# Pairwise alignment



Outline of the chapter:

1. Scoring matrices
2. Global, local, and overlap alignment of two sequences using dynamic programming
3. Affine gap penalty
4. Modifications that reduce the space and time complexity

# Comparing sequences (2)



Sequence comparison is the most important computation in bioinformatics.

**Key assumption**

Sequence similarity $\Rightarrow$ structural similarity $\Rightarrow$ functional similarity

Moreover, a high level of sequence similarity implies homology.

How to measure sequence similarity?

## Definition (Alignment)

Let $X$ and $Y$ be sequences over an alphabet $\Sigma$. An *alignment* $A$ of $X$ and $Y$ is obtained by inserting dashes ('-') so that both resulting sequences $X' = x'_1 \dots x'_n$ and $Y' = y'_1 \dots y'_n$ of equal length can be written one above the other in such a way that each character in the one sequence is opposite a unique character in the other sequence.

Note: Usually, we require that no two dashes are aligned in this way.

Example:

$$
\begin{array}{lccccccccccc}
X' = & \mathtt{Y} & \mathtt{E} & \mathtt{-} & \mathtt{S} & \mathtt{T} & \mathtt{E} & \mathtt{R} & \mathtt{D} & \mathtt{A} & \mathtt{Y} \\
Y' = & \mathtt{-} & \mathtt{E} & \mathtt{A} & \mathtt{S} & \mathtt{T} & \mathtt{E} & \mathtt{R} & \mathtt{-} & \mathtt{-} & \mathtt{-}
\end{array}
$$

# Sequence similarity

In order to evaluate an alignment we need a *scoring scheme* that provides a score for the alignment of any pair of symbols.

Scores can either reflect disimilarity or similarity. An example of the former case is the *edit distance*. In the following we focus on the latter case.

Let $X = x_1 \dots x_n$ and $Y = y_1 \dots y_m$ be sequences over an alphabet $\Sigma$.

A *similarity score* $s : \Sigma \cup \{-\} \times \Sigma \cup \{-\} \to \mathbb{R}$ assigns a value to each pair of characters in $\Sigma \cup \{-\}$.

Let $A$ be an alignment of $X$ and $Y$. The *score* $S(A)$ of $A$ is defined as

$$S(A) = \sum_{i=1}^{n} s(x'_i, y'_i).$$

# Sequence similarity (2)

For example, consider this alignment $A$:

$$X' = \texttt{R N A - R N A}$$
$$Y' = \texttt{A N A R R N -}$$

and these similarity scores:

| $s$ | A | R | N | – |
|---|---|---|---|---|
| A | 4 | −1 | −2 | −3 |
| R | −1 | 5 | 0 | −3 |
| N | −2 | 0 | 6 | −3 |
| – | −3 | −3 | −3 | 0 |

The similarity score for the alignment $A$ is calculated as follows:

$$X' = \texttt{R N A - R N A}$$
$$Y' = \texttt{A N A R R N -}$$
$$\texttt{-1 +6 +4 -3 +5 +6 -3} = 14 = S(A)$$

# Substitution matrices



To be able to score an alignment, we need to lookup a score term for each aligned pair of symbols.

**Definition (Substitution matrix)**

A *substitution matrix* $S$ over an alphabet $\Sigma = \{a_1, \ldots, a_k\}$ has $k \times k$ entries, where each entry $S(i, j)$ assigns a score for the substitution of the symbol $a_i$ by the symbol $a_j$ in an alignment.

(Substituion matrices do not provide values for the gap character "–").

# Substitution matrices (2)



A substitution matrix can be computed emprically by counting letters and substitutions that occur in a *database of trusted alignments*, as follows:

- Consider only non-gapped alignments
- Compute the relative frequency $f(a)$ of each letter $a$ in the database
- Compute the frequency $f(a, b)$ with which $a$ aligns to $b$ in the database
- We define a score as follows:

$$s(a, b) = \frac{f(a, b)}{f(a)f(b)}$$

The expression $f(a)f(b)$ is the *expected* number of substitutions of $a$ by $b$ under a random model.

The expression $f(a, b)$ is what we actually *observe* in the database.

# Substitution matrices (3)

In more detail, consider a non-gapped alignment

$$\begin{aligned} X &= x_1 x_2 \dots x_n \\ Y &= y_1 y_2 \dots y_n \end{aligned}$$

We will consider two different models.

*Random model R*: The two sequences are unrelated (not homologous). Then, each letter $a$ occurs independently with some probability $p_a$, and hence the probability of the two sequences is the product:

$$P(X, Y \mid R) = P(X \mid R)P(Y \mid R) = \prod_i p_{x_i} \prod_i p_{y_i}.$$

The probability $p_a$ can be approximated by the frequency $f(a)$ discussed above.

# Substitution matrices (4)

*Match* model $M$: The two sequences are related (homologous). The aligned pairs of residues $a$ and $b$ occur with a joint probability $p_{ab}$, which is the probability that $a$ and $b$ have each evolved from some unknown original residue $c$ as their common ancestor. Thus, the probability for the whole alignment is:

$$P(X, Y \mid M) = \prod_{i=1}^{n} p_{x_i y_i}.$$

The probability $p_{ab}$ can be approximated by the frequency $f(a, b)$.

# Odds ratio (5)

The two values $P(X, Y \mid R)$ and $P(X, Y \mid M)$ are called *likelihoods*.

Their ratio measures the "relative likelihood" that the sequences are related (model $M$) as opposed to being unrelated (model $R$). This ratio is called *odds ratio*:

$$\frac{P(X, Y \mid M)}{P(X, Y \mid R)} = \frac{\prod_i p_{x_i y_i}}{\prod_i p_{x_i} \prod_i p_{y_i}} = \prod_{i=1}^{n} \frac{p_{x_i y_i}}{p_{x_i} p_{y_i}}$$

# Log-odds ratio (6)

To obtain an additive scoring scheme, we take logarithms (base 2) to get the *log-odds ratio:*

$$\log\left(\frac{P(X,Y \mid M)}{P(X,Y \mid R)}\right) = \log\left(\prod_i \frac{p_{x_i y_i}}{p_{x_i} p_{y_i}}\right) = \sum_i \log\left(\frac{p_{x_i y_i}}{p_{x_i} p_{y_i}}\right) = \sum_{i=1}^{n} s(x_i, y_i),$$

with

$$s(a,b) = \log\left(\frac{f(a,b)}{f(a)f(b)}\right).$$

In a preprocessing step, the frequencies are calculated from a curated database of trusted alignments.

(This is a very simple example of "machine learning".)

# BLOCKS and BLOSUM matrices



One important family of substitution matrices that were produced as described above are the BLOSUM matrices (BLOSUM = BLOcks SUbstitution Matrix).

They were derived from the BLOCKS database (http://blocks.fhcrc.org/).

Blocks are multiple sequence alignments of ungapped segments of highly conserved regions of proteins.

In the calculation of a BLOSUM matrix, a threshold $t$ is given and then only sequences that are less than $t$ % identical are considered together.

For example, the BLOSUM62 matrix is based on comparisons of sequences that are less than 62% identical.

# BLOCKS and BLOSUM matrices (2)



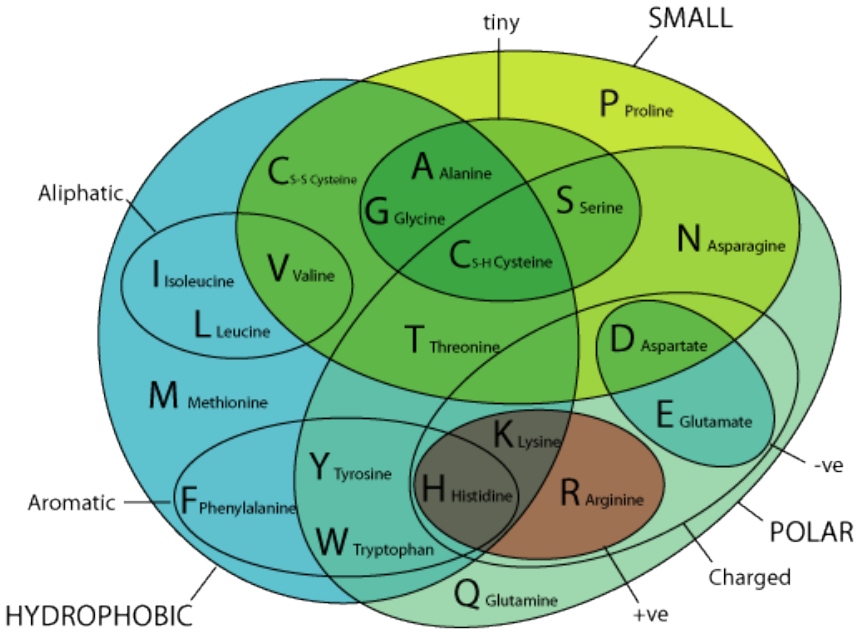
**Block IPB000039A**

```
ID   Ribosomal_L18e; BLOCK
AC   IPB000039A; distance from previous block=(-18,154)
DE   Ribosomal protein L18e
BL   LFS;   width=42; seqs=159; 99.5%=2045; strength=1341
RL18_SALSA|P24558     (  23) ITLLVKIYRLPGSKCSTAPFNKVVLRRLFMSRTHRPPMSVSR  48

RL18_TRYBB|P50885     (  25) YIKLLIKLYKFLGKRTNSPFNKLIHKRLLKSRNNRAPISLSR  12

P42791|RL182_ARATH    (  25) YLKLTVKLYRFLVRRTNSKFNGVILKRLFMSKVNKAPLSLSR   9
RL18_CICAR|O65729     (  22) YLKLLVKLYRFLVRRTDSNFNAVILKRLFMSKVNRPPLSLSR   4
RL18_HUMAN|Q07020     (  23) YLRLLVKLYRFLARRTNSTFNQVVLKRLFMSRTNRPPLSLSR   3
RL18_ICTPU|Q90YV0     (  23) YLRLLVKLYRFLARRSDAPFNKVVLKRLFMSKTNRPPLALSR   4
RL18_MOUSE|P35980     (  23) YLRLLVKLYRFLARRTNSTFNQVVLKRLFMSRTNRPPLSLSR   3
RL18_OREMO|Q9I836     (  23) YLRLLVKLYRFLARRSNAPFNRVVLRRLFMSRTNRPPIAVSR   5
RL18_RAT|P12001       (  23) YLRLLVKLYRFLARRTNSTFNQVVLKRLFMSRTNRPPLSLSR   3
RL18A_SCHPO|Q10192    (  24) YLKLLVKLYRFLARRTDSRFNKAILKRLFQSKTNRPPISISK   7
RL18B_XENLA|P02412    (  23) YLRLLVKLYRFLARRTNSSFNRVVLKRLFMSRTNRPPLSMSR   5
RL18_YEAST|P07279     (  25) YLKLLVKLYTFLARRTDAPFNKVVLKALFLSKINRPPVSVSR  11
RL18A_XENLA|P09897    (  23) YLRLLVKLYRFLARRTNSSFNRVVLKRLFMSRTNRPPLSMSR   5
```

BLOSUM62 is scaled so that its values are in "half-bits", ie. the log-odds are multiplied by 2 and then rounded to the nearest integer value:

| | A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 4 | -1 | -2 | -2 | 0 | -1 | -1 | 0 | -2 | -1 | -1 | -1 | -1 | -2 | -1 | 1 | 0 | -3 | -2 | 0 |
| R | -1 | 5 | 0 | -2 | -3 | 1 | 0 | -2 | 0 | -3 | -2 | 2 | -1 | -3 | -2 | -1 | -1 | -3 | -2 | -3 |
| N | -2 | 0 | 6 | 1 | -3 | 0 | 0 | 0 | 1 | -3 | -3 | 0 | -2 | -3 | -2 | 1 | 0 | -4 | -2 | -3 |
| D | -2 | -2 | 1 | 6 | -3 | 0 | 2 | -1 | -1 | -3 | -4 | -1 | -3 | -3 | -1 | 0 | -1 | -4 | -3 | -3 |
| C | 0 | -3 | -3 | -3 | 9 | -3 | -4 | -3 | -3 | -1 | -1 | -3 | -1 | -2 | -3 | -1 | -1 | -2 | -2 | -1 |
| Q | -1 | 1 | 0 | 0 | -3 | 5 | 2 | -2 | 0 | -3 | -2 | 1 | 0 | -3 | -1 | 0 | -1 | -2 | -1 | -2 |
| E | -1 | 0 | 0 | 2 | -4 | 2 | 5 | -2 | 0 | -3 | -3 | 1 | -2 | -3 | -1 | 0 | -1 | -3 | -2 | -2 |
| G | 0 | -2 | 0 | -1 | -3 | -2 | -2 | 6 | -2 | -4 | -4 | -2 | -3 | -3 | -2 | 0 | -2 | -2 | -3 | -3 |
| H | -2 | 0 | 1 | -1 | -3 | 0 | 0 | -2 | 8 | -3 | -3 | -1 | -2 | -1 | -2 | -1 | -2 | -2 | 2 | -3 |
| I | -1 | -3 | -3 | -3 | -1 | -3 | -3 | -4 | -3 | 4 | 2 | -3 | 1 | 0 | -3 | -2 | -1 | -3 | -1 | 3 |
| L | -1 | -2 | -3 | -4 | -1 | -2 | -3 | -4 | -3 | 2 | 4 | -2 | 2 | 0 | -3 | -2 | -1 | -2 | -1 | 1 |
| K | -1 | 2 | 0 | -1 | -3 | 1 | 1 | -2 | -1 | -3 | -2 | 5 | -1 | -3 | -1 | 0 | -1 | -3 | -2 | -2 |
| M | -1 | -1 | -2 | -3 | -1 | 0 | -2 | -3 | -2 | 1 | 2 | -1 | 5 | 0 | -2 | -1 | -1 | -1 | -1 | 1 |
| F | -2 | -3 | -3 | -3 | -2 | -3 | -3 | -3 | -1 | 0 | 0 | -3 | 0 | 6 | -4 | -2 | -2 | 1 | 3 | -1 |
| P | -1 | -2 | -2 | -1 | -3 | -1 | -1 | -2 | -2 | -3 | -3 | -1 | -2 | -4 | 7 | -1 | -1 | -4 | -3 | -2 |
| S | 1 | -1 | 1 | 0 | -1 | 0 | 0 | 0 | -1 | -2 | -2 | 0 | -1 | -2 | -1 | 4 | 1 | -3 | -2 | -2 |
| T | 0 | -1 | 0 | -1 | -1 | -1 | -1 | -2 | -2 | -1 | -1 | -1 | -1 | -2 | -1 | 1 | 5 | -2 | -2 | 0 |
| W | -3 | -3 | -4 | -4 | -2 | -2 | -3 | -2 | -2 | -3 | -2 | -3 | -1 | 1 | -4 | -3 | -2 | 11 | 2 | -3 |
| Y | -2 | -2 | -2 | -3 | -2 | -1 | -2 | -3 | 2 | -1 | -1 | -2 | -1 | 3 | -3 | -2 | -2 | 2 | 7 | -1 |
| V | 0 | -3 | -3 | -3 | -1 | -2 | -2 | -3 | -3 | 3 | 1 | -2 | 1 | -1 | -2 | -2 | 0 | -3 | -1 | 4 |

It is worthwhile investigating to what degree the values in a substitution matrix reflect biochemical properties of amino acids, as summarized here:




(Image from: Livingstone, C.D. and Barton, G.J. (1993), as displayed at `http://www.jalview.org/help/html/misc/aaproperties.html`)

For example, the BLOSUM62 values between *I* (isoleucine), *V* (valine) and *L* (leucine) are all positive, and these acids are the hydrophobic, aliphatic amino acids.

# Optimal alignment



Assume we are given a similarity scoring scheme. The *similarity* of two sequences $X$ and $Y$ is defined as the maximum alignment score attained by any alignment $A$ of $X$ and $Y$. Any alignment that attains the maximum score is called *optimal*.

The aim is to use optimal alignment so as to

- detect *homology* of sequences, i.e. *similarity due to a common evolutionary origin*, and to
- avoid *analogy*, i.e. *similarity due to convergent evolution*.

# Gap penalties (2)



Gaps in an alignment are undesirable and thus penalized. The standard cost associated with a run of gaps of length $k$ is given either by a *linear* score

$$\gamma(k) = -kd$$

or an *affine* penalty

$$\gamma(k) = -d - (k-1)e,$$

where $d$ is the *gap open* penalty, $e$ is the *gap extension* penalty and $d > e$.

# Gap penalties (3)

The advantage of using an affine gap penalty is that it encourages gaps to appear together in runs, as illustrated in the following example:

Linear gap penalty:

```
GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKL
GSAQVKGHGKK--------VA--D----A-SALSDLHAHKL
```

Affine gap penalty:

```
GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKL
GSAQVKGHGKKVADA---------------SALSDLHAHKL
```

# Alignment algorithms



Alignment algorithms based on *dynamic programming* are guaranteed to find optimal alignments.

The standard *dynamic programming* approach has three essential components:

- the recurrence relation,
- the tabular computation, and
- the traceback.

We will consider three variants of the alignment problem, namely

1. *global alignment*,
2. *local alignment*, and
3. *overlap* alignment.

# Global alignment: Needleman-Wunsch algorithm

The Needleman-Wunsch algorithm is a dynamic program that solves the problem of obtaining the best *global* alignment of two sequences $X$ and $Y$. For the case of a linear gap penalty $d > 0$, the recursion is defined as follows:

**Initialization:**

$F(i, 0) = -i \times d$ for all $i = 0, 1, 2, \ldots, n$,

$F(0, j) = -j \times d$ for all $j = 0, 1, 2, \ldots, m$

**Recursion:**

$$F(i,j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_j) \\ F(i-1, j) - d \\ F(i, j-1) - d \end{cases}$$

The score of the optimal global alignment is $\alpha = F(n, m)$

# Example of a global alignment matrix

Needleman-Wunsch matrix for sequences $X = \texttt{CTAATGCG}$ and $Y = \texttt{TAATGG}$, scores $s(a, a) = 1$, $s(a, b) = -1$ and linear gap cost $d = 1$:

| $F$ | 0 | C | T | A | A | T | G | C | G |
|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | |
| T | | | | | | | | | |
| A | | | | | | | | | |
| T | | | | | | | | | |
| G | | | | | | | | | |
| G | | | | | | | | | |

Score: ; Alignment:

# Example of a global alignment matrix (2)



Needleman-Wunsch matrix for sequences $X =$ CTAATGCG and $Y =$ TAATGG, scores $s(a, a) = 1$, $s(a, b) = -1$ and linear gap cost $d = 1$:

| $F$ | 0 | C | T | A | A | T | G | C | G |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 |
| T | -1 | -1 | 0 | -1 | -2 | -3 | -4 | -5 | -6 |
| A | -2 | -2 | -1 | 1 | 0 | -1 | -2 | -3 | -4 |
| T | -3 | -3 | -1 | 0 | 0 | 1 | 0 | -1 | -2 |
| G | -4 | -4 | -2 | -1 | -1 | 0 | 2 | 1 | 0 |
| G | -5 | -5 | -3 | -2 | -2 | -1 | 1 | 0 | 2 |

Score: 2

# Traceback

To obtain an alignment corresponding to this score, we must find the path of choices that the recursion made to obtain the score. This is called a *traceback*.

Starting at the last cell $(n, m)$, we repeatedly move to "the" previous cell from which the value $F(i, j)$ for the current cell $(i, j)$ was computed, until we reach the cell $(0, 0)$. At each step, we record a match, mismatch, insert or deletion, appropriately.

The selected traceback is shown in bold:

| $F$ | 0 | C | T | A | A | T | G | C | G |
|---|---|---|---|---|---|---|---|---|---|
| 0 | **0** | **-1** | -2 | -3 | -4 | -5 | -6 | -7 | -8 |
| T | -1 | -1 | **0** | **-1** | -2 | -3 | -4 | -5 | -6 |
| A | -2 | -2 | -1 | 1 | **0** | -1 | -2 | -3 | -4 |
| T | -3 | -3 | -1 | 0 | 0 | **1** | 0 | -1 | -2 |
| G | -4 | -4 | -2 | -1 | -1 | 0 | **2** | **1** | 0 |
| G | -5 | -5 | -3 | -2 | -2 | -1 | 1 | 0 | **2** |

Score: 2; Alignment:

| C | T | A | A | A | T | G | C | G |
|---|---|---|---|---|---|---|---|---|
| - | T | - | A | A | T | G | - | G |

# Complexity

What is the complexity of the Needleman-Wunsch algorithm?

**Space:** We need to store $(n+1) \times (m+1)$ numbers, so the space complexity is $O(nm)$.

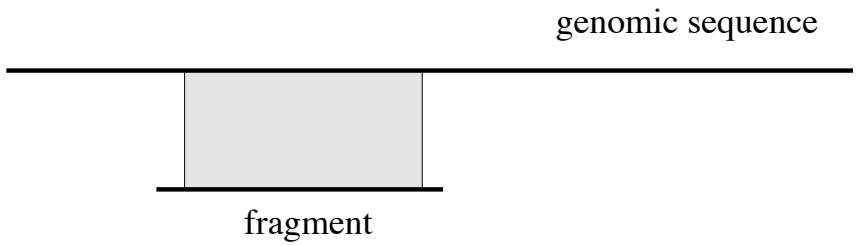
**Time:** Each cell of $F$ requires a constant number of operations, namely three additions and a max, so the time complexity is $O(nm)$. Traceback requires linear time.

Later we will see how to reduce the space complexity to linear and the time complexity to linear, too, the latter *only for very similar sequences*.

# Local alignment

Global alignment is applicable when we have two similar sequences that we want to align from end-to-end, e.g. two homologous genes from related species.

Often, however, we have two sequences $X$ and $Y$ and we would like to find the best match between *substrings* of both. For example, we may want to find the position of a fragment of DNA in a genomic sequence:

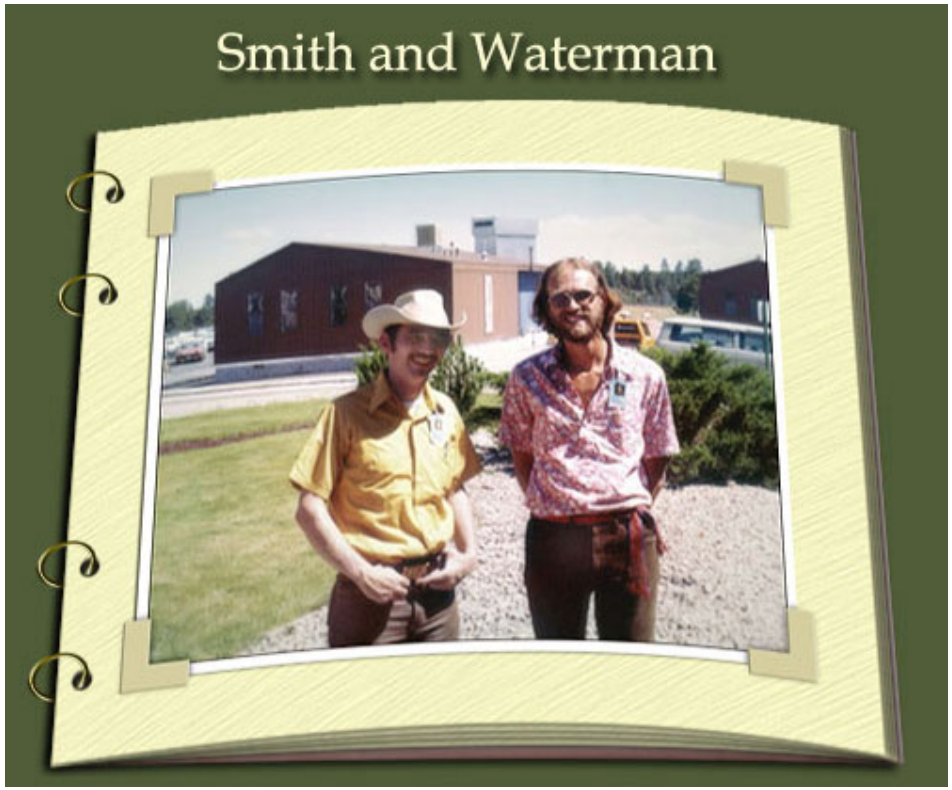



A local alignment is a best-scoring alignment of two substrings of $X$ and $Y$.

# Smith-Waterman algorithm

The exact definition of a local alignment is "the alignment returned by the Smith-Waterman algorithm".

- Published in: Smith, T. and Waterman, M. Identification of common molecular subsequences. J. Mol. Biol. 147:195-197, 1981




Temple Smith and Michael Waterman, picture from http://www-hto.usc.edu/people/msw/Waterman.html

# Smith-Waterman algorithm (2)

The Smith-Waterman local alignment algorithm:

**Initialization**:
$F(i,0) = 0$ for all $i = 0, 1, 2, \ldots, n$, and
$F(0,j) = 0$ for all $j = 0, 1, 2, \ldots, m$.

Recursion:

$$F(i,j) = \max \begin{cases} 0, \\ F(i-1, j-1) + s(x_i, y_j), \\ F(i-1, j) - d, \\ F(i, j-1) - d. \end{cases}$$

The value $F(i,j) = 0$ indicates " start a new alignment at $(i,j)$".

The cell with the highest score: $\arg\max F(i,j)$.

# Smith-Waterman algorithm (3)



The traceback:

- starts at cell with largest value, $\arg\max F(i, j)$, and
- ends at the first cell encountered that has value 0.

# Smith-Waterman algorithm: Traceback

For the algorithm to work, the average score for randomly aligned letters must be negative, i.e.:

$$\sum_{a,b \in \Sigma} p_a \cdot p_b \cdot s(a,b) < 0,$$

where $p_a$ and $p_b$ are the probabilities for the seeing the symbol $a$ or $b$ respectively, at any given position.

# Example



Smith-Waterman matrix for two sequences $X = \texttt{CTAATGCC}$ and $Y = \texttt{TATGG}$, with $s(a, a) = 1$, $s(a, b) = -1$ and gap cost $d = 1$:

| $F$ | 0 | C | T | A | A | T | G | C | C |
|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | |
| T | | | | | | | | | |
| A | | | | | | | | | |
| T | | | | | | | | | |
| G | | | | | | | | | |
| G | | | | | | | | | |

Score: ____; Alignment =

# Example (2)

Smith-Waterman matrix for two sequences $X = \texttt{CTAATGCC}$ and $Y = \texttt{TATGG}$, with $s(a, a) = 1$, $s(a, b) = -1$ and gap cost $d = 1$:

| $F$ | 0 | C | T | A | A | T | G | C | C |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| T | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| A | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 |
| T | 0 | 0 | 0 | 1 | 1 | 2 | 1 | 0 | 0 |
| G | 0 | 0 | 1 | 0 | 1 | 1 | 3 | 2 | 1 |
| G | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 1 |

Score: 3

Alignment $=$

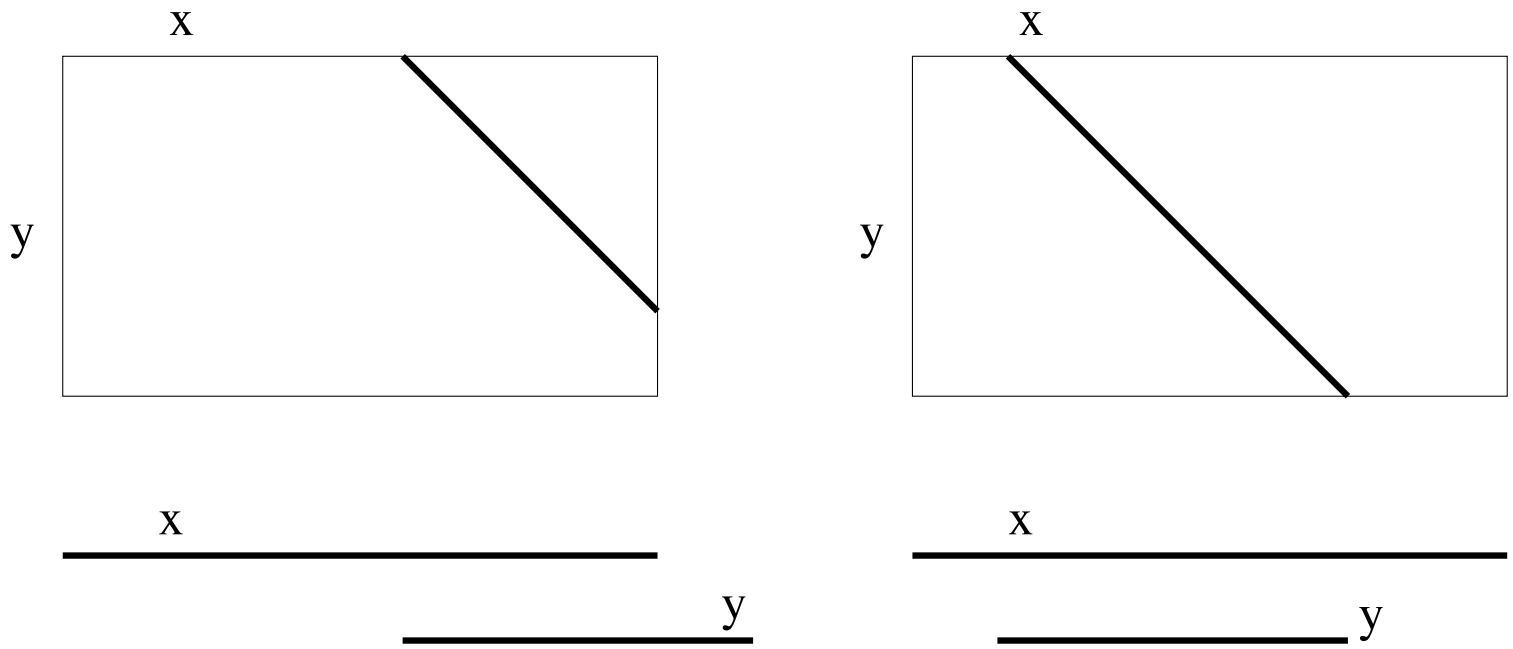
| | | | | | |
|---|---|---|---|---|---|
| T | A | A | T | G |
| T | - | A | T | G |

# Complexity of the algorithm

Same as the Needleman-Wunsch algorithm: $O(nm)$

# Overlap alignments

If we are given overlapping fragments of DNA that we would like to assemble, then we need an alignment method that does not penalize overhanging ends:



# Algorithm for finding overlap alignments (2)



Alignment should (a) be allowed to start anywhere on the top or left boundary of the matrix, and should (b) be allowed to end anywhere on the bottom or right boundary.

For (a), initialize as in S-W algorithm: $F(i, 0) = 0$ and $F(0, j) = 0$ for $i = 0, 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$.

For alignment, use the same recursion as the N-W algorithm.

For (b), start the traceback at the best scoring cell contained in the bottom-most row or rightmost column, i.e. start at

$$\arg\max\{F(i, j) \mid i = n \text{ or } j = m\}.$$

# Example of an overlap alignment

Overlap-alignment matrix for $X =$ ACATATT and $Y =$ TATTAC, with $s(a, a) = 1$, $s(a, b) = -1$ and gap cost $= 2$:

| | 0 | A | C | A | T | A | T | T |
|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | |
| T | | | | | | | | |
| A | | | | | | | | |
| T | | | | | | | | |
| T | | | | | | | | |
| A | | | | | | | | |
| C | | | | | | | | |

score: 4
ACATATT--
---TATTAC

Score=
Alignment=

aligning dna to protein:
i-3,j-1 and assign
similarity score