**Longest Common Subsequence**

A java string is essentially a sequence of characters. For example, the java string constant "AFRICA" is a character sequence of length 6. Like arrays, characters in a java string are indexed starting from 0. For a string $X$, we will write $X[i..j]$ to mean the substring of $X$ consisting of all characters from position $i$ to position $j - 1$ inclusive. For example, if $X$ is "AFRICA", then $X[1..4]$ is "FRI" and $X[0..6]$ is "AFRICA" itself.

If one deletes characters at certain positions from a given string $X$, what remains is called a subsequence of $X$. For example, deleting the characters at positions 1 and 4 from "AFRICA" leaves us with the string "ARIA". So we may say that "ARIA" is a subsequence of "AFRICA". Deleting no characters is also permitted. So, for instance, "AFRICA" is considered a subsequence of itself.

A sequence $Z$ is a common subsequence of sequences $X$ and $Y$ if $Z$ is a subsequence of both $X$ and $Y$. For instance, "DIN" is a common subsequence of "DYNAMICPROGRAMMING" and "DIVIDEANDCONQUER".

A longest common subsequence (LCS) of sequences $X$ and $Y$ is a common subsequence of $X$ and $Y$ of maximum possible length. For instance, "DYNAMICPROGRAMMING" and "DIVIDEANDCONQUER" have "DICOR" as an LCS. This is because "DICOR" is their common subsequence and no common subsequence of length 6 exists. "DICON" is another LCS, so LCS's are not unique.

**Problem**

Let two sequences $X = X_0X_1\ldots X_{m-1}$ and $Y = Y_0Y_1\ldots Y_{n-1}$ be given. We want to find an LCS of $X$ and $Y$.

**Dynamic Programming Solution**

For $0 \leq i < m$ and $0 \leq j < n$, let $\text{opt}(i, j)$ be the length of an LCS of $X[i..m)$ and $Y[j..n)$.
We seek $\text{opt}(0, 0)$. 
Optimal Substructure Property
Suppose $Z = Z[0..k]$ is an LCS of $X[i..m]$ and $Y[j..n]$.
If $X_i = Y_j$, then necessarily $X_i = Z_0$. We can then show that $Z[1..k]$ is an LCS of $X[i+1..m]$ and $Y[j+1..n]$.
If $X_i \neq Y_j$, then $X_i \neq Z_0$ or $Y_j \neq Z_0$. If $X_i \neq Z_0$, we can show that $Z$ is an LCS of $X[i+1..m]$ and $Y[j..n]$. If $Y_j \neq Z_0$, we can show that $Z$ is an LCS of $X[i..m]$ and $Y[j+1..n]$.
We know that one of the above cases must occur. This gives us the following recurrence.

Recurrence

$$
\text{opt}(i, j) = \begin{cases} 
0 & \text{if } i = m \text{ or } j = n \\
\text{opt}(i + 1, j + 1) + 1 & \text{if } 0 \leq i < m, \text{ and } 0 \leq j < n, \text{ and } X_i = Y_j \\
\max\{ \text{opt}(i, j + 1), \text{opt}(i + 1, j) \} & \text{if } 0 \leq i < m, \text{ and } 0 \leq j < n, \text{ and } X_i \neq Y_j
\end{cases}
$$

Example dynamic programming table

<table>
<thead>
<tr>
<th></th>
<th>s</th>
<th>a</th>
<th>i</th>
<th>n</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>t</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>n</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Longest Common Subsequence Algorithm

**Step 1.** Fill in a table of $\text{opt}(\cdot, \cdot)$ values, plus a companion table of maximizers. We can fill in the table row-by-row, column-by-column, or diagonal-by-diagonal.

**Step 2.** Find the LCS by following maximizer pointers, starting from $\text{opt}(0, 0)$. 