The Needleman-Wunsch algorithm for sequence alignment

7th Melbourne Bioinformatics Course

Vladimir Likić, Ph.D.
e-mail: vlikic@unimelb.edu.au

Bio21 Molecular Science and Biotechnology Institute
The University of Melbourne

Outline of the talk

- Sequence comparison and sequence alignment.
- Types of sequence alignment.
- The scoring scheme, substitution matrices, gaps.
- The Needleman-Wunsch algorithm for global sequence alignment.

Sequence comparison

- To observe patterns of conservation (or variability).
- To find the common motifs present in both sequences.
- To assess whether it is likely that two sequences evolved from the same sequence.
- To find out which sequences from the database are similar to the sequence at hand.

Two routes for sequence comparison

- *dotplot* – visual, qualitative
- *sequence alignment* – exact and quantitative. Involves:
  1. Construction of the best alignment between the sequences.
  2. Assessment of the similarity from the alignment.
- There are three different types of sequence alignment:
  - Global alignment
  - Local alignment
  - Multiple sequence alignment

Global sequence alignment

- The best alignment over the entire length of two sequences
- Suitable when the two sequences are of similar length, with a significant degree of similarity throughout.
- Example:

```
SIMILARITY
PI-LLAR---
```

Local sequence alignment

- Involving stretches that are shorter than the entire sequences, possibly more than one.

- Suitable when comparing substantially different sequences, which possibly differ significantly in length, and have only a short patches of similarity.

- For example, the local alignment of `SIMILARITY` and `PILLAR`:

```
MILAR
ILLAR
```

Multiple sequence alignment

- Simultaneous alignment of more than two sequences.
- Suitable when searching for subtle conserved sequence patterns in a protein family, and when more than two sequences of the protein family are available.
- For example:

```
SIMILARITY
PI-LLAR---
--MOLARITY
```

Alignment "by eye"

- Consider the "best" alignment of `ATGGCGT` and `ATGAGT`

```
ATGGCGT
*** !**
ATG-AGT
```

- Intuitively we seek an alignment to maximize the number of residue-to-residue matches.

A mathematical framework

- *Sequence alignment is the establishment of residue-to-residue correspondence between two or more sequences such that the order of residues in each sequence is preserved*.
- A gap, which indicates a residue-to-nothing match, may be introduced in either sequence.
- A gap-to-gap match is meaningless and is not allowed.

The scoring scheme

- Give two sequences we need a number to associate with each possible alignment (i.e. the **alignment score** = goodness of alignment).

- The **scoring scheme** is a set of rules which assigns the alignment score to any given alignment of two sequences.

1. The scoring scheme is residue based: it consists of **residue substitution scores** (i.e. score for each possible residue alignment), plus penalties for gaps.
2. The alignment score is the sum of substitution scores and gap penalties.

A simple scoring scheme

- **Use +1 as a reward for a match, -1 as the penalty for a mismatch, and ignore gaps**

- The best alignment "by eye" from before:

```
ATGGCGT
ATG-AGT
```

score: $+1 + 1 + 1 + 0 - 1 + 1 + 1 = 4$

- An alternative alignment:

```
ATGGCGT
A-TGAGT
```

score: $+1 + 0 - 1 + 1 - 1 + 1 + 1 = 2$

The substitution matrix

- A concise way to express the residue substitution costs can be achieved with a $N \times N$ matrix (N is 4 for DNA and 20 for proteins).
- The substitution matrix for the simple scoring scheme:

	C	T	A	G
C	1	-1	-1	-1
T	-1	1	-1	-1
A	-1	-1	1	-1
G	-1	-1	-1	1

A better substitution matrix

- A, G are purines (pyrimidine ring fused to an imidazole ring), T, C are pyrimidines (one six-membered ring).
- Assume we believe that from evolutionary standpoint purine/pyrimidine mutations are less likely to occur compared to purine/purine (pyrimidine/pyrimidine) mutations. Can we capture this in a substitution matrix?

	C	T	A	G
C	2	1	-1	-1
T	1	2	-1	-1
A	-1	-1	2	1
G	-1	-1	1	2

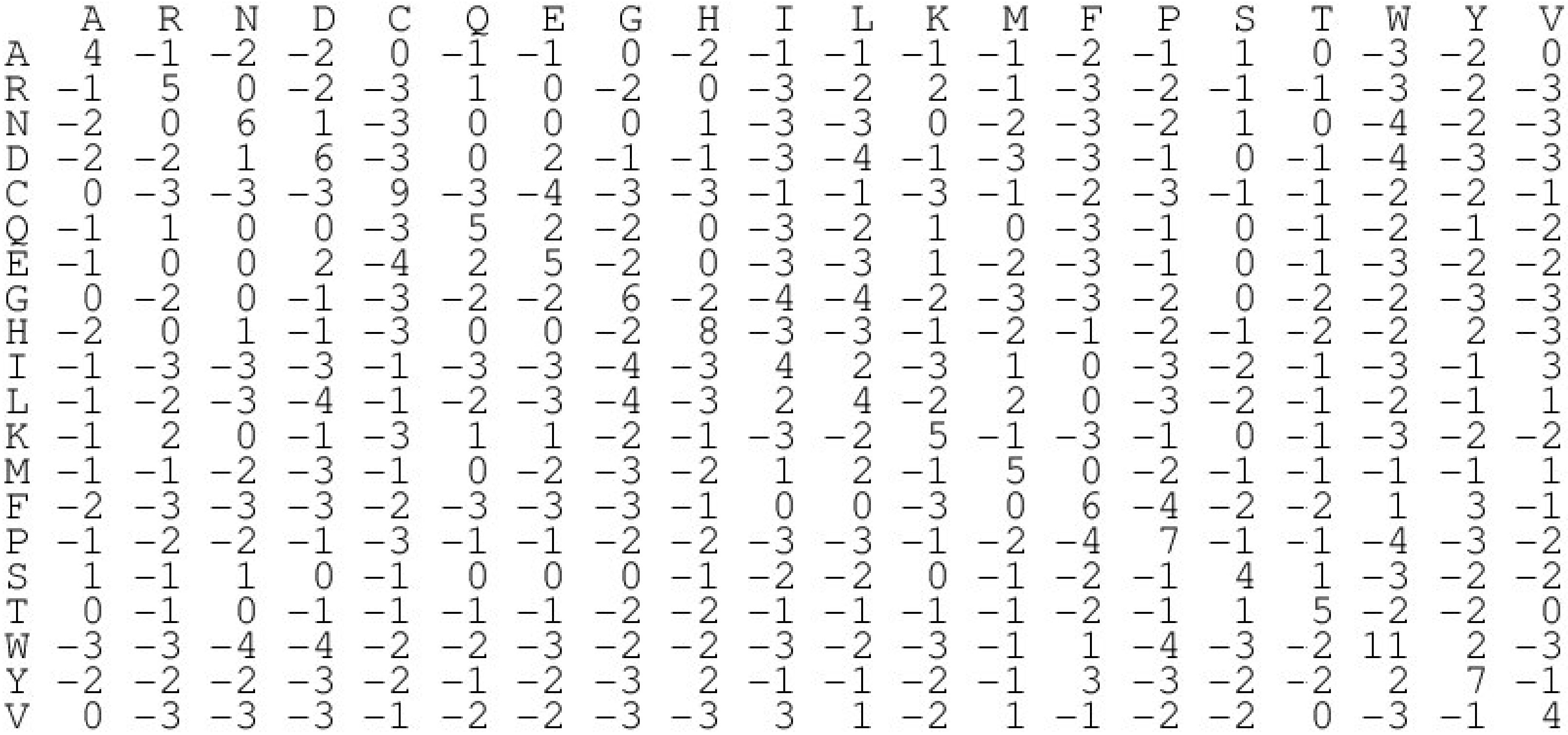
Protein substitution matrices

- Protein substitution matrices are significantly more complex than DNA scoring matrices.
- Proteins are composed of twenty amino acids, and physico-chemical properties of individual amino acids vary considerably.
- A protein substitution matrix can be based on any property of amino acids: size, polarity, charge, hydrophobicity.
- In practice the most important are **evolutionary substitution matrices**.

Evolutionary substitution matrices

- PAM (”point accepted mutation”) family
  - PAM250, PAM120, etc.
- BLOSUM (”Blocks substitution matrix”) family
  - BLOSUM62, BLOSUM50, etc.
- The substitution scores of both PAM and BLOSUM matrices are derived from the analysis of known alignments of closely related proteins.
- The BLOSUM matrices are newer and considered better.

BLOSUM62 substitution matrix



	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

Gaps

- So far we ignored gaps (amounts to gap penalty of 0)
- A gap corresponds to an insertion or a deletion of a residue
- A conventional wisdom dictates that the penalty for a gap must be several times greater than the penalty for a mutation. That is because a gap/extra residue
  - Interrupts the entire polymer chain
  - In DNA shifts the reading frame

Gap initiation and extension

- The conventional wisdom: the creation of a new gap should be strongly disfavored.
- However, once created insertions/deletions of chunks of more than one residue should be much less expensive (i.e. insertion of domains often occurs).
- A simple yet effective solution is **affine gap penalties**:

$$\gamma(n) = -o - (n-1)e$$

Affine gaps: a physical insight

- Affine gaps favor the alignment:

```
ATGTAGTGTATAGTACATGCA
ATGTAG-------TACATGCA
```

  Over the alignment:

```
ATGTAGTGTATAGTACATGCA
ATGTA--G--TA---CATGCA
```

- Exactly what we want from the biological viewpoint.

The alignment score with BLOSUM62

- Consider two alternative alignments of ANRGDFS and ANREFS with the gap opening penalty of 10:

```
ANRGDFS
ANR-EFS
```

score: $4+6+5-10+2+6+4=17$

```
ANRGDFS
ANRE-FS
```

score: $4+6+5-2-10+6+4=13$

- The scoring scheme provides us with the quantitative measure of how good is some alignment relative to alternative alignments. **However the scoring scheme does not tell us how to find the best alignment.**

How do we find the best alignment?

- Brute-force approach:
  - Generate the list all possible alignments between two sequences, score them
  - Select the alignment with the best score
- The number of possible global alignments between two sequences of length N is

$$\frac{2^{2N}}{\sqrt{\pi N}}$$

For two sequences of 250 residues this is $\sim 10^{149}$

The Needleman-Wunsch algorithm

- A smart way to reduce the massive number of possibilities that need to be considered, yet still guarantees that the best solution will be found (Saul Needleman and Christian Wunsch, 1970).
- The basic idea is to build up the best alignment by using optimal alignments of smaller subsequences.
- The Needleman-Wunsch algorithm is an example of dynamic programming, a discipline invented by Richard Bellman (an American mathematician) in 1953!

How does dynamic programming work?

- A divide-and-conquer strategy:
  - Break the problem into smaller subproblems.
  - Solve the smaller problems optimally.
  - Use the sub-problem solutions to construct an optimal solution for the original problem.

- Dynamic programming can be applied only to problems exhibiting the properties of *overlapping subproblems*. Examples include
  - Trevelling salesman problem
  - Finding the best chess move

The mathematics

- A matrix $D(i,j)$ indexed by residues of each sequence is built recursively, such that

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

subject to a boundary conditions. $s(i,j)$ is the substitution score for residues i and j, and g is the gap penalty.

A walk-through: an overview

- We consider all possible pairs of residue from two sequences (this gives rise to a 2D matrix representation).

- We will have two matrices: the score matrix and traceback matrix.

- The Needleman-Wunsch algorithm consists of three steps:
  1. Initialisation of the score matrix
  2. Calculation of scores and filling the traceback matrix
  3. Deducing the alignment from the traceback matrix

Consider the simple example

- The alignment of two sequences SEND and AND with the BLOSUM62 substitution matrix and gap opening penalty of 10 (no gap extension):

```
SEND
-AND score:  +1
A-ND score:  +3 ← the best
AN-D score:  -3
AND- score:  -8
```

The score and traceback matrices

- The cells of the score matrix are labelled $C(i,j)$ where $i = 1, 2, ..., N$ and $j = 1, 2, ..., M$

		S	E	N	D
	C(1,1)	C(1,2)	C(1,3)	C(1,4)	C(1,5)
A	C(2,1)	C(2,2)	C(2,3)	C(2,4)	C(2,5)
N	C(3,1)	C(3,2)	C(3,3)	C(3,4)	C(3,5)
D	C(4,1)	C(4,2)	C(4,3)	C(4,4)	C(4,5)

	S	E	N	D

Initialization

- The first row and the first column of the score and traceback matrices are filled during the initialization.

		S	E	N	D
	0	−10	−20	−30	−40
A	−10				
N	−20				
D	−30				

		S	E	N	D
	done	left	left	left	left
A	up				
N	up				
D	up				

Scoring

- The score matrix cells are filled by row starting from the cell $C(2,2)$

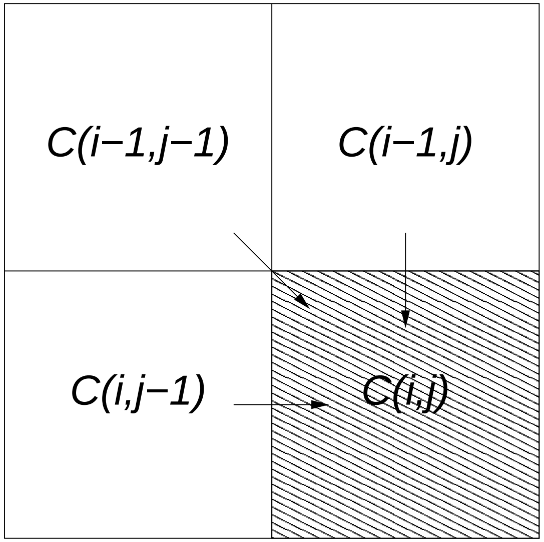
- The score of any cell $C(i,j)$ is the maximum of:

$$\begin{aligned} q_{diag} &= C(i-1,j-1) + S(i,j) \\ q_{up} &= C(i-1,j) + g \\ q_{left} &= C(i,j-1) + g \end{aligned}$$

where $S(i,j)$ is the substitution score for letters i and j, and g is the gap penalty.

Scoring– a pictorial representation

- The value of the cell $C(i,j)$ depends only on the values of the immediately adjacent northwest diagonal, up, and left cells:



The Needleman-Wunsch progression

- The first step is to calculate the value of $C(2,2)$:

		S	E	N	D
	0	−10	−20	−30	−40
A	−10	**?**			
N	−20				
D	−30				

	S	E	N	D
done	left	left	left	left
up	**?**			
up				
up				

The value of $C(2,2)$

- The calculation for the cell $C(2,2)$:

$$\begin{aligned} q_{diag} &= C(1,1) + S(S,A) = 0 + 1 = 1 \\ q_{up} &= C(1,2) + g = -10 + (-10) = -20 \\ q_{left} &= C(2,1) + g = -10 + (-10) = -20 \end{aligned}$$

Where $C(1,1)$, $C(1,2)$, and $C(2,1)$ are read from the score matrix, and $S(S,A)$ is the score for the $S \leftrightarrow A$ taken from the BLOSUM62 matrix.

Filling the score and traceback matrices

- For the score matrix $C(2,2) = q_{diag}$ which is 1. The corresponding cell of the traceback matrix is "diag":

		S	E	N	D
	0	−10	−20	−30	−40
A	−10	**1**			
N	−20				
D	−30				

		S	E	N	D
	done	left	left	left	left
A	up	**diag**			
N	up				
D	up				

The progression is recursive

- The next step is to calculate $C(2,3)$:

		S	E	N	D
	0	−10	−20	−30	−40
A	−10	**1**	**?**		
N	−20				
D	−30				

		S	E	N	D
	done	left	left	left	left
A	up	**diag**	**?**		
N	up				
D	up				

The value of $C(2,3)$

- The calculation for the cell $C(2,3)$

$$
\begin{aligned}
q_{diag} &= C(1,2) + S(E,A) = -10 + -1 = -11 \\
q_{up} &= C(1,3) + g = -20 + (-10) = -30 \\
q_{left} &= C(2,2) + g = 1 + (-10) = -9
\end{aligned}
$$

- Thus $C(2,3) = -9$ and the corresponding cell of the traceback matrix is "left".

The final score and traceback matrices

- After all cells are filled, the score and traceback matrices are:

		S	E	N	D
	0	−10	−20	−30	−40
A	−10	1	−9	−19	−29
N	−20	−9	−1	−3	−13
D	−30	−19	−11	2	3

	S	E	N	D
done	left	left	left	left
up	diag	left	left	left
up	diag	diag	diag	left
up	up	diag	diag	diag

The traceback

- Traceback = the process of deduction of the best alignment from the traceback matrix.

- The traceback always begins with the last cell to be filled with the score, i.e. the bottom right cell.

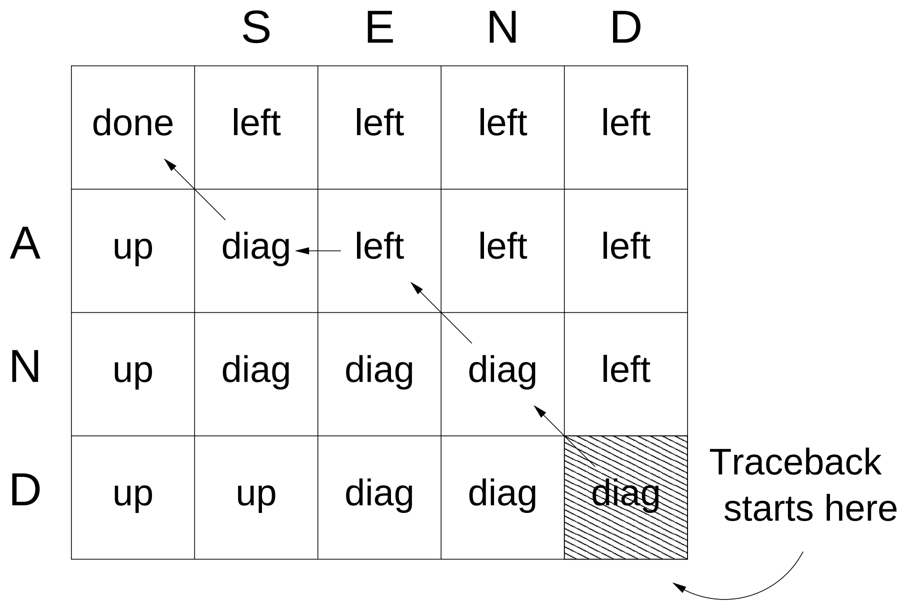
- One moves according to the traceback value written in the cell.

- There are three possible moves: diagonally (toward the top-left corner of the matrix), up, or left.

- The traceback is completed when the first, top-left cell of the matrix is reached ("done" cell).

The traceback path

- The traceback performed on the completed traceback matrix:



		S	E	N	D
	done	left	left	left	left
A	up	diag	left	left	left
N	up	diag	diag	diag	left
D	up	up	diag	diag	diag

Traceback starts here

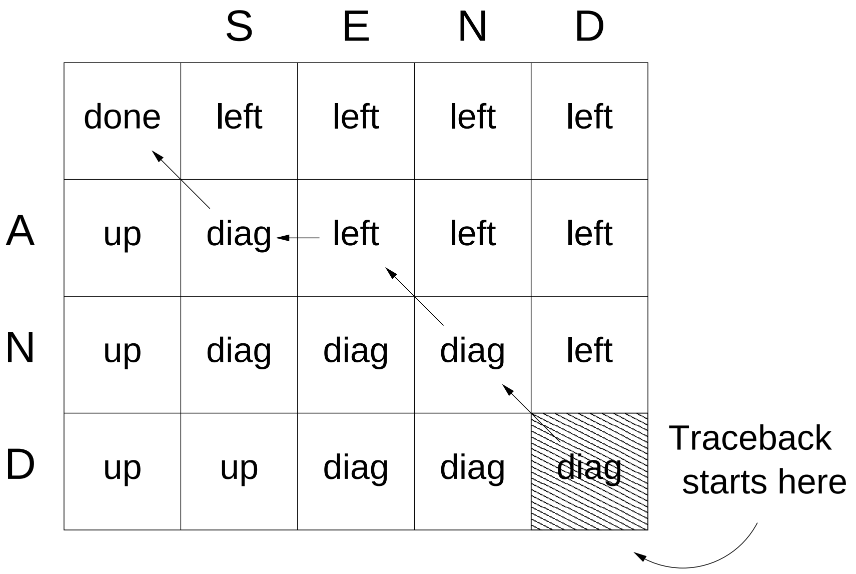
The best alignment

- The alignment is deduced from the values of cells along the traceback path, by taking into account the values of the cell in the traceback matrix:
  - *diag* – the letters from two sequences are aligned
  - *left* – a gap is introduced in the left sequence
  - *up* – a gap is introduced in the top sequence
- Sequences are aligned backwards.

The traceback step-by-step (1)

- The first cell from the traceback path is "diag" implying that the corresponding letters are aligned:

```
D
D
```



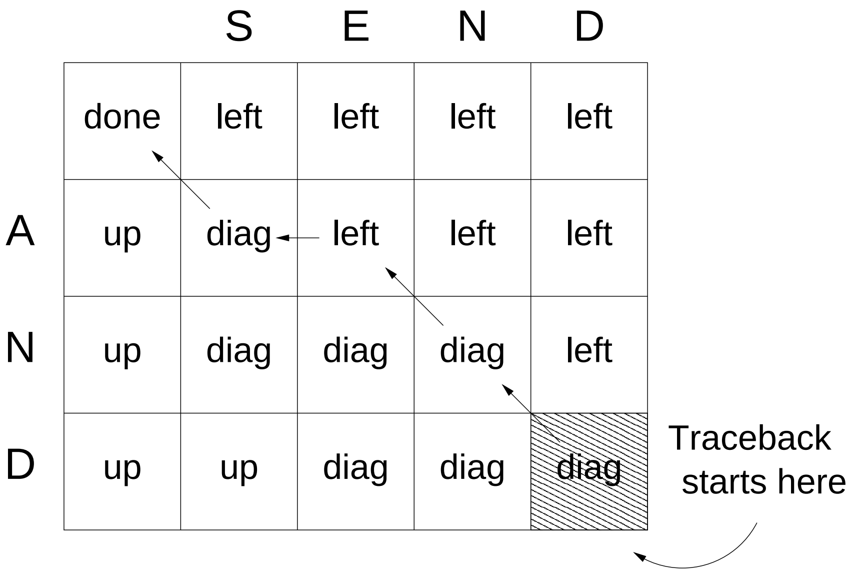
		S	E	N	D
	done	left	left	left	left
A	up	diag	left	left	left
N	up	diag	diag	diag	left
D	up	up	diag	diag	diag

Traceback starts here

The traceback step-by-step (2)

- The second cell from the traceback path is also ”diag” implying that the corresponding letters are aligned:

```
ND
ND
```

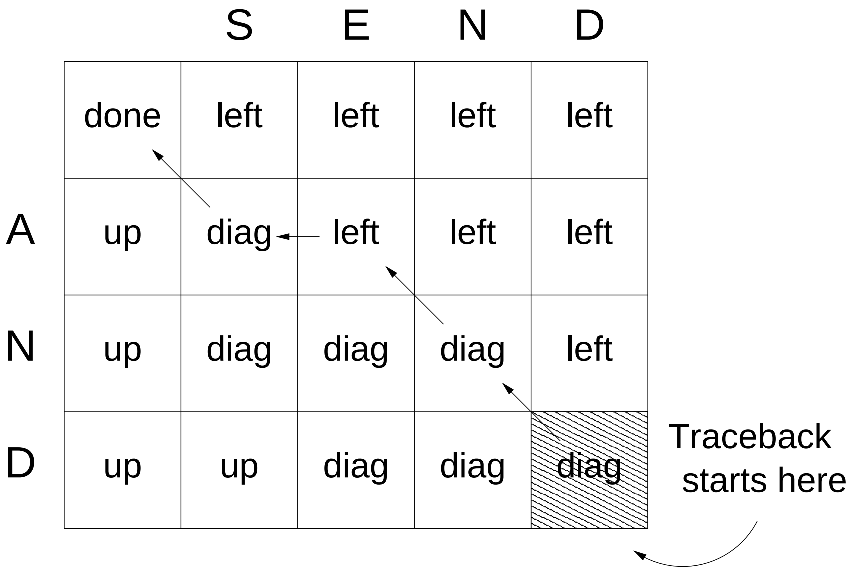


		S	E	N	D
	done	left	left	left	left
A	up	diag	left	left	left
N	up	diag	diag	diag	left
D	up	up	diag	diag	diag

The traceback step-by-step (3)

- The third cell from the traceback path is ”left” implying the gap in the left sequence (i.e. we stay on the letter `A` from the left sequence):

```
END
-ND
```



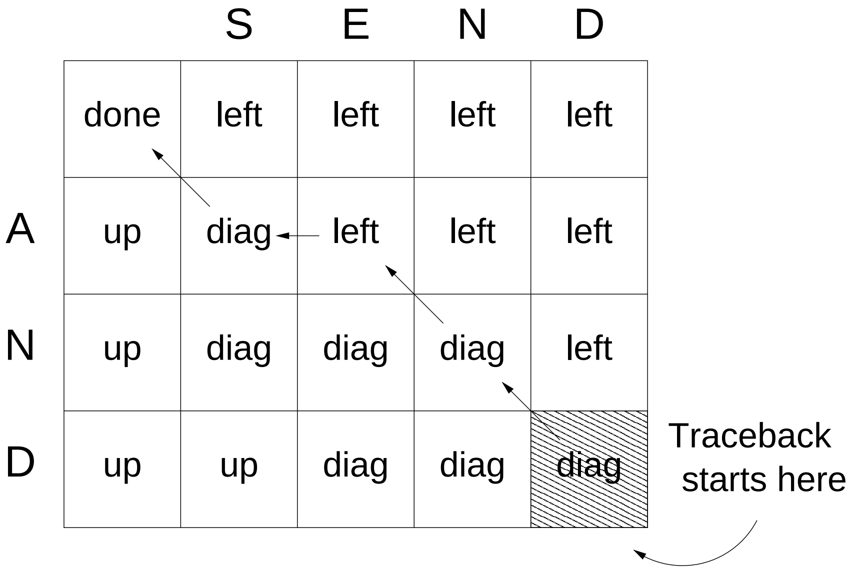
		S	E	N	D
	done	left	left	left	left
A	up	diag	left	left	left
N	up	diag	diag	diag	left
D	up	up	diag	diag	diag

Traceback starts here

The traceback step-by-step (4)

- The fourth cell from the traceback path is also ”diag” implying that the corresponding letters are aligned. We consider the letter A again, this time it is aligned with S:

```
SEND
A-ND
```



		S	E	N	D
	done	left	left	left	left
A	up	diag	left	left	left
N	up	diag	diag	diag	left
D	up	up	diag	diag	diag

Traceback starts here

Compare with the exhaustive search

- The best alignment via the Needleman-Wunsch algorithm:

```
    SEND
    A-ND
```

- The exhaustive search:

```
SEND
-AND score:  +1
A-ND score:  +3 ← the best
AN-D score:  -3
AND- score:  -8
```

A few observations

- It was much easier to align `SEND` and `AND` by the exhaustive search!
- As we consider longer sequences the situation quickly turns against the exhaustive search:
  - Two 12 residue sequences would require considering ~ 1 million alignments.
  - Two 150 residue sequences would require considering $\sim 10^{88}$ alignments ($\sim 10^{78}$ is the estimated number of atoms in the Universe).
- For two 150 residue sequences the Needleman-Wunsch algorithm requires filling a 150×150 matrix.

The summary

- The alignment is over the entire length of two sequences: the traceback starts from the lower right corner of the traceback matrix, and completes in the upper left cell of the matrix.

- The Needleman-Wunsch algorithm works in the same way regardless of the length or complexity of sequences, and *guarantees* to find the best alignment.

- The Needleman-Wunsch algorithm is appropriate for finding the best alignment of two sequences which are (*i*) of the similar length; (*ii*) similar across their entire lengths.