Dept. of Computer Science and Engineering
University of Rajshahi
www.ru.ac.bd

Dr. Shamim Ahmad

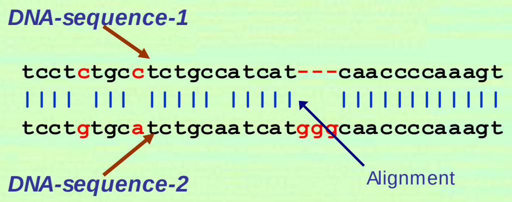
Causes for sequence (dis)similarity

mutation:	a nucleotide at a certain location is replaced by ***another*** nucleotide (e.g.: A**T**A → A**G**A)
insertion:	at a certain location one new nucleotide is inserted inbetween two existing nucleotides (e.g.: AA → A**G**A)
deletion:	at a certain location one existing nucleotide is deleted (e.g.: AC**T**G → AC**-**G)
indel:	an **in**sertion or a **del**etion

3.4 *Sequence alignment: global and local*

Find the similarity between two (or more) DNA-sequences by finding a good **alignment** between them.

The biological problem of sequence alignment



```
tcctctgcctctgccatcat---caaccccaaagt
|||| ||| ||||| |||||   ||||||||||||
tcctgtgcatctgcaatcatgggcaaccccaaagt
```

Sequence alignment - definition

Sequence alignment is an arrangement of two or more sequences, highlighting their similarity.

The sequences are padded with **gaps** (dashes) so that wherever possible, columns contain **identical characters** from the sequences involved

```
tcctctgcctctgccatcat---caaccccaaagt
|||| ||| ||||| |||||   ||||||||||||
tcctgtgcatctgcaatcatgggcaaccccaaagt
```

5

Algorithms

Needleman-Wunsch
Pairwise **global** alignment only.

Smith-Waterman
Pairwise, **local** (*or global*) alignment.

BLAST
Pairwise **heuristic** local alignment

6

Pairwise alignment

Pairwise sequence alignment methods are concerned with finding the best-matching piecewise local or global alignments of protein (amino acid) or DNA (nucleic acid) sequences.

Typically, the purpose of this is to find **homologues** (relatives) of a gene or gene-product in a database of known examples.

This information is useful for answering a variety of biological questions:

1. The identification of sequences of **unknown** structure or function.

2. The study of **molecular evolution**.

7

Global alignment

A **global alignment** between two sequences is an alignment in which all the characters in both sequences participate in the alignment.

Global alignments are useful mostly for finding closely-related sequences.

As these sequences are also easily identified by local alignment methods global alignment is now somewhat deprecated as a technique.

Further, there are several complications to molecular evolution (such as **domain shuffling**) which prevent these methods from being useful.

8

Global Alignment

Find the **global** best fit between two sequences

Example: the sequences **s** = VIVALASVEGAS and **t** = VIVADAVIS align like:

```
           V I V A L A S V E G A S
A(s,t) =   | | | |   |   |       |
           V I V A D A - V - - I S
```

indels

The Needleman-Wunsch algorithm

The **Needleman-Wunsch algorithm** (1970, J Mol Biol. 48(3):443-53) performs a global alignment on two sequences (**s** and **t**) and is applied to align protein or nucleotide sequences.

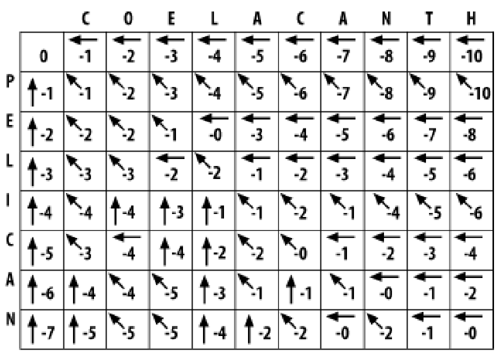
The Needleman-Wunsch algorithm is an example of **dynamic programming**, and is guaranteed to find the alignment with the maximum score.

The Needleman-Wunsch algorithm

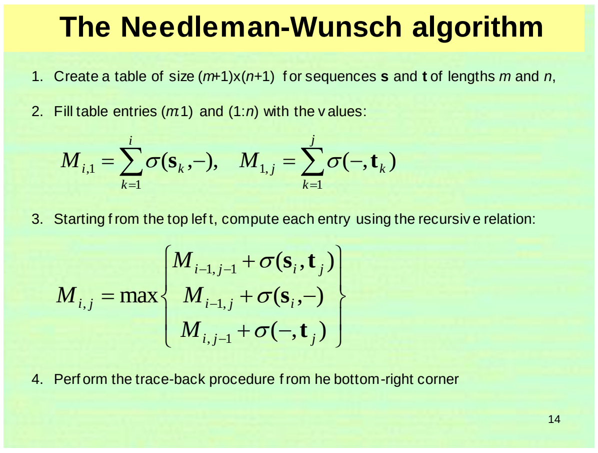
Of course this works for both DNA-sequences as for protein-sequences.

Alignment scoring function

The cost of aligning two symbols $\boldsymbol{x_i}$ and $\boldsymbol{y_j}$ is the ***scoring function*** $\boldsymbol{\sigma(x_i,y_j)}$



		C	O	E	L	A	C	A	N	T	H
	0	← -1	← -2	← -3	← -4	← -5	← -6	← -7	← -8	← -9	← -10
P	↑ -1	↖ -1	↖ -2	↖ -3	↖ -4	↖ -5	↖ -6	↖ -7	↖ -8	↖ -9	↖ -10
E	↑ -2	↖ -2	↖ -2	↖ -1	← -0	← -3	← -4	← -5	← -6	← -7	← -8
L	↑ -3	↖ -3	↖ -3	← -2	↖ -2	← -1	← -2	← -3	← -4	← -5	← -6
I	↑ -4	↖ -4	↑ -4	↑ -3	↑ -1	↖ -1	↖ -2	↖ -1	↖ -4	↖ -5	↖ -6
C	↑ -5	↖ -3	← -4	↑ -4	↑ -2	↖ -2	↖ -0	← -1	← -2	← -3	← -4
A	↑ -6	↑ -4	↖ -4	↖ -5	↑ -3	↖ -1	↑ -1	↖ -1	← -0	← -1	← -2
N	↑ -7	↑ -5	↖ -5	↖ -5	↑ -4	↑ -2	↖ -2	← -0	↖ -2	← -1	← -0



The Needleman-Wunsch algorithm

1. Create a table of size $(m+1)\text{x}(n+1)$ for sequences **s** and **t** of lengths m and n,
2. Fill table entries $(m:1)$ and $(1:n)$ with the values:

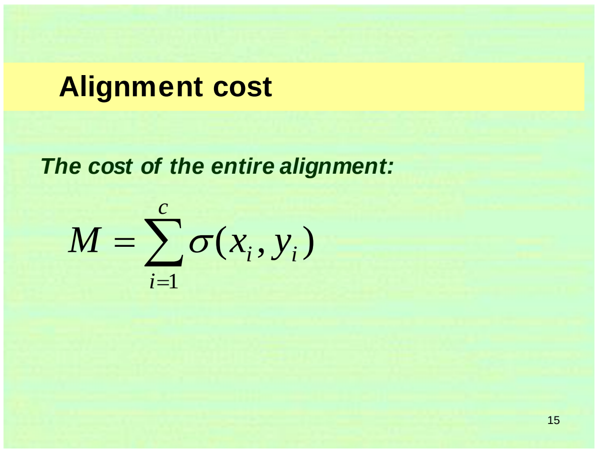
$$M_{i,1} = \sum_{k=1}^{i} \sigma(\mathbf{s}_k, -), \quad M_{1,j} = \sum_{k=1}^{j} \sigma(-, \mathbf{t}_k)$$

3. Starting from the top left, compute each entry using the recursive relation:

$$M_{i,j} = \max \begin{Bmatrix} M_{i-1,j-1} + \sigma(\mathbf{s}_i, \mathbf{t}_j) \\ M_{i-1,j} + \sigma(\mathbf{s}_i, -) \\ M_{i,j-1} + \sigma(-, \mathbf{t}_j) \end{Bmatrix}$$

4. Perform the trace-back procedure from he bottom-right corner

14

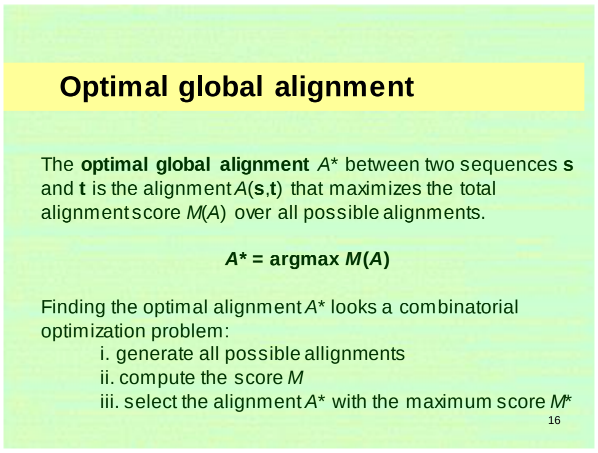


Alignment cost

The cost of the entire alignment:

$$M = \sum_{i=1}^{c} \sigma(x_i, y_i)$$

15



Optimal global alignment

The **optimal global alignment** A^* between two sequences **s** and **t** is the alignment $A(\mathbf{s},\mathbf{t})$ that maximizes the total alignment score $M(A)$ over all possible alignments.

$$\boldsymbol{A^* = \text{argmax}\ M(A)}$$

Finding the optimal alignment A^* looks a combinatorial optimization problem:

i. generate all possible allignments
ii. compute the score M
iii. select the alignment A^* with the maximum score M^*

16

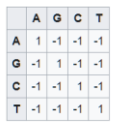
A simple scoring function

$\sigma(-,a) = \sigma(a,-) = -1$

$\sigma(a,b) = -1$ if $a \neq b$

$\sigma(a,b) = 1$ if $a = b$

Similarity Matrix



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

This substitution matrix can be described as: $s(a_i, b_j) = \begin{cases} +1, & a_i = b_j \\ -1, & a_i \neq b_j \end{cases}$

The substitution matrix

A more realistic scoring function is given by the biologically inspired substitution matrix :

-	A	G	C	T
A	10	-1	-3	-4
G	-1	7	-5	-3
C	-3	-5	9	0
T	-4	-3	0	8

Examples:

* **PAM** (*Point Accepted Mutation*) (Margaret Dayhoff)
* **BLOSUM** (*BLOck SUbstitution Matrix*) (Henikoff and Henikoff)

Scoring function

The cost for aligning the two sequences **s** = VIVALASVEGAS and **t** = VIVADAVIS :

```
          V I V A L A S V E G A S
A(s,t) =  | | | |   |   |       |
          V I V A D A - V - - I S
```

is:

***M*(*A*)** = 7 matches + 2 mismatches + 3 gaps
= 7 – 2 – 3 = **2**

The Needleman-Wunsch algorithm

For example, if the substitution matrix was

-	A	G	C	T
A	10	-1	-3	-4
G	-1	7	-5	-3
C	-3	-5	9	0
T	-4	-3	0	8

then the alignment:

```
AGACTAGTTAC
CGA---GACGT
```

with a **gap penalty** of -5, would have the following score…

$$S(A,C)+S(G,G)+S(A,A)+3\times d+S(G,G)+S(T,A)+S(T,C)+S(A,G)+S(C,T)$$
$$= -3+7+10-3\times 5+7+-4+0+-1+0=1$$

The Needleman-Wunsch algorithm

1. Create a table of size (*m*+1)x(*n*+1) for sequences **s** and **t** of lengths *m* and *n*,
2. Fill table entries (*m*,1) and (1:*n*) with the values:

$$M_{i,1}=\sum_{k=1}^{i}\sigma(\mathbf{s}_k,-),\quad M_{1,j}=\sum_{k=1}^{j}\sigma(-,\mathbf{t}_k)$$

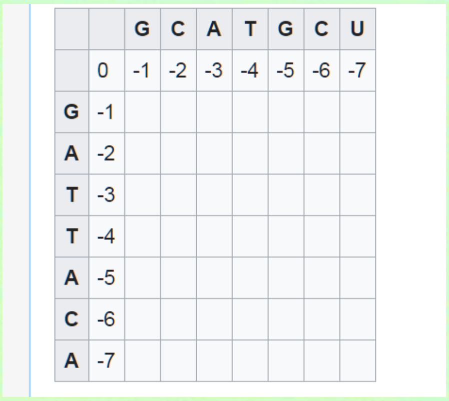
3. Starting from the top left, compute each entry using the recursive relation:

$$M_{i,j}=\max\begin{Bmatrix} M_{i-1,j-1}+\sigma(\mathbf{s}_i,\mathbf{t}_j) \\ M_{i-1,j}+\sigma(\mathbf{s}_i,-) \\ M_{i,j-1}+\sigma(-,\mathbf{t}_j) \end{Bmatrix}$$

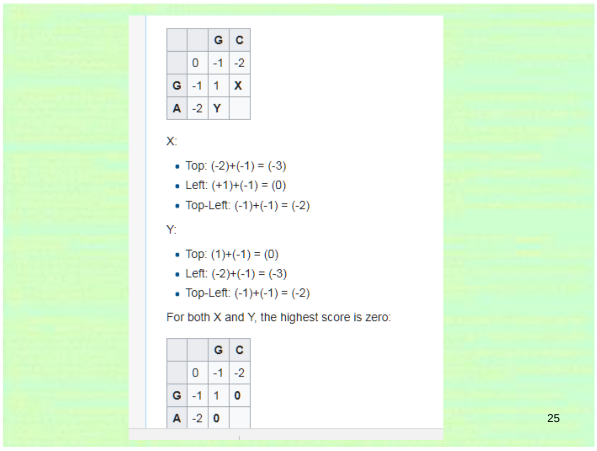
4. Perform the trace-back procedure from he bottom-right corner

•The path from the top or left cell represents an indel pairing
- •, so take the score of the left and the top cell
- • and add the score for indel to each of them.

•The diagonal path represents a match/mismatch
- •so take the score of the top-left diagonal cell
- • and add the score for match if the corresponding bases in the row and column are matching or
- •the score for mismatch if they do not.



		G	C	A	T	G	C	U
	0	-1	-2	-3	-4	-5	-6	-7
G	-1							
A	-2							
T	-3							
T	-4							
A	-5							
C	-6							
A	-7							



		G	C
	0	-1	-2
G	-1	1	X
A	-2	Y	

X:

- Top: (-2)+(-1) = (-3)
- Left: (+1)+(-1) = (0)
- Top-Left: (-1)+(-1) = (-2)

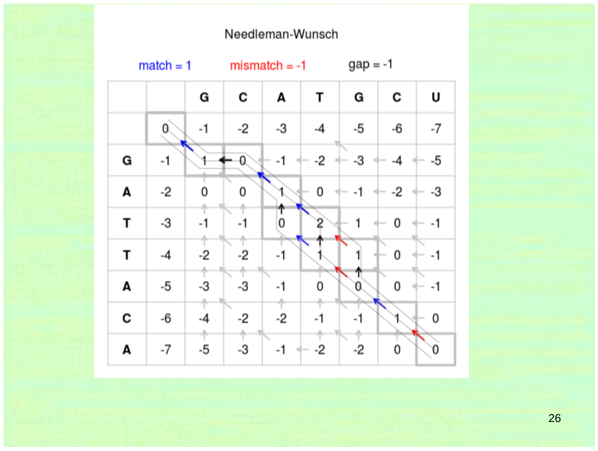
Y:

- Top: (1)+(-1) = (0)
- Left: (-2)+(-1) = (-3)
- Top-Left: (-1)+(-1) = (-2)

For both X and Y, the highest score is zero:

		G	C
	0	-1	-2
G	-1	1	0
A	-2	0	

25



Needleman-Wunsch

match = 1 mismatch = -1 gap = -1

		G	C	A	T	G	C	U
	0	-1	-2	-3	-4	-5	-6	-7
G	-1	1	0	-1	-2	-3	-4	-5
A	-2	0	0	1	0	-1	-2	-3
T	-3	-1	-1	0	2	1	0	-1
T	-4	-2	-2	-1	1	1	0	-1
A	-5	-3	-3	-1	0	0	0	-1
C	-6	-4	-2	-2	-1	-1	1	0
A	-7	-5	-3	-1	-2	-2	0	0

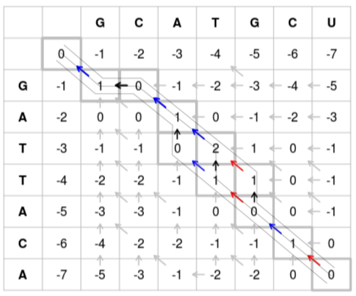
26

- A diagonal arrow represents a match or mismatch,
  - so the letters of the column and the letter of the row of the origin cell will align.
- A horizontal or vertical arrow represents an indel.
  - Horizontal arrows will align a gap ("-") to the letter of the column (the "top" sequence),
  - Vertical arrows will align a gap to the letter of the row (the "side" sequence).

- If there are multiple arrows to choose from
  - They represent a branching of the alignments.
- If two or more branches all belong to paths from the bottom right to the top left cell
  - They are equally viable alignments
  - In this case, note the paths as separate alignment candidates.

Needleman-Wunsch

match = 1 mismatch = -1 gap = -1



		G	C	A	T	G	C	U
	0	-1	-2	-3	-4	-5	-6	-7
G	-1	1	← 0	-1	-2	-3	-4	-5
A	-2	0	0	1	0	-1	-2	-3
T	-3	-1	-1	0	2	1	0	-1
T	-4	-2	-2	-1	1	1	0	-1
A	-5	-3	-3	-1	0	0	0	-1
C	-6	-4	-2	-2	-1	-1	1	0
A	-7	-5	-3	-1	-2	-2	0	0

U → CU → GCU → -GCU → T-GCU → AT-GCU → CAT-GCU → GCATG-CU
A → CA → ACA → TACA → TTACA → ATTACA → -ATTACA → G-ATTACA

29

		C	O	E	L	A	C	A	N	T	H
	0	← -1	← -2	← -3	← -4	← -5	← -6	← -7	← -8	← -9	← -10
P	↑ -1										
E	↑ -2										
L	↑ -3										
I	↑ -4										
C	↑ -5										
A	↑ -6										
N	↑ -7										

		C	O	E	L	A	C	A	N	T	H
	0	← -1	← -2	← -3	← -4	← -5	← -6	← -7	← -8	← -9	← -10
P	↑ -1	↖ -1									

		C	O	E	L	A	C	A	N	T	H
	0	← -1	← -2	← -3	← -4	← -5	← -6	← -7	← -8	← -9	← -10
P	↑ -1	↖ -1	↖ -2	↖ -3	↖ -4	↖ -5	↖ -6	↖ -7	↖ -8	↖ -9	↖ -10
E	↑ -2	↖ -2	↖ -2	↖ -1	← -0	← -3	← -4	← -5	← -6	← -7	← -8
L	↑ -3	↖ -3	↖ -3	← -2	↖ -2	← -1	← -2	← -3	← -4	← -5	← -6
I	↑ -4	↖ -4	↑ -4	↑ -3	↑ -1	↖ -1	↖ -2	↖ -1	↖ -4	↖ -5	↖ -6
C	↑ -5	↖ -3	← -4	↑ -4	↑ -2	↖ -2	↖ -0	← -1	← -2	← -3	← -4
A	↑ -6	↑ -4	↖ -4	↖ -5	↑ -3	↖ -1	↑ -1	↖ -1	← -0	← -1	← -2
N	↑ -7	↑ -5	↖ -5	↖ -5	↑ -4	↑ -2	↖ -2	← -0	↖ -2	← -1	← -0

Similarity Matrix

	A	G	C	T
A	1	-1	-1	-1
G	-1	1	-1	-1
C	-1	-1	1	-1
T	-1	-1	-1	4

Needleman Wunsch Sequence Alignment

The pseudo-code for the algorithm to compute the F matrix therefore looks like this (array and sequence indexes start at 0):

```
d ← MismatchScore
for i=0 to length(B)-1
        F(i,0) <- d*i
for j=0 to length(A)-1
        F(0,j) <- d*j
for i=1 to length(B)
        for j = 1 to length(A) {
                Choice1 <- F(i-1,j-1) + S(B(i), A(j))
                Choice2 <- F(i-1, j) + d
                Choice3 <- F(i, j-1) + d
                F(i,j) <- max(Choice1, Choice2, Choice3)
        }
```

•Once the F matrix is computed, the bottom right hand corner of the matrix is the maximum score for any alignment.

•To compute which alignment actually gives this score, you can start from the bottom right cell, and compare the value with the three possible sources(Choice1, Choice2, and Choice3 above) to see which it came from.

If Choice1, then A(j) and B(i) are aligned,
If Choice2, then B(i) is aligned with a gap, and
If Choice3, then A(j) is aligned with a gap.

Needleman Wunsch Sequence Alignment

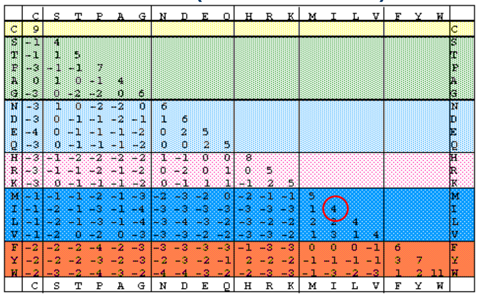
```
AlignmentA <- "" ; AlignmentB <- "“;
i <- length(B); j <- length(A);

while (i > 0 AND j > 0) {
    Score <- F(i,j); ScoreDiag <- F(i - 1, j - 1);
    ScoreLeft <- F(i, j - 1); ScoreUp <- F(i - 1, j);
    if (Score == ScoreDiag + S(A(j), B(i))) {
            AlignmentA <- A(j) + AlignmentA; AlignmentB <- B(i) + AlignmentB;
            i <- i – 1; j <- j – 1; }
    else if (Score == ScoreLeft + d) {
            AlignmentA <- A(j) + AlignmentA; AlignmentB <- "-" + AlignmentB;
            j <- j - 1 }
    else if (Score == ScoreUp + d) {
            AlignmentA <- "-" + AlignmentA; AlignmentB <- B(i) + AlignmentB;
            i <- i - 1 }
}
while (j > 0) { AlignmentA <- A(j) + AlignmentA; AlignmentB <- "-" + AlignmentB; j <- j - 1 }
while (i > 0) { AlignmentA <- "-" + AlignmentA; AlignmentB <- B(i) + AlignmentB; i <- i - 1 }
```

Substitution Score

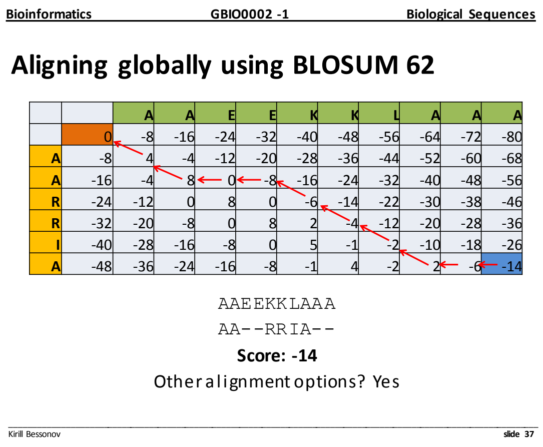
KATHOLIEKE UNIVERSITEIT LEUVEN

Substitution matrix (BLOSUM 50 matrix)



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

Log odds score can be positive (identities, conservative replacements) and negative



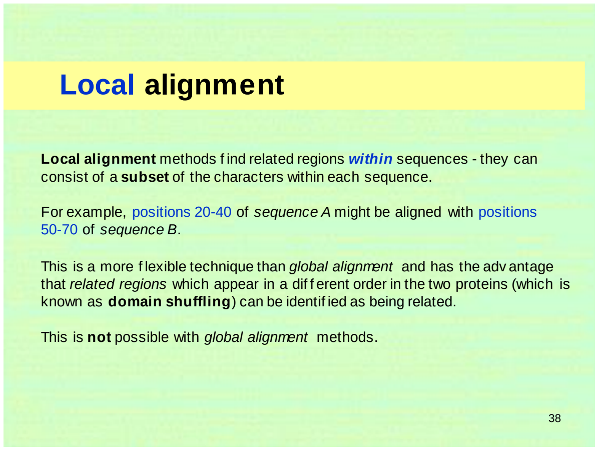
Aligning globally using BLOSUM 62

		A	A	E	E	K	K	L	A	A	A
	0	-8	-16	-24	-32	-40	-48	-56	-64	-72	-80
A	-8	4	-4	-12	-20	-28	-36	-44	-52	-60	-68
A	-16	-4	8	0	-8	-16	-24	-32	-40	-48	-56
R	-24	-12	0	8	0	-6	-14	-22	-30	-38	-46
R	-32	-20	-8	0	8	2	-4	-12	-20	-28	-36
I	-40	-28	-16	-8	0	5	-1	-2	-10	-18	-26
A	-48	-36	-24	-16	-8	-1	4	-2	2	-6	-14

```
AAEEKKLAAA
AA--RRIA--
```

Score: -14

Other alignment options? Yes



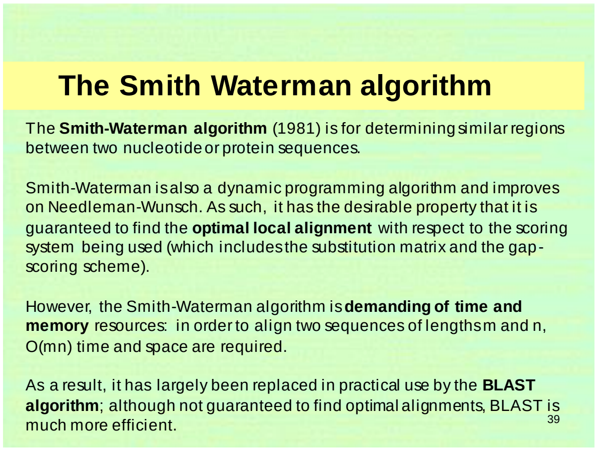
Local alignment

Local alignment methods find related regions ***within*** sequences - they can consist of a **subset** of the characters within each sequence.

For example, positions 20-40 of *sequence A* might be aligned with positions 50-70 of *sequence B*.

This is a more flexible technique than *global alignment* and has the advantage that *related regions* which appear in a different order in the two proteins (which is known as **domain shuffling**) can be identified as being related.

This is **not** possible with *global alignment* methods.



The Smith Waterman algorithm

The **Smith-Waterman algorithm** (1981) is for determining similar regions between two nucleotide or protein sequences.

Smith-Waterman is also a dynamic programming algorithm and improves on Needleman-Wunsch. As such, it has the desirable property that it is guaranteed to find the **optimal local alignment** with respect to the scoring system being used (which includes the substitution matrix and the gap-scoring scheme).

However, the Smith-Waterman algorithm is **demanding of time and memory** resources: in order to align two sequences of lengths m and n, O(mn) time and space are required.

As a result, it has largely been replaced in practical use by the **BLAST algorithm**; although not guaranteed to find optimal alignments, BLAST is much more efficient.

Smith–Waterman Algorithm

	Smith–Waterman algorithm	Needleman–Wunsch algorithm
Initialization	First row and first column are set to 0	First row and first column are subject to gap penalty
Scoring	Negative score is set to 0	Score can be negative
Traceback	Begin with the highest score, end when 0 is encountered	Begin with the cell at the lower right of the matrix, end at top left cell

The Smith-Waterman algorithm

1. Create a table of size (*m*+1)x(*n*+1) for sequences **s** and **t** of lengths *m* and *n*,
2. Fill table entries (1,1:*m*+1) and (1:n+1,1) with zeros.
3. Starting from the top left, compute each entry using the recursive relation:

$$M_{i,j} = \max\begin{Bmatrix} M_{i-1,j-1} + \sigma(\mathbf{s}_i, \mathbf{t}_j) \\ M_{i-1,j} + \sigma(\mathbf{s}_i, -) \\ M_{i,j-1} + \sigma(-, \mathbf{t}_j) \\ 0 \end{Bmatrix}$$

4. Perform the trace-back procedure from the maximum element in the table to the first zero element on the trace-back path.

42

Similarity Matrix

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

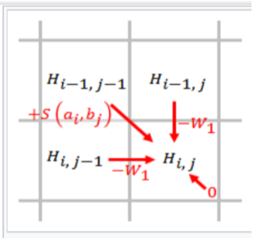
This substitution matrix can be described as: $s(a_i, b_j) = \begin{cases} +1, & a_i = b_j \\ -1, & a_i \neq b_j \end{cases}$

Initialize the scoring matrix

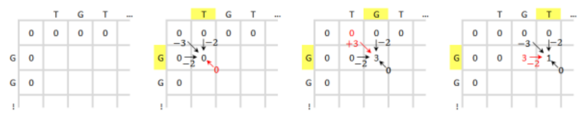
		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0								
G	0								
T	0								
T	0								
G	0								
A	0								
C	0								
T	0								
A	0								

Substitution matrix: $s(a_i, b_j) = \begin{cases} +3, & a_i = b_j \\ -3, & a_i \neq b_j \end{cases}$

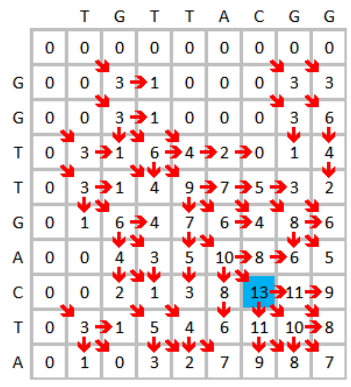
Gap penalty: $W_k = kW_1$
$W_1 = 2$



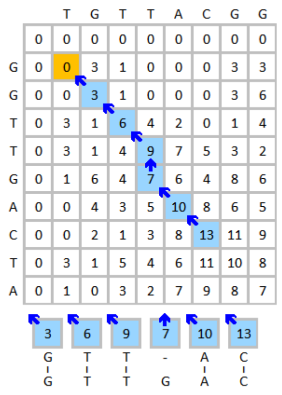
Simplified Smith–Waterman algorithm when linear gap penalty function is used



$$H_{ij} = \max \begin{cases} H_{i-1,j-1} + s(a_i, b_j), \\ H_{i-1,j} - W_1, \\ H_{i,j-1} - W_1, \\ 0 \end{cases}$$



		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7



		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

```
 3   6   9   7   10  13
 G   T   T   -   A   C
 |   |   |       |   |
 G   T   T   G   A   C
```

3. Fill the scoring matrix using the equation below.

$$H_{ij} = \max \begin{cases} H_{i-1,j-1} + s(a_i, b_j), \\ \max_{k \geq 1}\{H_{i-k,j} - W_k\}, \\ \max_{l \geq 1}\{H_{i,j-l} - W_l\}, \\ 0 \end{cases} \qquad (1 \leq i \leq n, 1 \leq j \leq m)$$

where

$H_{i-1,j-1} + s(a_i, b_j)$ is the score of aligning a_i and b_j,

$H_{i-k,j} - W_k$ is the score if a_i is at the end of a gap of length k,

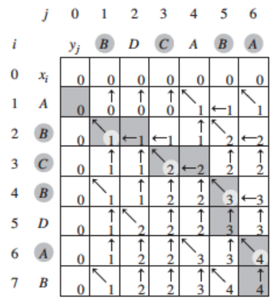
$H_{i,j-l} - W_l$ is the score if b_j is at the end of a gap of length l,

0 means there is no similarity up to a_i and b_j.

Step 3: Computing the length of a LCS

```
LCS-LENGTH(X, Y)
1  m = X.length
2  n = Y.length
3  let b[1..m, 1..n] and c[0..m, 0..n] be new tables
4  for i = 1 to m
5      c[i, 0] = 0
6  for j = 0 to n
7      c[0, j] = 0
8  for i = 1 to m
9      for j = 1 to n
10         if x_i == y_j
11             c[i, j] = c[i − 1, j − 1] + 1
12             b[i, j] = "↖"
13         elseif c[i − 1, j] ≥ c[i, j − 1]
14             c[i, j] = c[i − 1, j]
15             b[i, j] = "↑"
16         else c[i, j] = c[i, j − 1]
17             b[i, j] = "←"
18 return c and b
```

$\Rightarrow$



BCBA $\Leftarrow$ AB C BDAB
BDCAB A $\Leftarrow$

Simplified Smith–Waterman algorithm

When linear gap penalty function is used
A linear gap penalty has the same scores for opening and extending a gap:

Linear [edit]

A linear gap penalty has the same scores for opening and extending a gap:

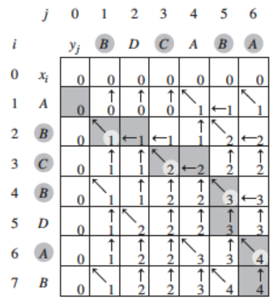
$W_k = kW_1$,

where W_1 is the cost of a single gap.

Step 4: Constructing a LCS (Backtracking)

BCBA $\Leftarrow$ AB C BDAB
BDCAB A $\Leftarrow$

```
PRINT-LCS(b, X, i, j)
1  if i == 0 or j == 0
2      return
3  if b[i, j] == "↖"
4      PRINT-LCS(b, X, i − 1, j − 1)
5      print x_i
6  elseif b[i, j] == "↑"
7      PRINT-LCS(b, X, i − 1, j)
8  else PRINT-LCS(b, X, i, j − 1)
```



Simplified Smith–Waterman algorithm

When linear gap penalty function is used
A linear gap penalty has the same scores for opening and extending a gap:

Linear [edit]

A linear gap penalty has the same scores for opening and extending a gap:

$$W_k = kW_1 ,$$

where W_1 is the cost of a single gap.

Step 4: Constructing a LCS (Backtracking)

BCBA ⇐ AB C BDAB ⇐
BDCAB A

```
PRINT-LCS(b, X, i, j)
1  if i == 0 or j == 0
2      return
3  if b[i, j] == "↖"
4      PRINT-LCS(b, X, i − 1, j − 1)
5      print x_i
6  elseif b[i, j] == "↑"
7      PRINT-LCS(b, X, i − 1, j)
8  else PRINT-LCS(b, X, i, j − 1)
```

j		0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

Dynamic Programming

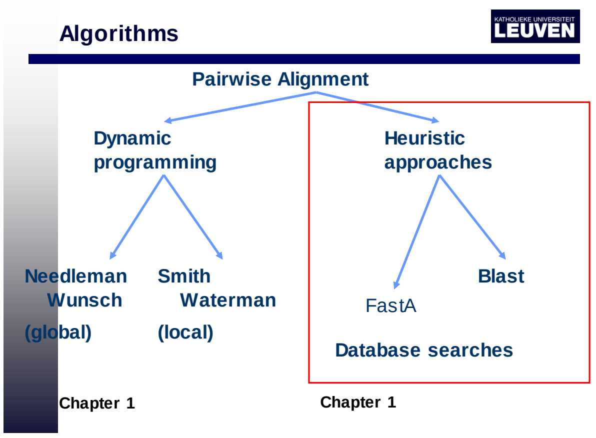
	GAP	M	N	A	L	S	D	R	T
GAP	0	0	0	0	0	0	0	0	0
M	0	6	0	0	4	0	0	0	0
G	0	0	6	1	0	5	1	0	0
S	0	0	1	7	0	2	5	1	1
D	0	0	2	1	3	0	6	4	1
R	0	0	0	0	0	3	0	12	3
T	0	0	0	1	0	1	3	0	15
T	0	0	0	1	0	1	1	2	3
E	0	0	1	0	0	0	4	0	2
T	0	0	0	2	0	1	0	3	3

```
SDRT
SDRT
```

Aligning locally using BLOSUM 62

		A	A	E	E	K	K	L	A	A	A
	0	0	0	0	0	0	0	0	0	0	0
A	0	4	4	0	0	0	0	0	4	4	4
A	0	4	8	3	0	0	0	0	4	8	8
R	0	0	3	8	3	2	2	0	0	3	7
R	0	0	0	3	8	5	4	0	0	0	2
I	0	0	0	0	0	5	2	6	0	0	0
A	0	4	4	0	0	0	4	1	10	4	4

```
KKLA
RRIA
```
Score: 10



Algorithms

Pairwise Alignment

Dynamic programming

Heuristic approaches

Needleman Wunsch

(global)

Smith Waterman

(local)

FastA

Blast

Database searches

Chapter 1

Chapter 1