Let’s look at another dynamic programming example:

**Longest Common Subsequence (LCS):**

Let $X = \langle x_1, x_2, \ldots, x_m \rangle$ be a sequence. Then, another sequence $Z = \langle z_1, \ldots, z_k \rangle$ is a subsequence of $X$ if there exists a strictly increasing sequence of indices $i_1, \ldots, i_k$ such that $z_j = x_{i_j}$ for all $1 \leq j \leq k$.

Given two sequences $X$ and $Y$, a third sequence $Z$ is a common subsequence of both $X$ and $Y$ if it is a subsequence of $X$ and a subsequence of $Y$. 
Consider the sequence \( X = \langle A, B, C, B, D, A, B \rangle \). Then, the sequence \( Z = \langle B, B, A, B \rangle \) is a subsequence of \( X \).

Similarly, let

\[
X = \langle A, B, C, B, D, A, B, C, D \rangle
\]

and

\[
\]

Then, \( Z = \langle A, C, D \rangle \) is a common subsequence of \( X \) and \( Y \).

What is the longest common subsequence of \( X \) and \( Y \)?
Step (i): Optimal Substructure

Let $X = \langle x_1, x_2, \ldots, x_m \rangle$ and $Y = \langle y_1, y_2, \ldots, y_n \rangle$ be two sequences, and let $Z = \langle z_1, z_2, \ldots, z_k \rangle$ be a LCS of $X$ and $Y$. Then:

- if $x_m = y_n$, then $z_k = x_m = y_n$ and $Z_{k-1}$ is a LCS of $X_{m-1}$ and $Y_{n-1}$ ($z_k$ has to be equal to $x_m/y_n$, otherwise $Z$ won’t be a LCS).

- if $x_m \neq y_n$, then:
  - $z_k \neq x_m \Rightarrow Z$ is an LCS of $X_{m-1}$ and $Y$.
  - $z_k \neq y_n \Rightarrow Z$ is an LCS of $X$ and $Y_{n-1}$.
Step (ii): A recursive solution

Definition: Let $c[i, j]$ be the length of the longest common subsequence between $X_i = \langle x_1, \ldots, x_i \rangle$ and $Y_j = \langle y_1, \ldots, y_j \rangle$.

Then $c[n, m]$ contains the length of an LCS of $X$ and $Y$, and:

$$c[i, j] = \begin{cases} 
0 & \text{if } i = 0 \text{ or } j = 0 \\
2 \cdot c[i - 1, j - 1] + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j \\
\max(c[i - 1, j], c[i, j - 1]) & \text{otherwise.}
\end{cases}$$
Step (iii): Bottom-up iterative computation

LCS(X,Y,m,n) /*X has m elements, Y has n elements */
for i = 1 to m c[i,0] := 0;
for j = 1 to n c[0,j] := 0;
for i = 1 to m
    for j = 1 to n
        if X[i] == Y[j]
            c[i,j] := c[i-1,j-1]+1;
            b[i,j] := "\";
        else
            if c[i-1,j] ≥ c[i,j-1]
                c[i,j] := c[i-1,j];
                b[i,j] := "↑";
            else
                c[i,j] := c[i,j-1];
                b[i,j] := "←";
Step (iv): Figuring out the LCS

Use a recursive algorithm: \( b[i, j] \) points to the table entry corresponding to the optimal subproblem solution chosen when computing \( c[i, j] \).

Print-LCS\((b, X, i, j)\)

\[
\begin{align*}
& \text{if } i = 0 \text{ return; } \\
& \text{if } j = 0 \text{ return; } \\
& \text{if } b[i, j] = "\downarrow" \\
& \quad \text{Print-LCS}\((b, X, i - 1, j - 1)\); \\
& \quad \text{print } X[i]; \\
& \text{else if } b[i, j] = "\uparrow" \\
& \quad \text{Print-LCS}\((b, X, i - 1, j)\); \\
& \text{else} \\
& \quad \text{Print-LCS}\((b, X, i, j - 1)\); \\
\end{align*}
\]
Consider the following example:

\[ X = \langle A, B, C, B, D, A, B \rangle \]

and

\[ Y = \langle B, D, C, A, B, A \rangle. \]

Let’s compute \( c \) and \( b \) on the board. Then, we’ll compute the LCS.