14.2.6
Longest Common Subsequence Problem
LCS Problem

Definition 14.7.
LCS between two strings $X$ and $Y$ is the length of longest common subsequence between $X$ and $Y$.

Example 14.8.
LCS between ABAZDC and BACBAD is 4 via ABAD

Derive a dynamic programming algorithm for the problem.
Definition 14.7.
LCS between two strings $X$ and $Y$ is the length of longest common subsequence between $X$ and $Y$.

Example 14.8.
LCS between ABAZDC and BACBAD is 4 via ABAD

Derive a dynamic programming algorithm for the problem.
LCS Problem

Definition 14.7.
LCS between two strings $X$ and $Y$ is the length of longest common subsequence between $X$ and $Y$.

\[
\begin{array}{c}
ABAZDC \\
BACBAD
\end{array}
\quad
\begin{array}{c}
ABAZDC \\
BACBAD
\end{array}
\]

Example 14.8.
LCS between ABAZDC and BACBAD is 4 via ABAD

Derive a dynamic programming algorithm for the problem.
LCS recursive definition

$A[1..n], B[1..m]$: Input strings.

$$LCS(i, j) = \begin{cases} 
0 & \text{ if } i = 0 \text{ or } j = 0 \\
\max \left( \begin{array}{l} 
LCS(i - 1, j), \\
LCS(i, j - 1), \\
1 + LCS(i - 1, j - 1)
\end{array} \right) & \text{ otherwise }
\end{cases}$$

Similar to edit distance... $O(nm)$ time algorithm $O(m)$ space.
LCS recursive definition

\( A[1..n], B[1..m] \): Input strings.

\[
LCS(i, j) = \begin{cases} 
0 & \text{if } i = 0 \text{ or } j = 0 \\
\max \left( LCS(i-1, j), LCS(i, j-1) \right) & \text{if } A[i] \neq B[j] \\
\max \left( LCS(i-1, j), LCS(i, j-1) \right) + 1 & \text{if } A[i] = B[j]
\end{cases}
\]

Similar to edit distance... \( O(nm) \) time algorithm \( O(m) \) space.
Longest common subsequence is just edit distance for the two sequences...

\( A, B \): input sequences
\( \Sigma \): “alphabet” all the different values in \( A \) and \( B \)

\[ \forall b, c \in \Sigma : b \neq c \quad \text{\( COST[b][c] = +\infty. \) } \]
\[ \forall b \in \Sigma \quad \text{\( COST[b][b] = 1 \) } \]

\( 1 \): price of deletion of insertion of a single character

Length of longest common subsequence = \( m + n - \text{ed}(A, B) \)
Longest common subsequence is just edit distance for the two sequences...

\( A, B \): input sequences
\( \Sigma \): “alphabet” all the different values in \( A \) and \( B \)

\[
\forall b, c \in \Sigma : b \neq c \quad \text{COST}[b][c] = +\infty.
\]

\[
\forall b \in \Sigma \quad \text{COST}[b][b] = 1
\]

\( 1 \): price of deletion of insertion of a single character

Length of longest common subsequence = \( m + n - \text{ed}(A, B) \)
THE END

... (for now)