Longest Common Subsequence

Input: Two sequences (strings etc)

Output: Longest common subsequence.

Examples of common subsequences:

- exercise
- determine

- morning
- triangle

- toward
- thousand

- eerie
- ring
- toad
LCS: Recursive Solution

Let $LCS(n,m)$ be the length of the longest common subsequence of $X[1 \ldots n]$ and $Y[1 \ldots m]$.

```plaintext
if ($\odot == n$ or $\odot == m$) return 0
if ($X[n] == Y[m]$)
    return 1 + LCS($n-1, m-1$)
return max (LCS($n, m-1$), LCS($n-1, m$))
```
LCS: Iterative Solution

\[ \text{LCS}(n,m): \quad // \text{recursive for reference} \]
\[
\text{if } 0 == n \text{ or } 0 == m \text{ then} \\
\quad \text{return } 0 \\
\text{else if } X[n] = Y[m] \text{ then} \\
\quad \text{return } 1 + \text{LCS}(n - 1, m - 1) \\
\text{else} \\
\quad \text{return } \max(\text{LCS}(n - 1, m), \text{LCS}(n, m - 1))
\]

\[
\text{declare } \text{LCS}[0..n, 0..m] \\
\text{//non-base:} \\
\text{for } i = 1 \ldots n \\
\quad \text{for } j = 1 \ldots m \\
\quad \text{// fill in LCS[i,j] as per above}
\]

\[
\text{for } (j=0 \ldots m) \\
\quad \text{LCS}[0, j] = 0 \\
\text{for } i=1 \ldots n \\
\quad \text{LCS}[i, 0] = 0
\]
LCS Example:

<table>
<thead>
<tr>
<th></th>
<th>M</th>
<th>O</th>
<th>R</th>
<th>N</th>
<th>I</th>
<th>N</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>R</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>N</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>L</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Finding the LCS itself

We have LCS[] filled in.

\[ i \leftarrow n, j \leftarrow m, S \leftarrow \text{empty stack} \]

\[ \text{while do} \]

// is \[ x[i] \] and \[ y[j] \] part of output?

\[ \text{good reinforcement} \]
Did we do everything we need to do?

- Describe *in English* the function
  - Not *how* it works (yet)
  - Yes *what it solves.*
  - Skipping this step = 0 on problem

- Give that function a meaningful variable name.
  - Not “OPT” or “DP” or “table”
  - Not a single letter either.

- Give recursive formulation.

- Describe the iterative running time.
  - Was cut-off in Friday’s slides. 😞