The number of all possible pairwise alignments (if gaps are allowed) is exponential in the length of the sequences. Therefore, the approach of "score every possible alignment and choose the best" is infeasible in practice. Efficient algorithms for pairwise alignment have been devised using dynamic programming (DP).

The key property of DP is that the problem can be divided into many smaller parts and the solution can be obtained from the solutions to these smaller parts.
The Needleman-Wunch Algorithm for Global Pairwise Alignment

- The problem is to align two sequences \( x (x_1 x_2 \ldots x_m) \) and \( y (y_1 y_2 \ldots y_n) \) finding the best scoring alignment in which all residues of both sequences are included.
- The score is assumed to be a measure of similarity, so the highest score is desired.
- Alternatively, the score could be an evolutionary distance, in which case the smallest score would be sought, and all uses of “max” in the algorithm would be replaced by “min.”

The key concept in all these algorithms is the matrix \( S \) of optimal scores of subsequence alignments.

The matrix has \((m+1)\) rows labeled \( 0 \rightarrow m \) and \((n+1)\) columns labeled \( 0 \rightarrow n \).

The rows correspond to the residues of sequence \( x \), and the columns correspond to the residues of sequence \( y \).

We’ll use as a working example the two sequences \( x=THISLINE \) and \( y=SALIGNED \) with BLOSUM-62 substitution matrix as the scoring matrix and linear gap penalty \( g \).

The optimal alignment of these two sequences is:

\[
\begin{align*}
THISLINE & - \\
\_SALIGNED & \\
\end{align*}
\]
The Matrix $S$

$S_{i,j}$ stores the score for the optimal alignment of all residues up to $x_i$ of sequence $x$ with all residues up to $y_j$ of sequence $y$.

$S_{i,0} = S_{0,i} = ig$

To complete the matrix, we use the formula:

$$S_{i,j} = \max \left\{ S_{i-1,j-1} + s(x_i, y_j), S_{i-1,j} + g, S_{i,j-1} + g \right\}$$

The global alignment can be obtained by traceback.
The alignment we obtained is not the one we expected in that it contains no gaps (the gap at the end is added only because the two sequences are of different lengths)

The “problem” is that the worst score in the BLOSUM-62 matrix is -4, which is significantly less than the gap penalty (-8), so gaps are unlikely to be present in the optimal alignment.

If instead we use a linear gap penalty of -4, inserting a gap becomes less severe, and a gapped alignment is more likely to be obtained.

Therefore it is very important to match the gap penalty to the substitution matrix used.

\[
S_{ij} = \max \begin{cases}
  S_{i-1,j-1} + (s(x_i, y_j)) \\
  S_{i-1,j} - g \\
  S_{i,j-1} + g
\end{cases}
\]

The global alignment can be obtained by traceback.
**Multiple Optimal Alignments**

- There may be more than one optimal alignment.
- During traceback, this is indicated by encountering an element that was derived from more than one of the three possible alternatives.
- The algorithm does not distinguish between these possible alignments, although there may be reasons (such as knowledge of molecular structure or function) for preferring one to the others.
- Most programs will arbitrarily report just one single alignment.

**General Gap Penalty**

- The algorithm we presented works with a linear gap penalty of the form \( g(n_{\text{gap}}) = -n_{\text{gap}}E \).
- For a general gap penalty model \( g(n_{\text{gap}}) \), one must consider the possibility of arriving at \( S_{i,j} \) directly via insertion of a gap of length up to \( i \) in sequence \( x \) or \( j \) in sequence \( y \).
General Gap Penalty

The algorithm now has to be modified to

\[ S_{i,j} = \max \left\{ \begin{align*} S_{i-1,j-1} &+ s(x_i,y_j) \\ S_{i-1,j} &+ g(n_{gap1}) \\ S_{i,j-1} &+ g(n_{gap2}) \end{align*} \right\} \]

This algorithm takes time that is proportional to \( mn^2 \), where \( m \) and \( n \) are the sequence lengths with \( n > m \).

Affine Gap Penalty

For an affine gap penalty \( g(n_{gap}) = -I - (n_{gap} - 1)E \), we can refine this algorithm to obtain an \( O(mn) \) algorithm.

Define two matrices

\[ V_{i,j} = \max \left\{ S_{i-1,j-1} + g(n_{gap1}) \right\} \]
\[ W_{i,j} = \max \left\{ S_{i-1,j} + g(n_{gap1}), S_{i,j-1} + g(n_{gap2}) \right\} \]

These matrices can be defined recursively as

\[ V_{i,j} = \max \left\{ V_{i-1,j-1} - I, V_{i-1,j} - E \right\} \]
\[ W_{i,j} = \max \left\{ W_{i-1,j-1} - I, W_{i,j-1} - E \right\} \]

Now, the matrix \( S \) can be written as

\[ S_{i,j} = \max \left\{ S_{i-1,j-1} + s(x_i,y_j), V_{i,j}, W_{i,j} \right\} \]
As mentioned before, sometimes local alignment is more appropriate (e.g., aligning two proteins that have just one domain in common).

The algorithmic differences between the algorithm for local alignment (Smith-Waterman algorithm) and the one for global alignment:

- Whenever the score of the optimal sub-alignment is less than zero, it is rejected (the matrix element is set to 0).
- Traceback starts from the highest-scoring matrix element.

Local Pairwise Alignment

$$S_{i,j} = \max \begin{cases} S_{i-1,j-1} + s(x_i, y_j) \\ (S_{i-1,j+1} + g(n_{gap1}))_{\leq n_{gap1} \leq i} \\ (S_{i,j+2} + g(n_{gap2}))_{\leq n_{gap2} \leq j} \\ 0 \end{cases}$$
Is there a linear-space algorithm for the problem?

- If only the maximal score is needed, the problem is simple.
- But even if the alignment itself is needed, there is a linear-space algorithm (originally due to Hirschberg 1975 and introduced into computational biology by Myers and Miller 1988).

Main observation

\[ S_{i,j} = \max_{0 \leq k \leq n} \{ S_{i,2} + S_{r,i}^k + S_{r,j}^n - k \} \]

- \( S_{i,j} \) score of the best alignment of last \( i \) residues of \( x \) with last \( j \) residues of \( y \).
Linear Space Alignment

- Compute $S_{m,n}$ and save row $m/2$
- Compute $S'_{m,n}$ and save row $m/2$
- Find $k^*$ that satisfies $S_{m,k^*} + S'_{n,n-k^*} = S_{m,n}$
- Recurse

Questions?