Sequence similarity

Why are we studying sequence similarity?

A high degree of similarity between two sequences usually implies similar function and/or structure.

Evolution at molecular level.

- Chimpanzee and man had a recent common ancestor
- Wings of birds and those of bats were evolved independently.
- How do we distinguish common ancestry and common appearances?
- Since genomic DNA is the blueprint for a living organism, evolution must be directly related to changes in DNA.
Other motivations

Hypothesizing the function of a new sequence.

In 1995, Fleischmann et al. were the first to succeed in sequencing the entire genome of a free-living organism.

They identified 1743 regions of this sequence likely to be sites of genes and set out to discover functions of those genes.

They translated the coding region into their corresponding amino acid sequence and searched for similar sequence in a protein database.

This search identified 1007 close matches.

Because the database also contains the function of the sequences, these close matches allowed them to make strong conjectures about the function of each matched gene.

These conjectures can then be tested in the laboratory.
Haemophilus influenzae

Haemophilus influenzae is a small, nonmotile, Gram-negative bacterium whose only natural host is human. Six H. influenzae serotype strains (a through f) have been identified on the basis of immunologically distinct capsular polysaccharide antigens. Non-typeable strains also exist and are distinguished by their lack of detectable capsular polysaccharide. They are commensal residents of the upper respiratory mucosa of children and adults and cause otitis media and respiratory infections, mostly in children. More serious invasive infection is caused almost exclusively by type b strains, with meningitis producing neurological sequelae in up to 50 percent of affected children. The complete genomic sequence is from the H. influenzae Rd strain KW20 (K. W. Wilcox and H. O. Smith, J. Bacteriol. 122, 443 (1975). It is significant in that it is the strain from which the first DNA restriction enzymes were purified. The genome is 1.83 Mb in size and contains approximately 1740 open reading frames.

Research the effects of multiple sclerosis.

Multiple sclerosis is an autoimmune disease in which the immune system attacks nerve cells in the patient.

More specifically, the immune system’s T-cells, which normally identify foreign bodies for immune system attacks, mistakenly identify proteins in the nerves’ myelin sheaths as foreign.

It was conjectured that the myelin sheath proteins identified by the T-cells were similar to viral and/or bacterial sheath proteins from an earlier infection.
In order to test this hypothesis, the following steps were carried out:

- the myelin sheath proteins were sequenced,
- a protein database was searched for similar bacterial and viral sequences, and
- laboratory tests were performed to determine if the T-cells attacked same proteins.

The result was the identification of certain bacterial and viral proteins that were confused with the myelin sheath proteins.

Hopefully further study of these confused proteins would identify the common features that caused the T-cells to misidentify the myelin sheath proteins.
The string alignment problem

Given two strings, how should we measure similarity between them?

Similarity is witnessed by finding a good “alignment” between two strings.

If \( x \) and \( y \) are each a single character or space, then \( \sigma(x, y) \) denotes the score of aligning \( x \) and \( y \).

If \( S \) is a string, then \( |S| \) denotes the length of \( S \) and \( S[i] \) denotes the \( i \)th character of \( S \).

Let \( S \) and \( T \) be strings of length \( n \) and \( m \) respectively. An alignment \( A \) maps \( S \) and \( T \) into strings \( S' \) and \( T' \) that may contain space characters, where

1. \( |S'| = |T'| \), and
2. the removal of all spaces from \( S' \) and \( T' \) leaves \( S \) and \( T \), respectively.
The value of the alignment $A$ is

$$
\sum_{i=1}^{L} \sigma(S'[i], T'[i]),
$$

where $L = |S'| = |T'|$.

An optimal alignment of $S$ and $T$ is one that has the maximum possible value. Sometimes, in stead of similarity, we want to find the distance between two strings.
When counting alignments, space aligned to space is not allowed since there is no point in matching two spaces.

It is clear that \( \max\{n, m\} \leq L \leq n + m \).

Let \( f(i, j) \) be the number of alignments of one sequence of \( i \) letters with another of \( j \) letters.

\[
f(n, m) = f(n - 1, m) + f(n, m - 1) + f(n - 1, m - 1).
\]

\[
f(n, n) \approx (1 + \sqrt{2})^{2n + 1} n^{-1/2}.
\]
When only the matches and not the details of indels are counted, the situation improves slightly.

Let $g(i, j)$ be this smaller number of alignments.

$$g(n, m) = 1 + \sum_{1 \leq i \leq n; 1 \leq j \leq m} g(i - 1, j - 1)$$
$$= g(n - 1, m) + g(n, m - 1) - g(n - 1, m - 1) + g(n - 1, m - 1)$$
$$= g(n - 1, m) + g(n, m - 1) = C_{n+m}^m.$$  
$$g(n, n) = C_{2n}^n \approx 2^{2n} (\sqrt{n\pi})^{-1}.$$  

Another way to derive this formula is the following.

There must be $k$ aligned pairs, $0 \leq k \leq \min\{n, m\}$. There are $C_n^k$ ways to choose $k$ letters from $S$ and there are $C_m^k$ ways to choose $k$ letters from $T$.

Therefore

$$g(n, m) = \sum_{k=0}^{\min\{n, m\}} C_n^k C_m^k = C_{n+m}^m.$$  

When $n = m = 1000$, $g(1000, 1000) \approx 10^{600}$ which means that a direct search is impossible.
Global distance alignment

Let $A = a_1a_2 \cdots a_n$ and $B = b_1b_2 \cdots b_m$.

Let $\delta$ be the cost for an insertion or a deletion and $d(a_i, b_j)$ be the cost of changing $a_i$ to $b_j$.

Set $D(i, j) = D(a_1 \cdots a_i, b_1 \cdots b_j)$, $D(0, 0) = 0$, $D(0, j) = j\delta$, and $D(i, 0) = i\delta$. Then

$$D(i, j) = \min \left\{ \begin{array}{l}
D(i-1, j-1) + d(a_i, b_j), \\
D(i, j-1) + \delta, \\
D(i-1, j) + \delta,
\end{array} \right\}.$$  

$D(A, B) = D(n, m)$.

The computation time is $\sum_{i=1}^{n} \sum_{j=1}^{m} 1 = O(nm)$. 
Indel functions

Frequently in sequence evolution, deletion (or insertion) of several adjacent letters are not the sum of single deletions (or insertions) but the result of one event.

It is sometimes required to weight these multiple indels differently from summing single indel weights.

Let $g(k)$ be the indel weight for an indel of $k$ bases. It is reasonable that $g(k) \leq kg(1)$.

Set $D(i, j) = D(a_1 \cdots a_i, b_1 \cdots b_j)$, $D(0, 0) = 0$, $D(0, j) = g(j)$, and $D(i, 0) = g(i)$. Then

$$D(i, j) = \min \left\{ \begin{array}{l} D(i - 1, j - 1) + d(a_i, b_j), \\
\min_{1 \leq k \leq j} \{ D(i, j - k) + g(k) \}, \\
\min_{1 \leq l \leq i} \{ D(i - l, j) + g(l) \} \end{array} \right\}.$$ 

The computation time is $\sum_{i=1}^{n} \sum_{j=1}^{m} (i + j) = O(n^2m + m^2n)$. 

11
It is possible to reduce the running time for some specific $g(k)$.

When $g(k)$ is linear (affine), the running time is $O(nm)$.

Let $g(k) = \alpha + \beta(k - 1)$ for constants $\alpha$ and $\beta$, where $\alpha - \beta \geq 0$.

$\alpha$: gap opening cost. $\beta$: gap extension cost.

Set $D(0, 0) = 0$,
$D(i, 0) = g(i)$, $E(i, 0) = D(i, 0) + \alpha - \beta$,
$D(0, j) = g(j)$, and $F(0, j) = D(0, j) + \alpha - \beta$.

$E(\cdot, \cdot)$: alignment ends with insertion.
$F(\cdot, \cdot)$: alignment ends with deletion.

If $E(i, j)$ and $F(i, j)$ satisfy

$$E(i, j) = \min\{D(i, j - 1) + \alpha, E(i, j - 1) + \beta\}$$

and

$$F(i, j) = \min\{D(i - 1, j) + \alpha, F(i - 1, j) + \beta\},$$

then

$$D(i, j) = \min\{D(i - 1, j - 1) + d(a_i, b_j), E(i, j), F(i, j)\}.$$
The identities that we need to establish are

\[ E(i, j) = \min_{1 \leq k \leq j} \{ D(i, j - k) + g(k) \} \]

and

\[ F(i, j) = \min_{1 \leq l \leq i} \{ D(i - l, j) + g(l) \}. \]

We now prove the identity for \( E(i, j) \).

For \( j = 1 \),
\[
E(i, 1) = \min \{ D(i, 0) + \alpha, E(i, 0) + \beta \} = \min \{ D(i, 0) + \alpha, D(i, 0) + \alpha - \beta + \beta \} = D(i, 0) + g(1) = \min_{1 \leq k \leq 1} \{ D(i, 1 - k) + g(k) \}.
\]

For \( j > 1 \),
\[
\min_{1 \leq k \leq j} \{ D(i, j - k) + g(k) \} = \min \{ \min_{2 \leq k \leq j} \{ D(i, j - k) + g(k) \}, D(i, j - 1) + g(1) \} = \min \{ \min_{1 \leq k-1 \leq j-1} \{ D(i, (j - 1) - (k - 1)) + g(k - 1) + \beta \}, D(i, j - 1) + \alpha \} = \min \{ E(i, j - 1) + \beta, D(i, j - 1) + \alpha \}.
\]
Let $g(k) = g_{ini} + kg_{ext}$ for constants $g_{ini}$ and $g_{ext}$.

$g_{ini} \geq 0$: gap initiation cost. $g_{ext} \geq 0$: gap extension cost.

$A(i, j)$: optimal alignment.
$D(i, j)$: optimal alignment ends with deletion of $a_i$.
$I(i, j)$: optimal alignment ends with insertion of $b_j$.

$A(0, 0) = 0$;
$A(i, 0) = g_{ini} + i * g_{ext}$; $A(0, j) = g_{ini} + j * g_{ext}$;
$I(i, 0) = A(i, 0) + g_{ini}$; $D(0, j) = A(0, j) + g_{ini}$;

$D(i, j) = \min \begin{cases} D(i - 1, j) + g_{ext} \\ A(i - 1, j) + g_{ini} + g_{ext} \end{cases}$

$I(i, j) = \min \begin{cases} I(i, j - 1) + g_{ext} \\ A(i, j - 1) + g_{ini} + g_{ext} \end{cases}$

$A(i, j) = \min \begin{cases} A(i - 1, j - 1) + d(a_i, b_j) \\ I(i, j) \\ D(i, j) \end{cases}$
Proof of $D(i, j)$:

Let $M(i, j)$ be the optimal alignment ends with $a_i$ aligning with $b_j$, then

$$A(i, j) = \min \left\{ M(i, j), I(i, j), D(i, j) \right\}$$

Since $D(i, j)$ ends with the deletion of $a_i$, then in the optimal alignment, we have three cases.

1. $a_{i-1}$ is aligned with a space.
2. $b_j$ is aligned with a space.
3. $a_{i-1}$ is aligned with $b_j$. 
Therefore, we have

\[ D(i, j) = \min \begin{cases} 
D(i - 1, j) + g_{ext} \\
I(i - 1, j) + g_{ini} + g_{ext} \\
M(i - 1, j) + g_{ini} + g_{ext} 
\end{cases} \]

\[ = \min \begin{cases} 
D(i - 1, j) + g_{ext} \\
D(i - 1, j) + g_{ini} + g_{ext} \\
I(i - 1, j) + g_{ini} + g_{ext} \\
M(i - 1, j) + g_{ini} + g_{ext} 
\end{cases} \]

\[ = \min \begin{cases} 
D(i - 1, j) + g_{ext} \\
A(i - 1, j) + g_{ini} + g_{ext} 
\end{cases} \]
Global similarity alignment

Now let \( s(a, b) \) to be a similarity measure on the alphabet, that is, we have \( s(a, a) > 0 \) for all \( a \); for some \((a, b)\) pairs, it is necessary that \( s(a, b) < 0 \).

- Similarity is rewarded by a positive score
- Aligning dissimilar letter is penalized by a negative score
- Indel penalty is \(-\hat{g}(a)\).

Set \( s(a, -) = s(-, a) = -\hat{g}(a) \), then the similarity of \( A \) and \( B \) is,

\[
S(A, B) = \max_{(A', B')} \sum_{i=1}^{L} s(a'_i, b'_i)
\]
Distance and similarity alignments

Finding global distance alignments and global similarity alignments are dual problems.

Let a similarity measure be given with \( s(a, b) \) and indel penalties \( \hat{g}(k) \).

Let a distance measure be given with \( d(a, b) \) and indel weight \( g(k) \).

Assume that there is a constant \( c \) such that \( s(a, b) = c - d(a, b) \) and \( \hat{g}(k) = g(k) - (kc)/2 \).

Then an alignment is similarity optimal if and only if it is distance optimal.
Proof:

\[ n + m = 2\#\text{matches} + \sum_k k\Delta_k \]

where matches means aligned letters and \( \Delta_k \) is the number of indels of length \( k \).

\[
D(A, B) = \min \left\{ \sum_{\text{matches}} d(a, b) + \sum_k g(k)\Delta_k \right\}
\]

\[
= \min \left\{ \sum_{\text{matches}} c - s(a, b) + \sum_k k\Delta_kc/2 + \sum_k \hat{g}(k)\Delta_k \right\}
\]

\[
= \min \left\{ c(n + m)/2 - \sum_{\text{matches}} s(a, b) + \sum_k \hat{g}(k)\Delta_k \right\}
\]

\[
= c(n + m)/2 - \max \left\{ \sum_{\text{matches}} s(a, b) - \sum_k \hat{g}(k)\Delta_k \right\}
\]

\[
= c(n + m)/2 - S(A, B)
\]
Note that $D(A, B) + S(A, B) = c(n + m)/2$, so large distance means small similarity.

Usually $0 \leq c \leq max_{a',b'}d(a', b')$ so that we have both positive and negative similarity values.

Let $m = max_{a',b'}d(a', b')$ and $g(k) = \alpha + (k - 1)\beta$, then $\beta \geq m/2$ (triangle inequality).

$$\hat{g}(k) = \alpha + (k - 1)\beta - kc/2$$
$$\hat{g}(k) = (\alpha - c/2) + (k - 1)(\beta - c/2) \geq 0$$

This is, in general, a proof from distance to similarity. For general similarity, if we have $s(a, a) \neq s(b, b)$, then this approach will not lead to a distance measure.
If \( g(k, a_1, \ldots, a_k) \) or \( \hat{g}(k, a_1, \ldots, a_k) \) is linear or affine function, 
\[
g(k, a_1, \ldots, a_k) = g_{ini} + \sum_{i=1}^{k} d(a_i, -) \quad \text{and} \quad \hat{g}(k, a_1, \ldots, a_k) = g_{ini} - \sum_{i=1}^{k} s(a_i, -),
\]
then there are simple proofs for both directions.

Given a similarity measure 
\[
s(a, b) \quad \text{and} \quad \hat{g}(k, a_1, \ldots, a_k) = g_{ini} - \sum_{i=1}^{k} s(a_i, -)
\]
then
\[
d(a, b) = \frac{s(a, a) + s(b, b)}{2} - s(a, b),
\]
\[
g(k, a_1, \ldots, a_k) = g_{ini} + \sum_{i=1}^{k} \left( \frac{s(a_i, a_i)}{2} - s(a_i, -) \right)
\]
is a distance measure.
Given a distance measure 

\[ d(a, b) \] and 
\[ g(k, a_1, \ldots, a_k) = g_{ini} + \sum_{i=1}^{k} d(a_i, -) \]

then

\[ s(a, b) = \frac{d(a, -) + d(b, -)}{2} - d(a, b), \]

\[ \hat{g}(k, a_1, \ldots, a_k) = g_{ini} + \sum_{i=1}^{k} \frac{d(a_i, -)}{2} \]

is a similarity measure.
Fitting one sequence into another

Consider the problem of fitting $A$ to $B$, assuming that $n$ is much smaller than $m$.

The problem is to find

$$T(A, B) = \max\{S(A, b_k b_{k+1} \cdots b_{l-1} b_l) : 1 \leq k \leq l \leq m\}.$$ 

A direct solution according to the definition is of $O(n m^3)$.

Another approach is better.

Let

$$T_{i,j} = \max\{S(a_1 a_2 \cdots a_i, b_k b_{k+1} \cdots b_j) : 1 \leq k \leq j + 1\}.$$ 

Then,

$$T_{0,j} = 0, 0 \leq j \leq m$$
$$T_{i,0} = -i \delta, 0 \leq i \leq n$$

$$T_{i,j} = \max\{T_{i-1,j-1} + s(a_i, b_j), T_{i,j-1} - \delta, T_{i-1,j} - \delta\}$$

$$T(A, B) = \max\{T_{n,j} : 1 \leq j \leq m\}.$$
Local alignment

This is probably the most useful dynamic programming algorithm for current problems in molecular biology.

The problem is to find

$$ H(A, B) = \max \{0; S(a_i \cdots a_j, b_k \cdots b_l) : 1 \leq i \leq j \leq n, 1 \leq k \leq l \leq m \} $$

Define $H_{i,j}$ to be the maximum similarity of two segments ending at $a_i$ and $b_j$:

$$ H_{i,j} = \max \{0; S(a_x \cdots a_i, b_y \cdots b_j) : 1 \leq x \leq i, 1 \leq y \leq j \} $$

Then

$$ H_{0,j} = 0, 0 \leq j \leq m $$
$$ H_{i,0} = 0, 0 \leq i \leq n $$

$$ H_{i,j} = \max \{0, H_{i-1,j-1} + s(a_i, b_j), H_{i,j-1} - \delta, H_{i-1,j} - \delta \} $$

$$ H(A, B) = \max \{H_{i,j} : 1 \leq i \leq n; 1 \leq j \leq m \}.$$
If we consider arbitrary gap function, then

\[ H(0, 0) = 0, \ H(0, j) = 0, \text{ and } H(i, 0) = 0. \]

\[ H(i, j) = \max \left\{ \begin{array}{l}
0,
H(i - 1, j - 1) + s(a_i, b_i),
\max_{1 \leq k \leq j} \{H(i, j - k) - \hat{g}(k)\},
\max_{1 \leq l \leq i} \{H(i - l, j) - \hat{g}(l)\}
\end{array} \right\}. \]

\[ H(A, B) = \max \{H(k, l) : 1 \leq k \leq n, 1 \leq l \leq m\}. \]
When \( \hat{g}(k) \) is linear, that is \( \hat{g}(k) = \alpha + \beta(k - 1) \) for constants \( \alpha \) and \( \beta \).

Set
\[
H(0, 0) = 0, \\
H(i, 0) = 0, \ E(i, 0) = 0, \\
H(0, j) = 0, \text{ and } F(0, j) = 0.
\]

If \( E(i, j) \) and \( F(i, j) \) satisfy
\[
E(i, j) = \max\{H(i, j - 1) - \alpha, E(i, j - 1) - \beta\}
\]
and
\[
F(i, j) = \max\{H(i - 1, j) - \alpha, F(i - 1, j) - \beta\},
\]
then
\[
H(i, j) = \max\{0, H(i - 1, j - 1) + s(a_i, b_j), E(i, j), F(i, j)\}.
\]
Let \( g(k) = g_{ini} + kg_{ext} \) for constants \( g_{ini} \) and \( g_{ext} \).

\( g_{ini} \geq 0 \): gap initiation cost. \( g_{ext} \geq 0 \): gap extension cost.

**A**\((i,j)\): optimal local alignment.  
**D**\((i,j)\): optimal local alignment ends with deletion of \( a_i \).  
**I**\((i,j)\): optimal local alignment ends with insertion of \( b_j \).

\[
A(0,0) = 0;  
A(i,0) = 0; \quad A(0,j) = 0; \quad I(i,0) = 0; \quad D(0,j) = 0;  
\]

\[
D(i,j) = \max \begin{cases}  
D(i-1,j) - g_{ext} \\
A(i-1,j) - g_{ini} - g_{ext} 
\end{cases}
\]

\[
I(i,j) = \max \begin{cases}  
I(i,j-1) - g_{ext} \\
A(i,j-1) - g_{ini} - g_{ext} 
\end{cases}
\]

\[
A(i,j) = \max \begin{cases}  
0  \\
A(i-1,j-1) + s(a_i,b_j) \\
I(i,j)  \\
D(i,j) 
\end{cases}
\]
Algorithm $S$
Input: $n$, $m$, $A$, $B$, $s(,)$, $\delta$.
Output: $S$.

1. $S_{i,0} = -i\delta$, $i = 0, 1, \cdots n$
   
   $S_{0,j} = -j\delta$, $j = 0, 1, \cdots m$

2. For $i = 1$ to $n$
   
   For $j = 1$ to $m$
   
   $S_{i,j} = \max \begin{cases} 
   S_{i-1,j-1} + s(a_i, b_j), \\
   S_{i-1,j} - \delta, \\
   S_{i,j-1} - \delta 
   \end{cases}$

3. $S = S(n, m)$

Algorithm $S$ uses $O(nm)$ time and space.
Algorithm $S^*$
Input: $n$, $m$, $A$, $B$, $s(\cdot,\cdot)$, $\delta$.
Output: $S_{n,i}$, $0 \leq i \leq m$

1. $T_{1,j} = -j\delta$, $0 \leq j \leq m$
   For $i = 1$ to $n$
     For $j = 0$ to $m$
       $T_{0,j} = T_{1,j}$
       $T_{1,0} = -i\delta$
     For $j = 1$ to $m$
       $T_{1,j} = \max \left\{ T_{0,j-1} + s(a_i, b_j), T_{0,j} - \delta, T_{1,j-1} - \delta \right\}$

2. $S_{n,j} = T_{1,j}$, $0 \leq j \leq m$

Algorithm $S^*$ uses $O(nm)$ time and $O(n + m)$ space.

How can we trace back using linear space?
Let \( A_{1,i} = a_1 a_2 \cdots a_i, \ B_{1,j} = b_1 b_2 \cdots b_j, \  \hat{A}_{i+1,n} = a_n a_{n-1} \cdots a_{i+1} \) and \( \hat{B}_{j+1,m} = b_m b_{m-1} \cdots b_{j+1} \).

Algorithm \( SL \)

Input: \( n, m, A, B, s(,), \delta. \)  
Output: \( C \)

1. If \( m = 0 \) then \( C = \emptyset \)
   
   If \( n = 1 \) then
   
   If \( j \) satisfies \( s(a_1, b_j) = \max_k \{ s(a_1, b_k) \} > -2\delta \)
   
   then \( C = (1, j) \)
   
   else \( C = \emptyset \)

2. \( i = \lfloor n/2 \rfloor \)
   
   \( S^*(i, m, A_{1,i}, B_{1,m}, S_1) \)
   
   \( S^*(n - i, m, \hat{A}_{i+1,n}, \hat{B}_{1,m}, S_2) \)

3. \( M = \max_{0 \leq j \leq m} \{ S_1(j) + S_2(m - j) \} \)
   
   \( k = \min \{ j : S_1(j) + S_2(m - j) = M \} \)

4. \( SL(i, k, A_{1,i}, B_{1,k}, c_1) \)
   
   \( SL(n - i, m - k, A_{i+1,n}, B_{k+1,m}, c_2) \)

5. convert \( (j, l) \in c_2 \) to \( (i + j, k + l) \in c'_2 \).

6. \( C = c_1 \cup c'_2 \)
\[ S(a_1 \ldots a_n, b_1 \ldots b_m) = \]
\[ = \max_{0 \leq j \leq m} \left\{ S(a_1 \ldots a_{n/2}, b_1 \ldots b_j) + S(a_{n/2+1} \ldots a_n, b_{j+1} \ldots b_m) \right\} \]
\[ = \max_{0 \leq j \leq m} \left\{ S(A_{1,n/2}, B_{1,j}) + S(\hat{A}_{n/2+1,n}, \hat{B}_{j+1,m}) \right\} \]

Let \( S_2() \) be the solution of \( S^*(n - n/2, m, \hat{A}_{i+1,n}, \hat{B}_{1,m}, S_2) \), then
\[ S_2(m - j) = S(a_{n/2+1} \ldots a_n, b_{j+1} \ldots b_m). \]
Time complexity:

\[ T(n, m) = \frac{n}{2}m + \frac{n}{2}m + T(n/2, k) + T(n/2, m - k) \]
\[ = nm + T(n/2, k) + T(n/2, m - k). \]

We can now prove that \( T(n, m) \leq 2nm. \)

\[ T(n, m) = nm + T(n/2, k) + T(n/2, m - k) \]
\[ \leq nm + 2\frac{n}{2}k + 2\frac{n}{2}(m - k) \]
\[ = nm + nm \]
\[ = 2nm. \]
Linear space local alignment

\[ A = a_1 a_2 \cdots a_n \]
\[ B = b_1 b_2 \cdots b_m \]

1. Find \( i, j \) such that the best local alignment ends at
   \[ a_1 a_2 \cdots a_i \]
   \[ b_1 b_2 \cdots b_j \]

2. Use \( a_i a_{i-1} \cdots a_1 \) and
   \[ b_j b_{j-1} \cdots b_1 \]
   Find \( k, l \) such that the best alignment ends at
   \[ a_i a_{i-1} \cdots a_k \]
   \[ b_j b_{j-1} \cdots b_l \]

3. Use \( SL \) for
   \[ a_i a_{i-1} \cdots a_k \]
   \[ b_j b_{j-1} \cdots b_l \]
Example: Global distance alignment

\[ d(x, x) = 0, \quad d(x, y) = 1, \quad d(x, -) = 1. \]
Example: Global similarity alignment

\[
s(x, x) = 1, \ s(x, y) = -1, \ s(x, -) = -2.
\]
Example: Fit one sequence into another

\[
\begin{pmatrix}
GACACCATCGAA
gGCGCAAACCTT
\end{pmatrix}
\]

\[
\begin{pmatrix}
T-1-1-1-1-1-1-1
A302022201
T-521213211-1
A-743022210
A-96521313211
\end{pmatrix}
\]

\[
\begin{pmatrix}
TCGCGGTTATGACATGCTAGC
CGCCGGAAAGAGAATG
\end{pmatrix}
\]

Table 9.4: Matrix for best fit of TATAAT into the E. coli promoter of lacI

\[
s(x, x) = 1, \ s(x, y) = -1, \ s(x, -) = -2.
\]
Example: Local alignment

\[ s(x, x) = 2, \quad s(x, y) = -1, \quad s(x, -) = -2. \]