Pairwise alignment

We will discuss:

- The principle of *dynamic programming*
- Dot plots
- Scoring schemes
- Alignment algorithms based on dynamic programming

References

- R. Durbin, S. Eddy, A. Krogh, G. Mitchison: *Biological sequence analysis*. Cambridge University Press, 1998. ISBN 0-521-62971-3 (Chapter 2)
- Neil C. Jones, Pavel A. Pevzner: *An Introduction to Bioinformatics Algorithms*. MIT Press, Cambridge, MA, 2004. ISBN 0-262-10106-8
- David B. Mount: *Bioinformatics*. Sequence and Genome Analysis. Cold Spring Harbor Laboratory Press, New York, 2001. ISBN 0-87969-608-7

Paradigm: Dynamic Programming

Powerful algorithmic design paradigm

Principle

Compute solutions of “bigger” problems by combining solutions of smaller subproblems. Storing all solutions avoids recomputing the same quantity over and over again, and a potential exponential blow-up in the running time.

Well-suited if subproblems share subsubproblems.

Typically applied to *optimization problems*. Three components:

1. Recursively define the value of an optimal solution
2. Compute values of subproblems in bottom-up fashion (storing the solution values)
3. Construct solution (traceback)

Paradigm: Dynamic Programming (2)

Examples

Fibonacci numbers*, knapsack, longest common subsequence†, alignments in database search (BLAST, FASTA), multiple alignments (clustalW), gene finding (GENSCAN), RNA-folding (mfold), phylogenetic inference (PHYLIP), . . .

*blackboard

†see Ex. 3 on first assignment

Pairwise sequence alignment: motivation

Comparative Genomics

- Gene finding, gene function determination
  Compare sequence to genes of known function. Success stories:
  - Example 1: ν-sis-oncogene* (discovered 1984). Comparison with all known genes led to striking similarity with regular growth gene. Conclusion: oncogene responsible for growth at the wrong time.
  - Example 2: discovery of the cystic fibrosis gene (1989). Location narrowed down to 10^6 nucleotides on chromosome 7. Compared to all genes $\rightarrow$ similarity with gene for ATP binding proteins. Shed light on the nature of cystic fibrosis.
  - ...

*oncogenes are virus genes that cause cancer-like transformation of infected cells

Pairwise sequence alignment: motivation (2)

- High sequence similarity (*e.g.*, human—mouse: 97%) between species allows to study other organisms in order to understand humans (*e.g.*, Waardenburg's syndrome)
- derive information about common origin (evolutionary trees)
- . . .

Sequence alignment (informally)

Two strings: $\longrightarrow$ Alignment:

```
IMISSMISSISSIPPI          I-MISSMISSISIPPI-
                            |||| ||    ||||
MYMISSISAHIPPIE           MYMISS-ISAH-IPPIE
```

`I`: Isoleucine, `M`: Methionine, `S`: Serine, `P`: Proline, `Y`: Tyrosine, `A`: Alanine, `H`: Histidine, `E`: Glutamic Acid

Sequence alignment (informally) (2)

Two sequences are (globally) aligned by writing them across a page in two rows. Identical (or similar) characters are placed in the same column and called *matches*. Non-identical characters are either placed in the same column as a *mismatch*, or they are opposite to a *gap* in the other sequence. Also, we disallow columns that consist of gaps only.

Two strings: $\longrightarrow$ Alignment:

```
IMISSMISSISSIPPI          I-MISSMISSISSIPPI-
                            |||| ||    ||||
MYMISSISAHIPPIE           MYMISS-ISA-H-IPPIE
```

another alignment:

```
IMISSMISSIS-SIPPI-
 |   ||||||  ||||
-M--YMISSISAHIPPIE
```

Sequence alignment (formally)

One way to formalize pairwise sequence alignment is as follows: We are given two sequences $x = (x_1, x_2, \dots, x_m)$ and $y = (y_1, y_2, \dots, y_n)$ over an alphabet Σ. Let $\texttt{-} \notin \Sigma$ be the *gap symbol*, also called *space*. Let $h\colon (\Sigma \cup \{\texttt{-}\})^* \to \Sigma^*$ be the mapping that removes all gap symbols from a sequence over the alphabet $\Sigma \cup \{\texttt{-}\}$.

For example, $h(\texttt{MYMISS-ISAH-IPPIE}) = \texttt{MYMISSISAHIPPIE}$.

Then a global alignment of x and y is a pair of sequences x', y' such that

$$h(x') = x, \quad h(y') = y, \quad |x'| = |y'|, \quad \text{and} \quad (x'_i, y'_i) \neq (\texttt{-}, \texttt{-}) \text{ for all } i.$$

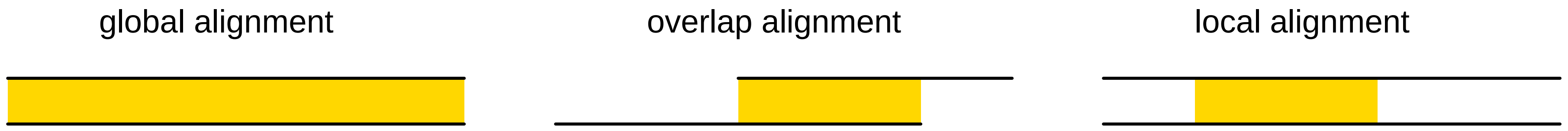
(Here $|\cdot|$ denotes length.)

More terminology

- *Match* - same (or similar) letter in both rows
- *Mismatch* - different letters in both rows
- *Insertion* - the letter opposite to a space
- *Deletion* - the space opposite to a letter
- *Indel* - a column containing a space

More terminology (2)

Depending on the input data, there are a number of different variants of alignment that are considered, among them *global alignment*, *overlap alignment*, and *local alignment*.



In an overlap alignment, we do not charge the end gaps (hence it is also called end-gap free alignment).

A local alignment is the same as a global alignment of two substrings of the sequences.

Finding alignments

How many alignments are there? Answer: many (blackboard)

How can we find a good pairwise alignment?

Why not simply *visualize* the data?

Dot plots

We can draw a matrix spanned up by two sequences and place a dot in each cell for which the correspondig symbols match. Stretches of matching symbols will show up as diagonal lines this way.

```
  IMISSMISSISSIPPI
M  *   *
Y
M  *   *
I * *   *  *  *  *
S    **  ** **
S    **  ** **
I * *   *  *  *  *
S    **  ** **
A
H
I * *   *  *  *  *
P              **
P              **
I * *   *  *  *  *
E
```

Dot plots (2)

To obtain cleaner pictures, a *window size* w and a *stringency* s are used: A dot is only drawn at point (x, y) if within w positions around it there are at least s matches.

	I	M	I	S	S	M	I	S	S	I	S	I	P	P	I
M	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
I	1	0	1	0	0	0	1	0	0	1	0	1	0	0	1
S	0	0	0	1	1	0	0	1	1	0	1	0	0	0	0
S	0	0	0	1	1	0	0	1	1	0	1	0	0	0	0
I	1	0	1	0	0	0	1	0	0	1	0	1	0	0	1
S	0	0	0	1	1	0	0	1	1	0	1	0	0	0	0
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	1	0	1	0	0	0	1	0	0	1	0	1	0	0	1
P	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
P	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
I	1	0	1	0	0	0	1	0	0	1	0	1	0	0	1
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

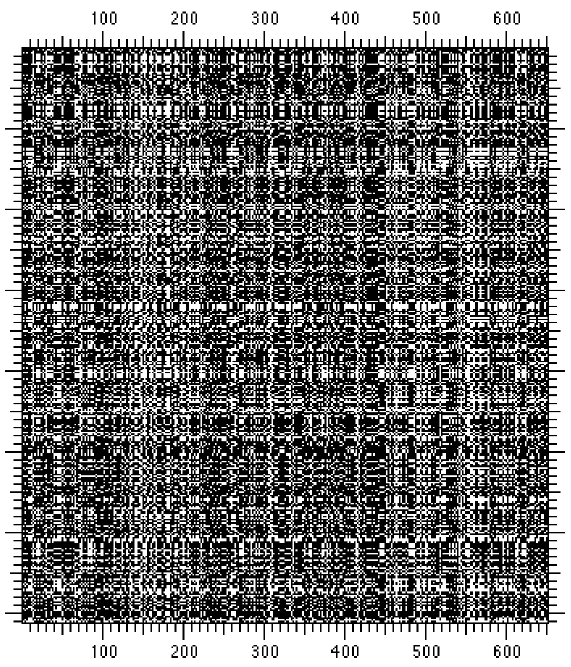
$w = 1$, $s = 1$

	I	M	I	S	S	M	I	S	S	I	S	I	P	P	I
M	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
Y	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0
M	0	2	0	0	0	2	0	0	1	0	1	0	0	1	0
I	1	0	3	1	0	0	3	1	0	2	0	1	0	0	1
S	0	1	1	3	1	0	1	3	1	1	2	0	1	0	0
S	0	1	0	1	2	2	0	1	3	1	2	1	0	1	0
I	1	0	2	1	1	1	2	1	1	3	0	2	0	0	1
S	0	1	0	2	1	0	0	2	1	0	2	0	1	0	0
A	0	0	0	0	1	1	0	0	1	1	0	1	0	0	0
H	0	1	0	0	0	1	0	0	1	0	1	0	0	1	0
I	1	0	1	0	0	0	1	0	0	1	0	2	1	0	1
P	0	1	0	1	0	0	0	1	0	0	1	1	3	1	0
P	0	1	0	0	0	1	0	0	1	0	1	0	1	3	1
I	1	0	1	0	0	0	1	0	0	1	0	1	0	1	2
E	0	1	0	1	0	0	0	1	0	0	1	0	1	0	0

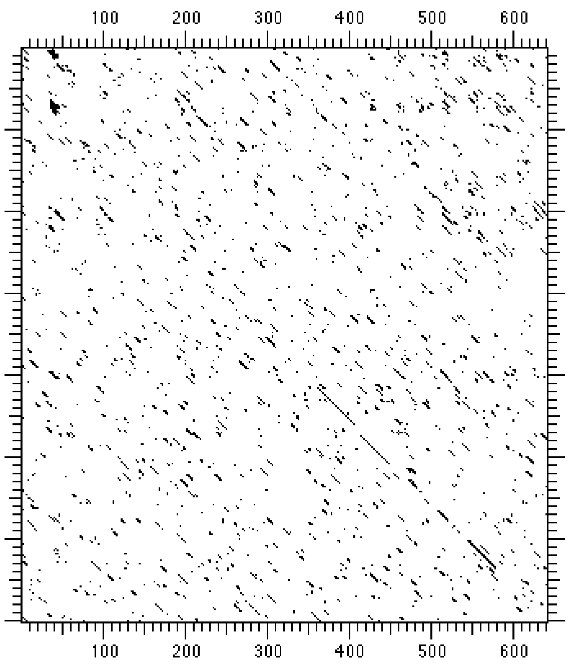
$w = 3$, $s = 3$

Dot plots (3)

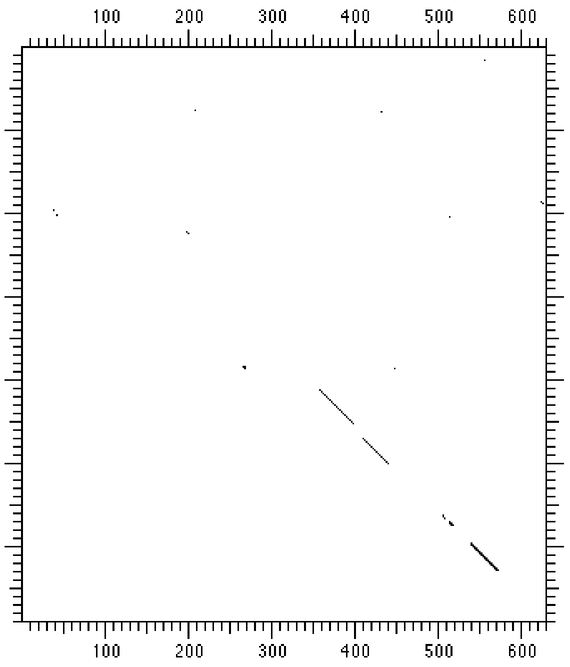
Here is a biological example:



$w = 1$, $s = 1$



$w = 11$, $s = 7$



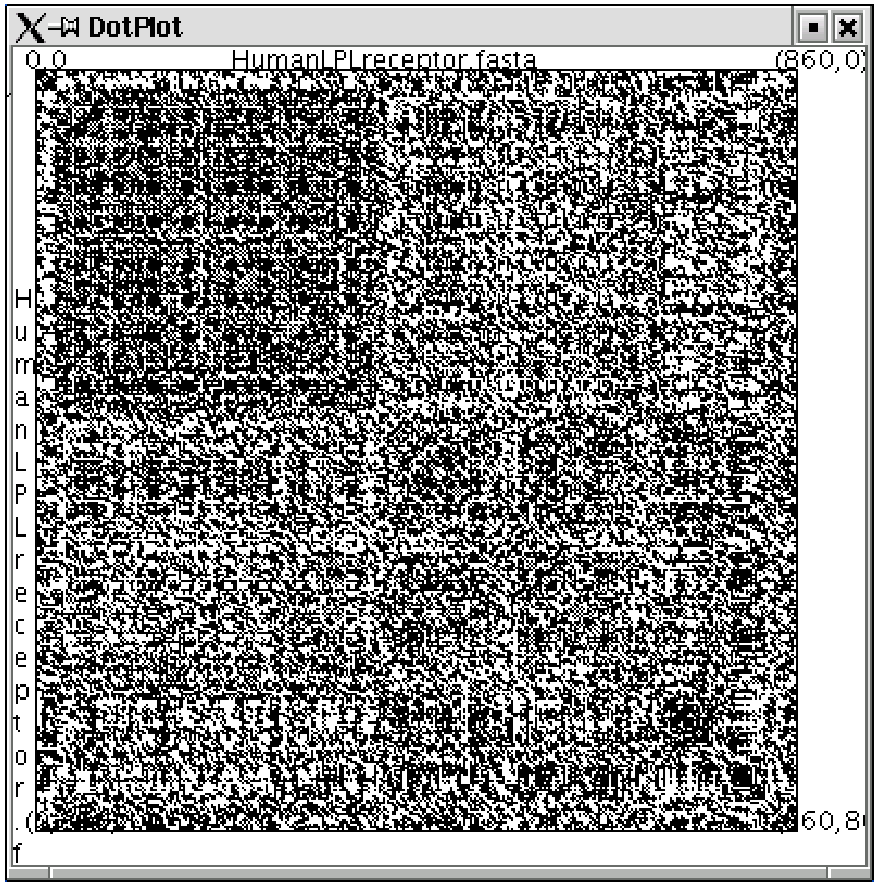
$w = 23$, $s = 15$

Tools on the web to play around with:

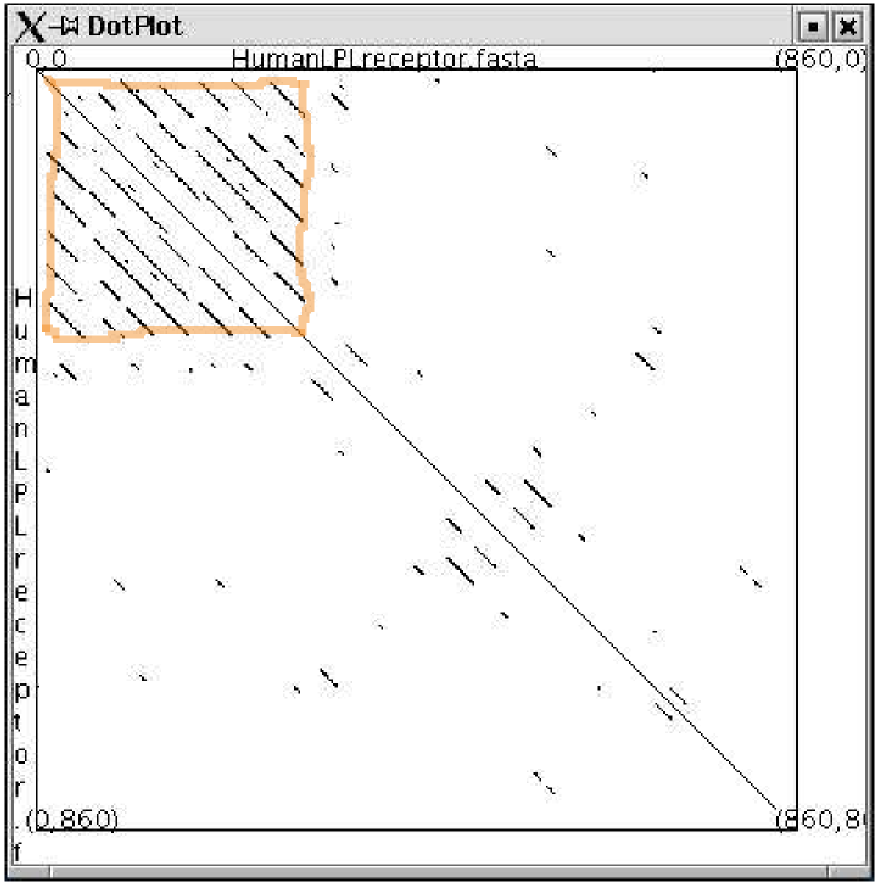
- http://www.isrec.isb-sib.ch/java/dotlet/Dotlet.html
- http://arbl.cvmbs.colostate.edu/molkit/dnadot/

Repeat detection using dot plots

Here is another one. These dot plots of the human LDL receptor (protein sequence) against itself reveal many *repeats* in the first 300 positions.



$w = 1, s = 1$



$w = 23, s = 7$

How to score it?

So we *see* there is an alignment, but do we really *have* it, written in two rows?

To come up with an algorithm, we need a formal concept when an alignment is “good”, i.e., *significant*. Let us have a look at some good and bad examples before we formally introduce a *scoring scheme*.

Significance of alignments

In the alignments below, the middle row contains a letter for identical amino acids, and a + if the amino acids are similar.

1. An alignment between very similar human alpha- and beta hemoglobins:

```
HBA_HUMAN  GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKL
           G+ +VK+HGKKV  A+++++AH+D++ +++++LS+LH  KL
HBB_HUMAN  GNPKVKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKL
```

2. Plausible alignment to leghaemoglobin from yellow lupin:

```
HBA_HUMAN  GSAQVKGHGKKVADALTNAVAHV---D--DMPNALSALSDLHAHKL
           ++ ++++H+ KV   + +A  ++          +L+ L+++H+ K
LGB2_LUPLU NNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKG
```

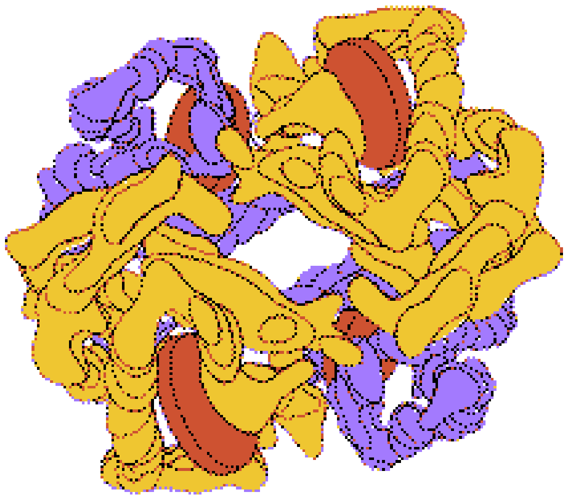
3. A spurious high-scoring alignment of human alpha globin to a nematode glutathione S-transferase homologue:

```
HBA_HUMAN  GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSD----LHAHKL
           GS+ + G +   +D L  ++ H+ D+  A +AL D    ++AH+
F11G11.2   GSGYLVGDSLTFVDLL--VAQHTADLLAANAALLDEFPQFKAHQE
```

Significance of alignments (2)

1. Alignment between very similar human alpha and beta hemoglobins:

```
HBA_HUMAN   GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKL
            G+ +VK+HGKKV  A+++++AH+D++ +++++LS+LH  KL
HBB_HUMAN   GNPKVKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKL
```



In (1), there are many positions at which the two corresponding residues are identical. Many others are functionally conservative. E.g., the `D`-`E` pair towards the end: both negatively charged amino acids.

Significance of alignments (3)

2. Plausible alignment to leghaemoglobin from yellow lupin:

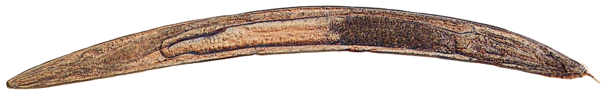
```
HBA_HUMAN  GSAQVKGHGKKVADALTNAVAHV---D--DMPNALSALSDLHAHKL
           ++ ++++H+ KV   + +A  ++          +L+ L+++H+ K
LGB2_LUPLU NNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKG
```

In (2), we also see a biologically meaningful alignment, as it is known that the two proteins are evolutionarily related, have the same 3D structure and both have the same function. However, there are many fewer identities and gaps have been introduced in the sequences.

Significance of alignments (4)

3. A spurious high-scoring alignment of human alpha globin to a nematode glutathione S-transferase homologue:

```
HBA_HUMAN   GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSD----LHAHKL
            GS+ + G +   +D L  ++ H+ D+  A +AL D    ++AH+
F11G11.2    GSGYLVGDSLTFVDLLVAQHTADLL--AANAALLDEFPQFKAHQE
```



In (3), we see an alignment with a similar number of identities or conservative changes. However, this is a spurious alignment between two proteins that have completely different structure and function.

Scoring schemes

The basic mutational processes are *substitutions*, *insertions* and *deletions*. Substitutions give rise to *mismatches*. Insertions and deletions give rise to *gaps*.

The *total score* assigned to an alignment is the

1. sum of terms for matches and mismatches, plus the
2. sum of terms for gaps (i.e., indels).

Formally we can treat the space character just like any other. Then we obtain the following additive scoring scheme:

The score of an alignment (x', y') is

$$\sum_i \delta(x'_i, y'_i),$$

where $\delta\colon (\Sigma \cup \{\texttt{-}\})^2 \to \mathbb{R}$ is a *score matrix*.

Hamming distance and Levenshtein distance

So how should we choose the **score matrix** δ?

In the simplest case, we do not allow any gaps at all and charge all mismatches at unit cost. This is called the *Hamming distance.* The alignment is forced to be on one diagonal of the dot plot.

If we charge all insertions, deletions, and mismatches at unit cost, we obtain the *Levenshtein distance.* This measure is also called the *edit distance*, because it counts the number of elementary edit operations needed to transform one sequence into the other.

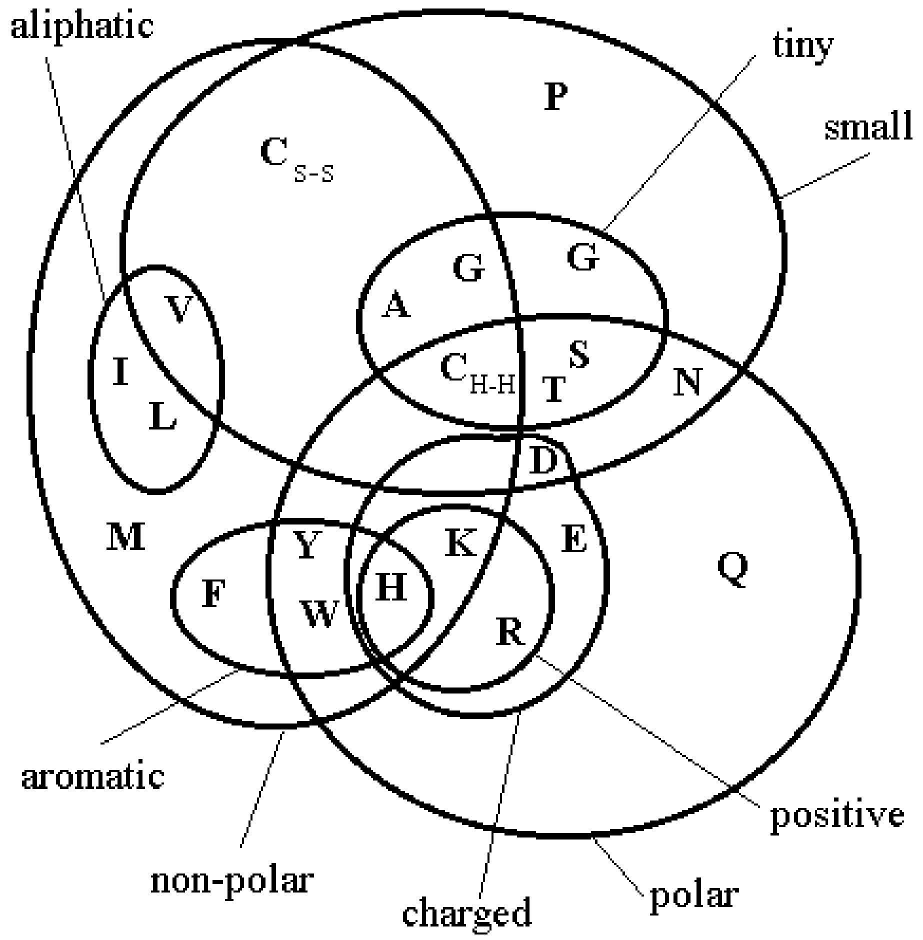
Both scoring schemes have applications in biological sequence analysis, especially for nucleic acids . . . but . . . generally it is better to

1. distinguish the type of a mismatch, and

2. take the length of consecutive gaps into account.

Classification of amino acids

Amino acids can be grouped according to chemical properties.



The BLOSUM50 Matrix

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X	*
A	5	-2	-1	-2	-1	-1	-1	0	-2	-1	-2	-1	-1	-3	-1	1	0	-3	-2	0	-2	-1	-1	-5
R	-2	7	-1	-2	-4	1	0	-3	0	-4	-3	3	-2	-3	-3	-1	-1	-3	-1	-3	-1	0	-1	-5
N	-1	-1	7	2	-2	0	0	0	1	-3	-4	0	-2	-4	-2	1	0	-4	-2	-3	4	0	-1	-5
D	-2	-2	2	8	-4	0	2	-1	-1	-4	-4	-1	-4	-5	-1	0	-1	-5	-3	-4	5	1	-1	-5
C	-1	-4	-2	-4	13	-3	-3	-3	-3	-2	-2	-3	-2	-2	-4	-1	-1	-5	-3	-1	-3	-3	-2	-5
Q	-1	1	0	0	-3	7	2	-2	1	-3	-2	2	0	-4	-1	0	-1	-1	-1	-3	0	4	-1	-5
E	-1	0	0	2	-3	2	6	-3	0	-4	-3	1	-2	-3	-1	-1	-1	-3	-2	-3	1	5	-1	-5
G	0	-3	0	-1	-3	-2	-3	8	-2	-4	-4	-2	-3	-4	-2	0	-2	-3	-3	-4	-1	-2	-2	-5
H	-2	0	1	-1	-3	1	0	-2	10	-4	-3	0	-1	-1	-2	-1	-2	-3	2	-4	0	0	-1	-5
I	-1	-4	-3	-4	-2	-3	-4	-4	-4	5	2	-3	2	0	-3	-3	-1	-3	-1	4	-4	-3	-1	-5
L	-2	-3	-4	-4	-2	-2	-3	-4	-3	2	5	-3	3	1	-4	-3	-1	-2	-1	1	-4	-3	-1	-5
K	-1	3	0	-1	-3	2	1	-2	0	-3	-3	6	-2	-4	-1	0	-1	-3	-2	-3	0	1	-1	-5
M	-1	-2	-2	-4	-2	0	-2	-3	-1	2	3	-2	7	0	-3	-2	-1	-1	0	1	-3	-1	-1	-5
F	-3	-3	-4	-5	-2	-4	-3	-4	-1	0	1	-4	0	8	-4	-3	-2	1	4	-1	-4	-4	-2	-5
P	-1	-3	-2	-1	-4	-1	-1	-2	-2	-3	-4	-1	-3	-4	10	-1	-1	-4	-3	-3	-2	-1	-2	-5
S	1	-1	1	0	-1	0	-1	0	-1	-3	-3	0	-2	-3	-1	5	2	-4	-2	-2	0	0	-1	-5
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	2	5	-3	-2	0	0	-1	0	-5
W	-3	-3	-4	-5	-5	-1	-3	-3	-3	-3	-2	-3	-1	1	-4	-4	-3	15	2	-3	-5	-2	-3	-5
Y	-2	-1	-2	-3	-3	-1	-2	-3	2	-1	-1	-2	0	4	-3	-2	-2	2	8	-1	-3	-2	-1	-5
V	0	-3	-3	-4	-1	-3	-3	-4	-4	4	1	-3	1	-1	-3	-2	0	-3	-1	5	-4	-3	-1	-5
B	-2	-1	4	5	-3	0	1	-1	0	-4	-4	0	-3	-4	-2	0	0	-5	-3	-4	5	2	-1	-5
Z	-1	0	0	1	-3	4	5	-2	0	-3	-3	1	-1	-4	-1	0	-1	-2	-2	-3	2	5	-1	-5
X	-1	-1	-1	-1	-2	-1	-1	-2	-1	-1	-1	-1	-1	-2	-2	-1	0	-3	-1	-1	-1	-1	-1	-5
*	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	1

Gap penalties

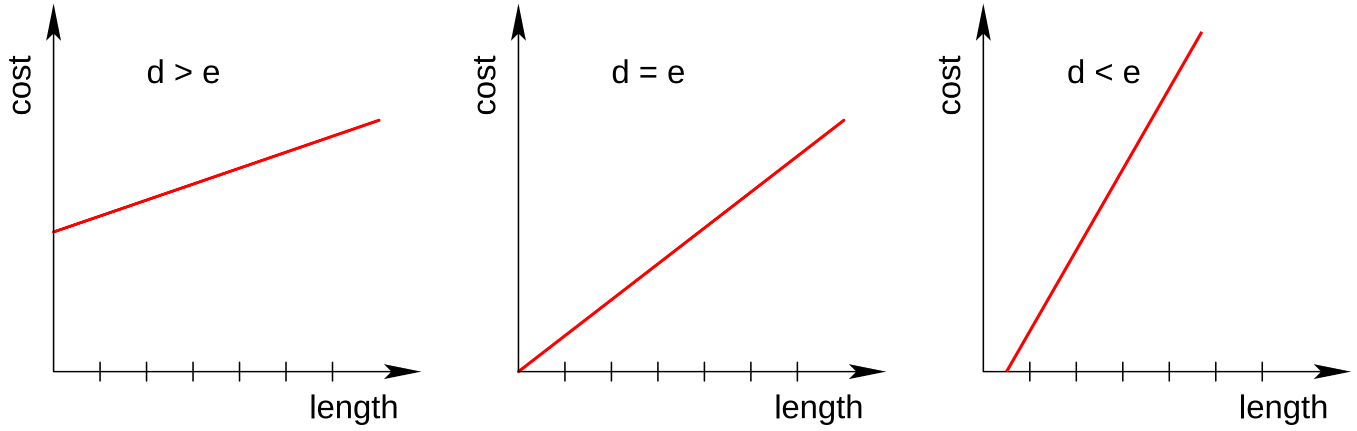
Gaps in an alignment are undesirable and thus penalized. In its simplest form, the cost associated with a gap of length g is given by a *linear* score,

$$\gamma(g) = -gd\,.$$

An *affine* score, however,

$$\gamma(g) = -d - (g-1)e$$

often produces better results. Here d is the *gap open* penalty and e is the *gap extension* penalty.



Gap penalties (2)

Usually one sets $e < d$, i.e., there is a large penalty for opening a gap, but a smaller penalty for extending it. Then affine gap costs favor alignments with fewer but larger gaps. For example, an intron.

Using linear gap penalties:

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

Using affine gap penalties:

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

The case $d < e$ is sometimes used when comparing output of DNA sequencing machines. There it happens frequently that single bases are left out near the end of a read.

Remarks on scoring schemes

1. The scoring schemes we have seen so far are based solely on primary structure. This is a reasonable approximation especially for DNA. For RNA, one observes that bases which are coupled by secondary structure are highly correlated. But even for proteins the primary structure can tell us a lot.

2. We will not explain the probabilistic background of the additive scoring scheme (in this lecture), but simply take it as granted. Thus we will not explain, e.g., how the BLOSUM50 matrix was derived from experimental data, assuming a certain model of evolution, etc.

3. The alignment algorithms we describe next can be generalized to affine gap cost, but technically they become a bit more complicated, so we will stick to linear gap costs at the beginning.

Algorithms for pairwise sequence alignment

Alignment algorithms

Given a scoring scheme and two input sequences, we need an algorithm to compute the highest-scoring alignment of the sequences.

We will discuss alignment algorithms based on *dynamic programming*. Dynamic programming algorithms play a central role in computational sequence analysis. They are guaranteed to find the highest-scoring alignment.

Note: For large sequences exact "DP algorithms" can be too slow and *heuristics* (such as BLAST, FASTA, MUMMER, QUASAR,. . .) are then used which perform very well in most cases, but will miss the best alignment for some sequence pairs.

Global alignment: Needleman-Wunsch algorithm

(Saul Needleman and Christian Wunsch, 1970; improved by Peter Sellers, 1974)

Consider the problem of finding the optimal *global alignment* of two sequences x and y. The Needleman-Wunsch algorithm is a “dynamic program” that solves this problem. What does this mean?

The **idea** is to build up a table of optimal alignments for all pairs of prefixes of x and y $\ldots$ using (already known) optimal alignments of shorter prefixes of x and y.

We are given two sequences $x = (x_1, x_2, \ldots, x_m)$ and $y = (y_1, y_2, \ldots, y_n)$. We will compute a matrix, usually called *dynamic programming table*,

$$F : \{0, 1, 2, \ldots, m\} \times \{0, 1, 2, \ldots, n\} \to \mathbb{R}$$

in which $F(i, j)$ equals the best score of an alignment of the two prefixes $(x_1, x_2, \ldots, x_i)$ and $(y_1, y_2, \ldots, y_j)$.

Global alignment (2)

Outline of the algorithm: We fill the table F recursively, bottom-up*. We start with initial cases like $F(0,0) = 0$ and then compute each $F(i,j)$ from $F(i-1, j-1)$, $F(i-1, j)$ and $F(i, j-1)$:

$$\begin{array}{c|ccccccccc}
 & & x_1 & x_2 & \dots & x_{i-1} & & x_i & \dots & x_m \\
\hline
 & F(0,0) & F(1,0) & F(0,2) & & & & \vdots & & \\
y_1 & F(0,1) & & & & & & \vdots & & \\
y_2 & F(0,2) & & & & & & \vdots & & \\
\vdots & & & & & & & \vdots & & \\
y_{j-1} & & & & & F(i-1,j-1) & & F(i,j-1) & & \\
 & & & & & & \nwarrow & \uparrow & & \\
y_j & \dots & \dots & \dots & \dots & F(i-1,j) & \leftarrow & F(i,j) & & \\
\vdots & & & & & & & & & \\
y_n & & & & & & & & &
\end{array}$$

This is applied until the whole matrix F is filled with values. Then $F(m,n)$ is the score of the best global alignment.

*This corresponds to step 2 on the DP paradigm slides

Global alignment (3)

Why *can* we apply such a **recursion**?

There are three ways how the last column of an alignment of $(x_1, x_2, \ldots, x_i)$ and $(y_1, y_2, \ldots, y_j)$ can look like:

x_i aligns to y_j:

I	G	A	x_i
L	G	V	y_j

x_i aligns to a gap:

A	I	G	A	x_i
G	V	y_j	-	-

y_j aligns to a gap:

G	A	x_i	-	-
S	L	G	V	y_j

We obtain $F(i,j)$ as the largest score that arises from one of these cases*:

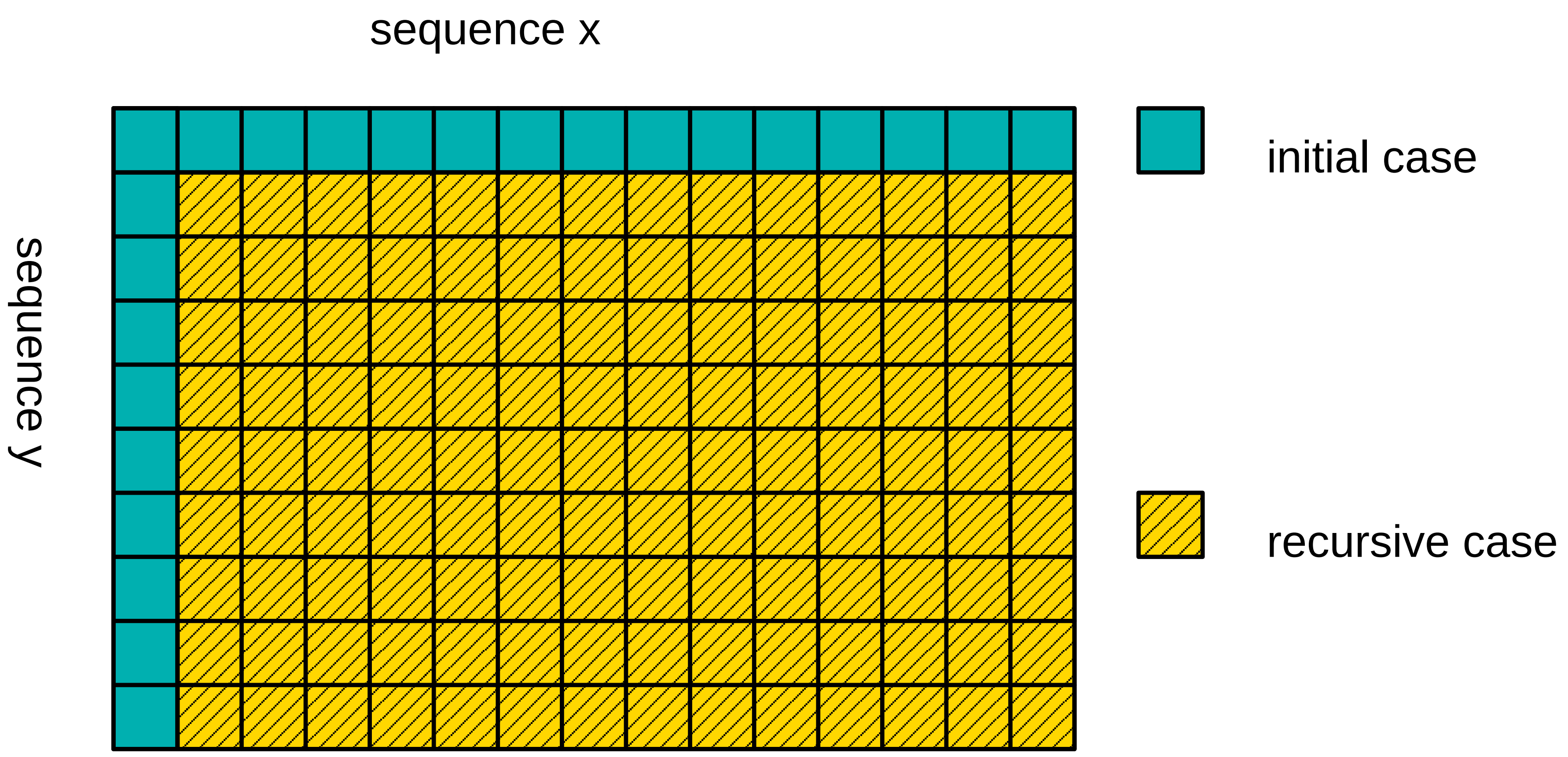
$$F(i,j) := \max \begin{cases} F(i-1, j-1) + s(x_i, y_j) \\ F(i-1, j) - d \\ F(i, j-1) - d \end{cases}$$

*This corresponds to step 1 on the DP paradigm slides

Global alignment (4)

To complete the description of the recursion, we need to set the **initial values** on the upper and the left boundary, $F(i, 0)$ and $F(0, j)$:

We set $F(i, 0) = id$ for $i = 0, 1, \ldots, m$ and $F(0, j) = jd$ for $j = 0, 1, \ldots, n$.



Global alignment (5)

The final value $F(m,n)$ is the *score* of the best global alignment between x and y.

To obtain an *alignment* corresponding to this score, we still need to find the path of choices that has led the recursion to the final score. This is called a *traceback*[*]

However this is actually easy, as we only have to store one of the symbols

$$T(i,j) \in \{\leftarrow, \nwarrow, \uparrow\}$$

(or a subset thereof) whenever we assign a value to a “DP entry” $F(i,j)$.

[*]This corresponds to step 3 on the DP paradigm slides.

Needleman-Wunsch algorithm

```
Input: two sequences $x$ and $y$
Output: optimal alignment and score $\alpha$
Initialization:
Set $F(0,0) := 0$
Set $F(i,0) := -id$ and $T(i,0) := (i-1,0)$ for all $i = 1,2,\dots,m$
Set $F(0,j) := -jd$ and $T(0,j) := (0,j-1)$ for all $j = 1,2,\dots,n$
Recurrence:
for $i = 1,2,\dots,m$ do:
    for $j = 1,2,\dots,n$ do:
        Set $F(i,j) := \max \begin{cases} F(i-1,j-1) + s(x_i,y_i) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases}$
        Set backtrace $T(i,j)$ to the maximizing pair $(i',j')$ (encoded as $\in \{\leftarrow, \nwarrow, \uparrow\}$)
The best score is $\alpha := F(m,n)$
Set $(i,j) := (m,n)$
Traceback:
repeat
    if $T(i,j) = (i-1,j-1)$ print $\binom{x_{i-1}}{y_{j-1}}$
    else if $T(i,j) = (i-1,j)$ print $\binom{x_{i-1}}{-}$ else print $\binom{-}{y_{j-1}}$
    Set $(i,j) := T(i,j)$
until $(i,j) = (0,0)$.
```

An example of global alignment

We will use two short amino acid sequences for illustration:

`HEAGAWGHEE` and `PAWHEAE`.

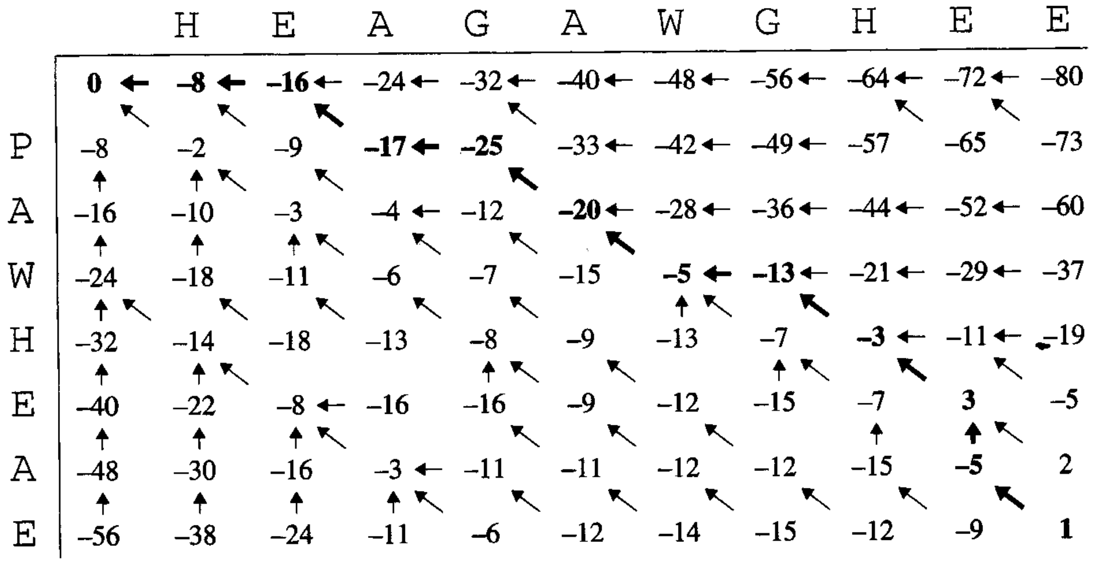
To score the alignment we will use the BLOSUM50 matrix and a gap cost of $d = 8$.

Example (2)

$d = 8$

BLOSUM50 values:

	H	E	A	G	A	W	G	H	E	E
P	-2	-1	-1	-2	-1	-4	-2	-2	-1	-1
A	-2	-1	**5**	0	**5**	-3	0	-2	-1	-1
W	-3	-3	-3	-3	-3	**15**	-3	-3	-3	-3
H	**10**	0	-2	-2	-2	-3	-3	**10**	0	0
E	0	**6**	-1	-3	-1	-3	-3	0	**6**	**6**
A	-2	-1	**5**	0	**5**	-3	0	-2	-1	-1
E	0	**6**	-1	-3	-1	-3	-3	0	**6**	**6**



		H	E	A	G	A	W	G	H	E	E
	0	**–8**	**–16**	–24	–32	–40	–48	–56	–64	–72	–80
P	–8	–2	–9	**–17**	**–25**	–33	–42	–49	–57	–65	–73
A	–16	–10	–3	–4	–12	**–20**	–28	–36	–44	–52	–60
W	–24	–18	–11	–6	–7	–15	**–5**	**–13**	–21	–29	–37
H	–32	–14	–18	–13	–8	–9	–13	–7	**–3**	–11	–19
E	–40	–22	–8	–16	–16	–9	–12	–15	–7	**3**	–5
A	–48	–30	–16	–3	–11	–11	–12	–12	–15	**–5**	2
E	–56	–38	–24	–11	–6	–12	–14	–15	–12	–9	**1**

```
HEAGAWGHE-E
--P-AW-HEAE
```

DP matrix:

Durbin *et al.* (1998)

Needleman-Wunsch Java applet

There is a nice Java applet illustrating the NW algorithm on the web:

`http://lectures.molgen.mpg.de/PracticalSection/AliApplet/index.html`

Complexity of the Needleman-Wunsch algorithm

We need to store $(n+1)(m+1)$ numbers. Each number takes a constant number of calculations to compute: just three sums and a max.

Hence, the algorithm requires $O(nm)$ time and memory.

Equivalence of similarity and distance

For global alignments, there is a nice equivalence between *similarity scores* (where the goal is to maximize) and *distance scores* (where the goal is to minimize), due to Smith and Waterman:

Theorem.
Let $s_{\text{sim}}(x,y) \geq 0$ be a similarity score for $x, y \in \Sigma$ and $d_{\text{sim}}(\ell) \geq 0$ be a gap penalty for gaps of length ℓ. Let $\widehat{s} := \max_{x,y} s_{\text{sim}}(x,y)$. Define a distance score by

$$s_{\text{dist}}(x,y) := \widehat{s} - s_{\text{sim}}(x,y)$$

and let

$$d_{\text{dist}}(\ell) := \ell\frac{\widehat{s}}{2} + d_{\text{sim}}(\ell).$$

Then every maximum similarity global alignment with respect to $s_{\text{sim}}, d_{\text{sim}}$ is also a minimum distance global alignment with respect to $s_{\text{dist}}, d_{\text{dist}}$, and vice versa.

Proof. Blackboard.