
Only do the first two bullets. Figure 1 contains a \texttt{T}EX template to fill in twice, once for the first bullet and once for the second bullet.

We follow the notation style used in class.

**First bullet — optimal global alignment.** The recurrence for these scoring values is as follows. The base cases are

\[ s(\lambda, W_j) = -j, \]

for \( 0 \leq j \leq m \), and

\[ s(V_i, \lambda) = -i, \]

for \( 0 \leq i \leq n \). The general case is

\[
\begin{align*}
  s(V_i, W_j) &= \max \left\{ \begin{array}{ll}
    s(V_{i-1}, W_{j-1}) + 1 & \text{if } v_i = w_j; \\
    s(V_{i-1}, W_{j-1}) - 1 & \text{if } v_i \neq w_j; \\
    s(V_{i-1}, W_j) - 1; & \\
    s(V_i, W_{j-1}) - 1,
  \end{array} \right. \\
\end{align*}
\]

for \( 1 \leq i \leq n \) and \( 1 \leq j \leq m \).

The dynamic programming table is in Figure 2.

The optimal score is \( s(V_9, W_9) = -1 \).

There are many backtracking paths to follow. One of them is shown in red. It corresponds to the following optimal global alignment

\[
\begin{align*}
  &T \quad A \quad C \quad G \quad G \quad G \quad T \quad A \quad T \\
  &G \quad G \quad A \quad C \quad - \quad G \quad - \quad T \quad A \quad C \quad G
\end{align*}
\]

By computation, the score is

\[-1 - 1 + 1 + 1 - 1 + 1 + 1 - 1 - 1 - 1 = -1,\]

as expected.
Figure 1: \LaTeX{} template for dynamic programming in second problem.
Figure 2: Dynamic programming table for optimal global alignment.
Second bullet — optimal local alignment. The recurrence for these scoring values is as follows. The base cases are

\[ s(\lambda, W_j) = 0, \]

for \(0 \leq j \leq m\), and

\[ s(V_i, \lambda) = 0, \]

for \(0 \leq i \leq n\). The general case is

\[ s(V_i, W_j) = \max \begin{cases} s(V_{i-1}, W_{j-1}) + 1 & \text{if } v_i = w_j; \\ s(V_{i-1}, W_{j-1}) - 1 & \text{if } v_i \neq w_j; \\ s(V_{i-1}, W_j) - 1; \\ s(V_i, W_{j-1}) - 1; \\ 0, \end{cases} \]

for \(1 \leq i \leq n\) and \(1 \leq j \leq m\).

The dynamic programming table is in Figure 3.

The optimal score is \(s(V_4, W_9) = 4\), the largest score in the table.

There is a unique backtracking path to follow. It is shown in red. It corresponds to the following optimal local alignment

\[
\begin{array}{c}
T \\
A \\
C \\
G
\end{array}
\quad \begin{array}{c}
T \\
A \\
C \\
G
\end{array}
\]

By computation, the score is

\[+1 + 1 + 1 + 1 = 4,\]

as expected.
Figure 3: Dynamic programming table for optimal local alignment.

Follow the dynamic programming paradigm. Give pseudocode for the resulting algorithm.

The shortest supersequence of strings $V$ and $W$ corresponds to a shortest alignment of $V$ and $W$ in which no mismatches are allowed. The subproblems to solve are the usual:

$$ V_i = v_1 v_2 \cdots v_i, $$
for $0 \leq i \leq n$, and

$$ W_j = w_1 w_2 \cdots w_j, $$
for $0 \leq j \leq m$. Let $s(V_i, W_j)$ be the length of a shortest common supersequence of $V_i$ and $W_j$. The base cases are

$$ s(\lambda, W_j) = j, $$
for $0 \leq j \leq m$, and

$$ s(V_i, \lambda) = i, $$
for $0 \leq i \leq n$. The general case is, for $1 \leq i \leq n$ and $1 \leq j \leq m$,

$$ s(V_i, W_j) = \min \left\{ \begin{array}{ll}
    s(V_{i-1}, W_{j-1}) + 1 & \text{if } v_i = w_j; \\
    s(V_{i-1}, W_{j-1}) + 2 & \text{if } v_i \neq w_j; \\
    s(V_{i-1}, W_j) + 1 & \\
    s(V_i, W_{j-1}) + 1.
\end{array} \right. $$

Translated into pseudocode, the required algorithm is in Figure 4.
ShortestSupersequence\((V, W)\)

\begin{array}{l}
\triangleright V = v_1v_2\cdots v_n \\
\triangleright W = w_1w_2\cdots w_m \\
\triangleright \text{Base cases} \\
\text{for } j \leftarrow 0 \text{ to } m \\
\quad \text{do } s(\lambda, W_j) \leftarrow j \\
\text{for } i \leftarrow 0 \text{ to } n \\
\quad \text{do } s(V_i, \lambda) \leftarrow i \\
\triangleright \text{General case} \\
\text{for } i \leftarrow 1 \text{ to } n \\
\quad \text{do for } j \leftarrow 1 \text{ to } m \\
\quad \quad \text{do if } v_i = w_j \\
\quad \quad \quad \text{then } s(V_i, W_j) \leftarrow s(V_{i-1}, W_{j-1}) + 1 \\
\quad \quad \quad \text{else } s(V_i, W_j) \leftarrow s(V_{i-1}, W_{j-1}) + 2 \\
\quad \quad \quad \text{if } s(V_i, W_j) > s(V_{i-1}, W_j) + 1 \\
\quad \quad \quad \quad \text{then } s(V_i, W_j) \leftarrow s(V_{i-1}, W_j) + 1 \\
\quad \quad \quad \text{if } s(V_i, W_j) > s(V_i, W_{j-1}) + 1 \\
\quad \quad \quad \quad \text{then } s(V_i, W_j) \leftarrow s(V_i, W_{j-1}) + 1 \\
\text{return } s(V_n, W_m)
\end{array}

Figure 4: Pseudocode for the shortest supersequence algorithm.