- Longest Common Subsequence (LCS)

\[ a[7] = \text{ababced} \]
\[ b[7] = \text{abbece} \]

For last character of \( a[7] \) and \( b[7] \), we need to decide whether they belong to the LCS.

<table>
<thead>
<tr>
<th>in</th>
<th>out</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Case 1: neither are in LCS

\[ LCS = LCS('ababced', 'abbece') \]

Case 2: last char of \( a[7] \) is in LCS, last char of \( b[7] \) is not.

\[ LCS('ababcede', 'abbece') \]

Not precise, letter 'e' in \( a[7] \) may not be in LCS.

Attempt 2: case 1, last char of \( a[7] \) is not in LCS

\[ LCS = LCS('ababced', 'abbece') \]

Case 2: last char of \( b[7] \) is not in LCS

\[ LCS = LCS('ababcede', 'abbece') \]

Case 3: last chars of both \( a[7] \) and \( b[7] \) are in LCS only valid if the last chars are equal.

\[ a[7] = \text{ababced}, b[7] = \text{abbece} \]

\[ LCS = LCS('ababc', 'abbece') + 'd' \]
state: let \( f[i,j] \) be the length of LCS for \( a[1...i], b[1...j] \)

transition:
\[
f[i,j] = \max \left\{ \begin{array}{ll}
f[i-1,j] \\
f[i,j-1] \\
f[i-1,j-1] + 1 \text{ if } a[i]=b[j]
\end{array} \right. 
\]

base case:
\[
i = 0 \quad f[0,j] = 0 \quad \text{for every } j
\]
\[
j = 0 \quad f[i,0] = 0 \quad \text{for every } i
\]

running time:
\[
\# \text{ states} \times \text{time for evaluating transition func.} = 0(1)
\]
\[
\text{length of } \frac{a}{b} = 0(nm)
\]

- "Voice recognition"

input: \( n \) # of sound segments
\( k \) # of phonemes

\( a[1...n, 1...k] \) \( a[i,j] \): score of assigning phoneme \( j \)
to sound segment \( i \)

\( b[1...k, 1...k] \) \( b[i,j] \): score of phoneme \( j \) appear immediately after phoneme \( i \)

solution: sequence \( v[1...n] \), \( v[i] \in \{1, 2, ..., k\} \)

\( v[i] \): phoneme assigned to sound segment \( i \)

\[
\text{Score} = \sum_{i=1}^{n} a[i, v[i]] + \sum_{i=1}^{n-1} b[v[i], v[i+1]]
\]

Score from assigning phonemes to sound segments
Score from how likely phonemes appear together

goal: maximize the score
state: \( f[i, j] \): max score for first \( i \) sound segments, sound segment \( i \) is phoneme \( j \)

transition function: if sound segment \( i \) is phoneme \( p \)
then \[ f[i, j] = \max_{p=1,\ldots,k} \left( f[i-1, p] + b[p, j] + a[i, j] \right) \]

\[ f[i, j] = \max \left( f[i-1, p] + b[p, j] \right) + a[i, j] \]

base case: \( f[1, j] = a[1, j] \)

\( f[1, j] = a[1, j] \) for all \( j \)

for \( i = 2 \) to \( n \)
for \( j = 1 \) to \( k \)
evaluate transition function \( f[i, j] \)
return \( \max_{i=1,\ldots,k} f[n, j] \)

running time: \( \# \) states = \( n \times k \)
time to evaluate transition function: \( O(k) \)

\( n \times k \times O(k) = O(nk^2) \)