
3 c	0	0 ↑	1 ↑	1 ↑	1 ↑	**2 ↑**	2 ↑	2 ↑	2 ↑	3 ↖
4 b	0	0 ↑	1 ↑	2 ↖	2 ←	2 ↑	**3 ↖**	3 ←	3 ↖	3 ↑
5 f	0	0 ↑	1 ↑	2 ↑	2 ↑	2 ↑	3 ↑	**4 ↖**	**4 ←**	4 ←
6 f	0	0 ↑	1 ↑	2 ↑	2 ↑	2 ↑	3 ↑	4 ↖	**4 ↑**	4 ↑
7 c	0	0 ↑	1 ↑	2 ↑	2 ↑	2 ↑	3 ↑	4 ↑	4 ↑	**5 ↖**
8 b	0	0 ↑	1 ↑	2 ↖	2 ↑	2 ↑	3 ↖	4 ↑	5 ↖	**5 ↑**

An LCS is $x_1x_2x_4x_5x_7 = y_3y_5y_6y_7y_9 =$ **b a b f c**.

This computation takes $\Theta(m+n)$ time.