Introduction to the course &
The pairwise sequence alignment problem

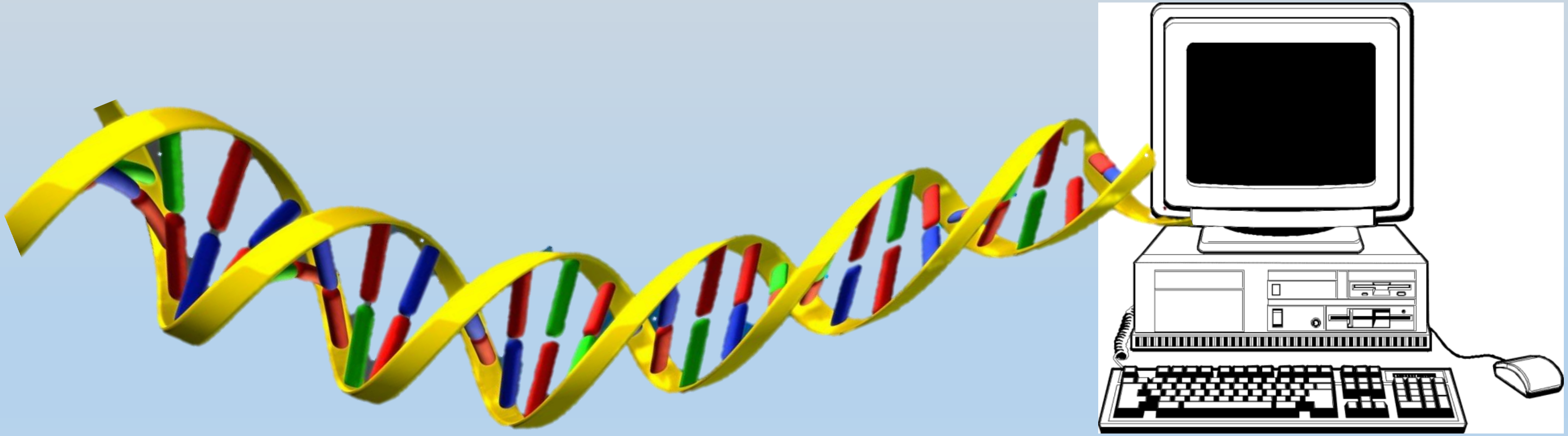
Lecture 1

Reichman University Efi Arazi School of Computer Science

Computational biology

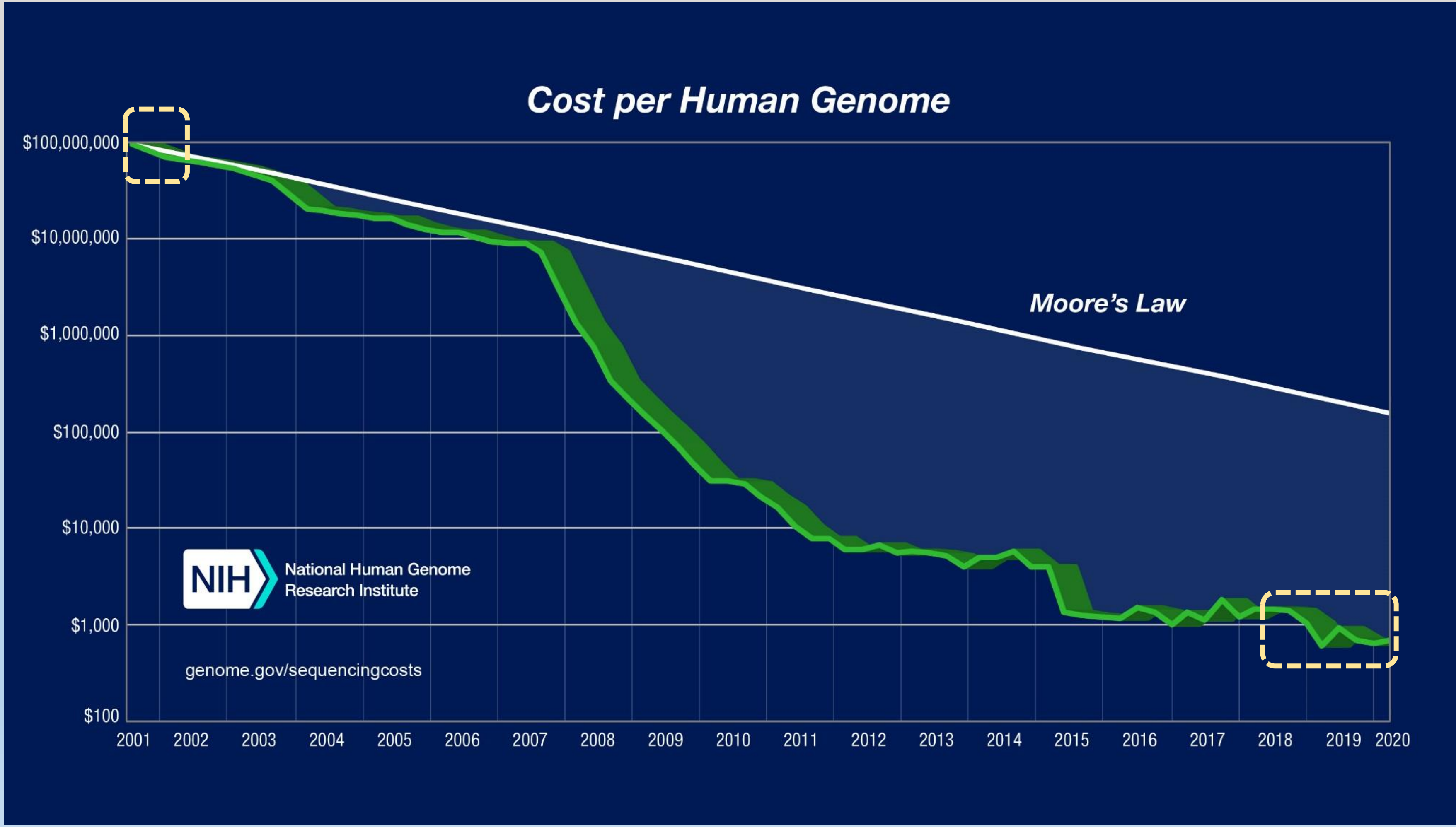
Using computational and statistical tools in biological research

- Algorithms
- Mathematical modeling
- Statistics



Computational biology

Molecular sequence data in the past two decades



This course

We will cover **classical topics** in computational biology, focusing on **sequence analysis**:

- Sequence alignment
- Hidden Markov Models (HMMs)
- Phylogenetic reconstruction – studying evolution and history

Emphasis will be on **algorithmic** aspects – design and proof of properties

We will also address the problem of **modeling** complex problems

This course

What do you need to know?

- Algorithms
- Probability

The biological background required will be provided throughout the course. For a basic introduction of the main concepts, see video uploaded to Panoptro + slides on Piazza. Feel free to ask me questions and consult your best friend – Google.

This course

Material

- The course doesn't follow a specific text book, but the following can be useful if you are looking for a reference:
  - **Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.**
    Richard Durbin (Editor), S. Eddy, A. Krogh, G. Mitchison (Contributor),
    Cambridge University Press, Cambridge, UK.
    http://books.google.co.il/books/about/Biological_sequence_analysis.html?id=R5P2GlJvigQC&redir_esc=y
  - **Inferring Phylogenies.**
    Joseph Felsenstein,
    Sinauer Associates, Sunderland, Massachusetts, USA.
    http://www.sinauer.com/inferring-phylogenies.html

Administration

Lectures:

- Core material will be summarized in slides
- Lectures recorded via Zoom

Administration

Lectures:

- Core material will be summarized in slides
- Lectures recorded via Zoom

Homework:

- Counts for 70% of the final grade
- 4 assignments. Each assignment will contain theory/ algorithms questions and a practical component involving **implementation in code** and analysis
- **Submit in pairs!** But do think about problems individually
- Concluding assignment – 30% of the final grade. Details will be given later

Administration

Reichman University Efi Arazi School of Computer Science

Communication:

- Through the piazza website: https://piazza.com/runi.ac.il/fall2023/cs3571/info
- Office hours: Tuesday @ 17:00 in my office C127 in CS building, or via Zoom
- Contact through piazza or by e-mail: ilan.gronau@runi.ac.il

Lecture overview

- Formulation of the alignment problem
- An efficient algorithm for global alignment
- An efficient algorithm for local alignment
- Other variants of the alignment problem

Brief background in molecular biology
[see video lecture on Panoptro]

Lecture overview

- Formulation of the alignment problem ←
- An efficient algorithm for global alignment
- An efficient algorithm for local alignment
- Other variants of the alignment problem

Brief background in molecular biology
[see video lecture on Panoptro]

Pairwise sequence alignment

Genome Browser Example: looking up the promoter region of a mouse gene in the human genome (the 1 kb sequence before the start of a gene is typically involved in regulation of gene expression)

1. Open Mouse genome browser on the region around gene PAX9: http://genome-euro.ucsc.edu/cgi-bin/hgTracks?db=mm10&position=chr12%3A56694766-56724210
2. Click on gene (blue line with changing widths)
3. Under 'Sequence and Links to Tools and Databases' table click on 'Genomic sequence'
4. Select only 'promoter / Upstream 1000 bp' and press 'submit' and then Copy all text of sequence
5. Open Blast webpage in the National Center for Bioinformatics (NCBI): http://blast.ncbi.nlm.nih.gov/Blast.cgi
6. Click on 'nucleotide blast'
7. Paste sequence under 'Query Sequence'
8. Under 'Database' select Genomic + transcript database and then 'Human genomic plus transcript (Human G+T)'
9. BLAST away…

Pairwise sequence alignment

Genome Browser Example:

- Query sequence is the 1000 bases before the PAX9 gene in the Mouse genome.
- A segment of length 411 of the query sequence (positions 298-708) is matched against a segment of length 416 on Chromosome 14 of the Human genome.
- This matching also includes mismatched bases (e.g. C–T), and gaps (base matched to nothing).

Score	Expect	Identities	Gaps	Strand
481 bits(260)	5e-133	368/419(88%)	11/419(2%)	Plus/Plus

Features: [74770 bp at 5' side: homeobox protein nkx-2.8]
[4516 bp at 3' side: paired box protein pax-9]

```
Query  298       taataaacaaacgcaatcata-c-cataggaggctcaaagccaTCCTTCTAAAAGCAATT  355
                 ||||||||||||| || |||| | ||| ||| |||| |  |  ||||||| |||||||||
Sbjct  36657159  TAATAAACAAACGTAACCATATCACAT-GGA-GCTCCACACTGTCCTTCTGAAAGCAATT  36657216

Query  356       CCACTGTTGAAAACTGAAAAATGGGTGTAATTTGCGTTCAGAAAGTGATGTTAGGGTCAC  415
                 || |||||||||||||||||||||||||||||||| |||||||||| |||||||||||||
Sbjct  36657217  CCGCTGTTGAAAACTGAAAAATGGGTGTAATTTGCTTTCAGAAAGTAATGTTAGGGTCAC  36657276

Query  416       GGCAATTTCCCGGGCCGTTATTTATTTTTACTTTAAAGTCTTTAGGGAAGATTTGCTTAT  475
                 ||||||||| |||||||||||||||||||||||||||||||||||| |||||||||||||
Sbjct  36657277  GGCAATTTCTCGGGCCGTTATTTATTTTTACTTTAAAGTCTTTAGGAAAGATTTGCTTAT  36657336

Query  476       AGACTCGG---A--TAC-AGTATGAGAACCCCGGCAACCCCGCTCGCCTTCAAAGCGGGA  529
                 |  |||||   |  | | | |  | |||  || |||| ||||||||||||||| ||| | 
Sbjct  36657337  ATGCTCGGAAAACTTCCAAATGCGCGAATACCAGCAATCCCGCTCGCCTTCAACGCGTG-  36657395

Query  530       GGGGGCCTGAAGGTGGAGAACAATTACTGAGTGACGCTAATATGGGGAAACTGAAAAGAA  589
                 |||     |  ||||| |||||||||||||||||||||||||||||||||||||||||||
Sbjct  36657396  GGGTAAGGGGGGGTGGGGAACAATTACTGAGTGACGCTAATATGGGGAAACTGAAAAGAA  36657455

Query  590       ATGTCGATTGTTTTTATTGTAACAGAAGGAGTGAGCAAACAGAAAAACCAACCCCGGCTG  649
                 |||||||||||||||||||||| |||||||||||||||||||||||||||||||||| ||
Sbjct  36657456  ATGTCGATTGTTTTTATTGTAATAGAAGGAGTGAGCAAACAGAAAAACCAACCCCGGGTG  36657515

Query  650       ATCGGAAACAGGCAGGCGGAGAATGAAAAGTGGGTTTCAGGCCGCAACAGGCCCCTCCC  708
                 |||||||||||||||||||||||| ||||| ||||||||| | ||||||||||||||||
Sbjct  36657516  ATCGGAAACAGGCAGGCGGAGAATTAAAAGCGGGTTTCAGACAGCAACAGGCCCCTCCC  36657574
```

Pairwise sequence alignment

The objective: find and quantify similarities between sequences (DNA / RNA / protein)

The premise: sequence similarity indicates shared ancestry (homology) and related function.

Example:

```
AGTTCTTGCGC-ATCGATTCCGAGCAGGCGTAAT
AGTCCTTGCGCCAT-GAT---GAACAGGCTTAAT
```

Alignment:

- Expand sequences to be the same length by adding gap symbols (-)
- Each column of the alignment is either a pair of letters or a letter mapped to a gap symbol, indicating sequence insertion or deletion (indel)
- Maximize **matches** and minimize **mismatches** and **gaps** (also called indels – **in**sertion/**del**etion)

Global alignment – formulation

Input: two sequences $S_{1..n}$, $T_{1..m}$ over the same alphabet Σ

Valid alignment: two sequences *S'* and *T'* over the alphabet Σ∪{-} that satisfy:

- Removing the gap labels ('-') from *S'* and *T'* gives *S* and *T*, respectively
- *S'* and *T'* have the same length $L \geq \max\{m,n\}$)

→ The alignment is represented by a sequence of pairs (columns) $\{(S'_i, T'_i)\}_{i=1}^{L}$, each belonging to the Cartesian product (Σ∪{-}) x (Σ∪{-})

Scoring alignments:

- The score is the sum of scores across columns (additivity)
- The score of each column is given by a pre-specified scoring scheme

$$\sigma: (\Sigma\cup\{-\}) \times (\Sigma\cup\{-\}) \rightarrow \mathbb{R}$$

Example

Input:

S = AGTTCTTGCGCATCGATTCCGAGCAGGCGTAAT
T = AGTCCTTGCGCCATGATGAACAGGCTTAAT

Valid alignments:

A_1 :
S' = AGTTCTTGCGCATCGATTCCGAGCAGGCGTAAT
T' = AGTCCTTGCGCCAT---GATGAACAGGCTTAAT

A_2 :
S' = AGTTCTTGCGC-ATCGATTCCGAGCAGGCGTAAT
T' = AGTCCTTGCGCCAT-GAT---GAACAGGCTTAAT

Scoring alignments:

$\sigma(x,x) = 2$ match
$\sigma(x,y) = -2 \ (x \neq y)$ mismatch (substitution)
$\sigma(x,-) = \sigma(-,x) = -3$ insertion/deletion (indel)

$\sigma(A_1) = 23x2 + 8x(-2) + 3x(-3) = 21$
$\sigma(A_2) = 26x2 + 3x(-2) + 5x(-3) = 31$

Alignment scores

The scoring scheme should capture biochemical features of interest

- The idea is typically to try to maximize the matches and penalize for mutations
- Point mutations (base substitutions) are more common than insertions / deletions and are thus typically penalized less
- Some substitutions are more common than others
  - transitions (A←→G and C←→T) vs. transversions ({A,G}←→{C,T}) in DNA
  - similarities between amino acids (e.g., size, charge) in protein sequences
- Most scoring schemes are symmetric (because direction of operation is typically unknown)

We will discuss how scores are determined in Lecture #9

Alignment with maximum score

The number of possible alignments of two sequences of length m and n is

$$\binom{m+n}{m} < A(m,n) < \binom{m+n}{m}^2$$

(left as self exercise)

An exhaustive approach is unfeasible

The challenge is to decide when to open gaps, and this often requires "looking ahead" to see whether this ends up paying off

Lecture overview

- Formulation of the alignment problem
- An efficient algorithm for global alignment ←
- An efficient algorithm for local alignment
- Other variants of the alignment problem

Recursive formulation

Consider the last column of an alignment of $S_{1..n}$ and $T_{1..m}$ and distinguish between 3 cases:

match / mismatch	**Insertion in S**	**Insertion in T**
`******************`S_n	`******************`S_n	`******************`-
`******************`T_m	`******************`-	`******************`T_m
alignment of $S_{1..n\text{-}1}$ and $T_{1..m\text{-}1}$	**alignment of $S_{1..n\text{-}1}$ and $T_{1..m}$**	**alignment of $S_{1..n}$ and $T_{1..m\text{-}1}$**

Key observation: if an alignment of $S_{1..n}$ and $T_{1..m}$ has maximum score, then one of the following must hold:

- Last column (S_n ,T_m) preceded by a max-score alignment of $S_{1..n\text{-}1}$ and $T_{1..m\text{-}1}$
- Last column (S_n ,-) preceded by a max-score alignment of $S_{1..n\text{-}1}$ and $T_{1..m}$
- Last column (- ,T_m) preceded by a max-score alignment of $S_{1..n}$ and $T_{1..m\text{-}1}$

Recursive argument – proof

Claim: if an alignment of $S_{1..n}$ and $T_{1..m}$ has maximum score, then one of the following must hold:

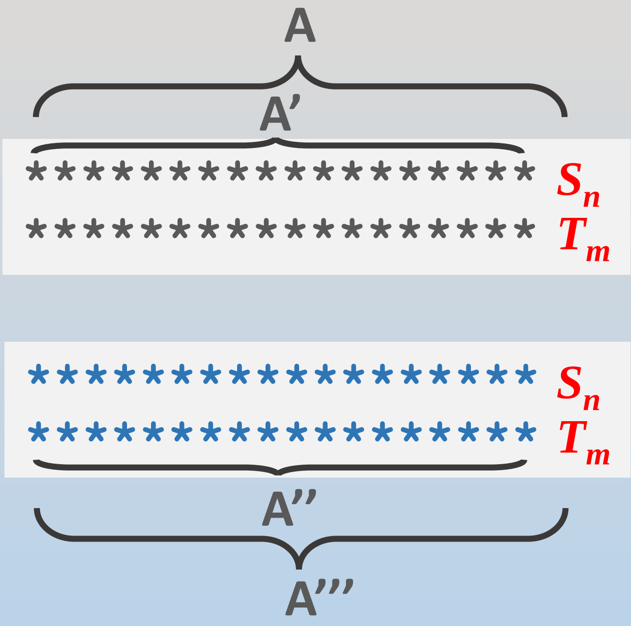
- Last column (S_n ,T_m) preceded by a max-score alignment of $S_{1..n\text{-}1}$ and $T_{1..m\text{-}1}$
- Last column (- ,T_m) preceded by a max-score alignment of $S_{1..n}$ and $T_{1..m\text{-}1}$
- Last column (S_n ,-) preceded by a max-score alignment of $S_{1..n\text{-}1}$ and $T_{1..m}$

Proof: Let A be a max-score alignment of $S_{1..n}$ and $T_{1..m}$. Its last column must be one of the following: (S_n ,T_m), (S_n ,-), or (- ,T_m). We prove the claim case by case

Recursive argument – proof

Proof: Let A be a max-score alignment of $S_{1..n}$ and $T_{1..m}$. Its last column must be one of the following: (S_n, T_m), $(S_n, -)$, or $(-, T_m)$.

Case I: If the last column of A is (S_n, T_m), then the remaining columns represent an alignment A’ of $S_{1..n\text{-}1}$ and $T_{1..m\text{-}1}$.



We will prove that for any alignment A’’ of $S_{1..n\text{-}1}$ and $T_{1..m\text{-}1}$ we have

$\sigma(A'') \leq \sigma(A')$

Consider the alignment A’’’ obtained by adding column (S_n, T_m) to A’’

This is an alignment of $S_{1..n}$ and $T_{1..m}$, so by optimality of A, we have

$\sigma(A''') \leq \sigma(A)$

So: $\sigma(A'') = \sigma(A''') - \sigma(S_n, T_m) \leq \sigma(A) - \sigma(S_n, T_m) = \sigma(A')$

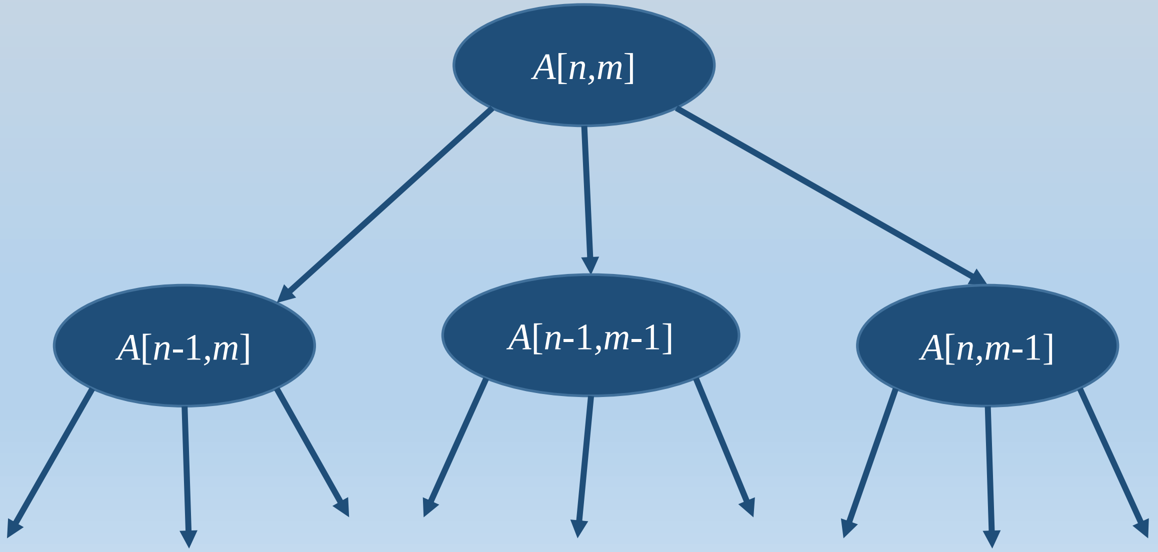
➔ The other two cases are proven similarly

Q.E.D ■

Recursive formulation

Recursive algorithm:

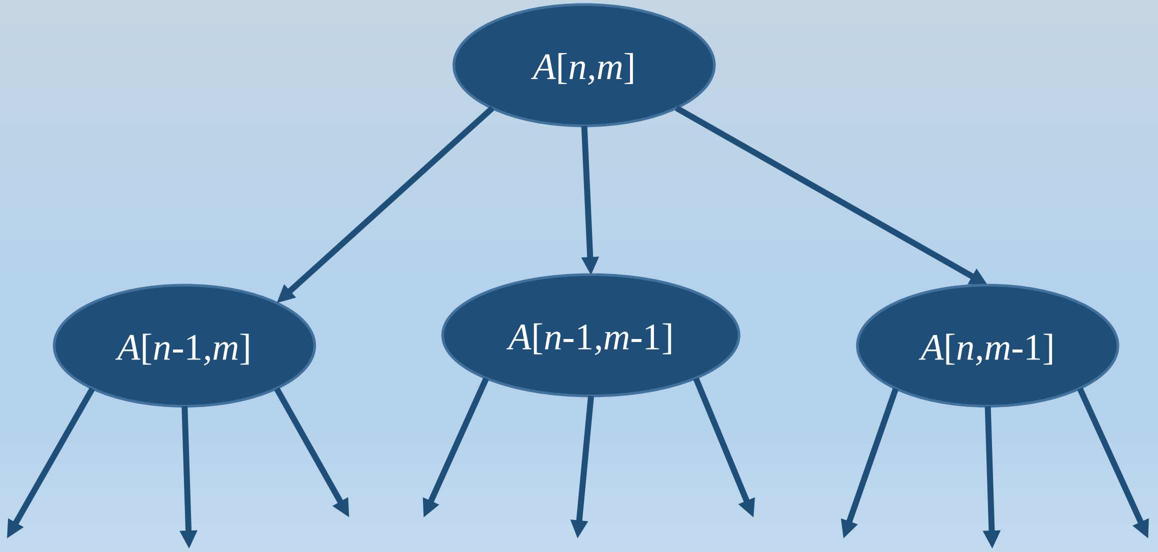
- Find maximum score alignments for
  - $S_{1..n-1}$ and $T_{1..m-1}$ (score $A[n\text{-}1,m\text{-}1]$)
  - $S_{1..n-1}$ and $T_{1..m}$ (score $A[n\text{-}1,m]$)
  - $S_{1..n}$ and $T_{1..m-1}$ (score $A[n,m\text{-}1]$)
- Determine $max\{ A[n\text{-}1,m\text{-}1]+\sigma(S_n ,T_m) \;;\; A[n\text{-}1,m] +\sigma(S_n ,\text{-}) \;;\; A[n,m\text{-}1] +\sigma(\text{-} ,T_m) \}$
- The case resulting in maximum implies the maximum score alignment



Recursive formulation – complexity

Recursion tree:

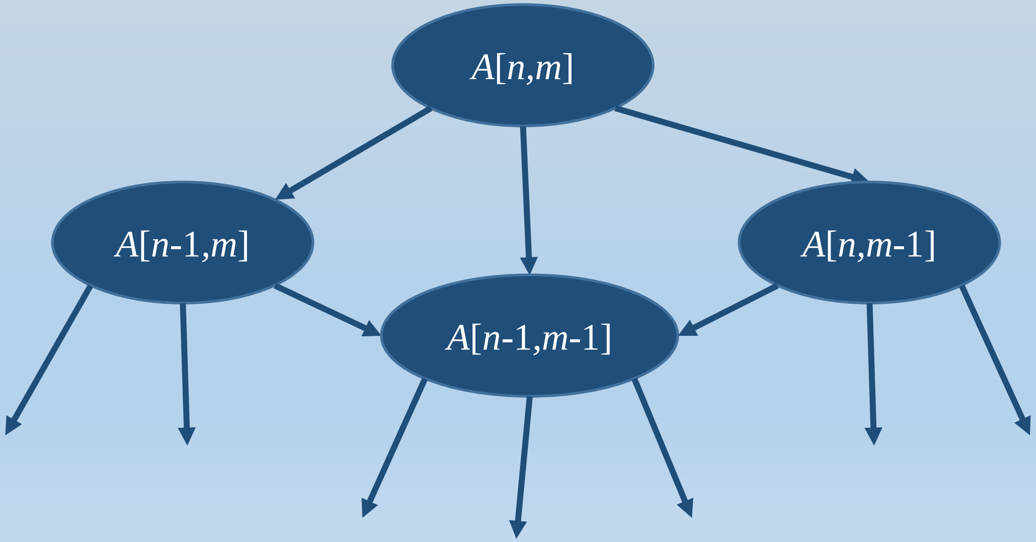
- Each recursion instance calls 3 daughter instances
- Depth of recursion tree is between $\max\{m,n\}$ and $m+n$
- Time complexity of naïve implementation is $\Omega(3^n)$



Recursive formulation – complexity

Recursion tree:

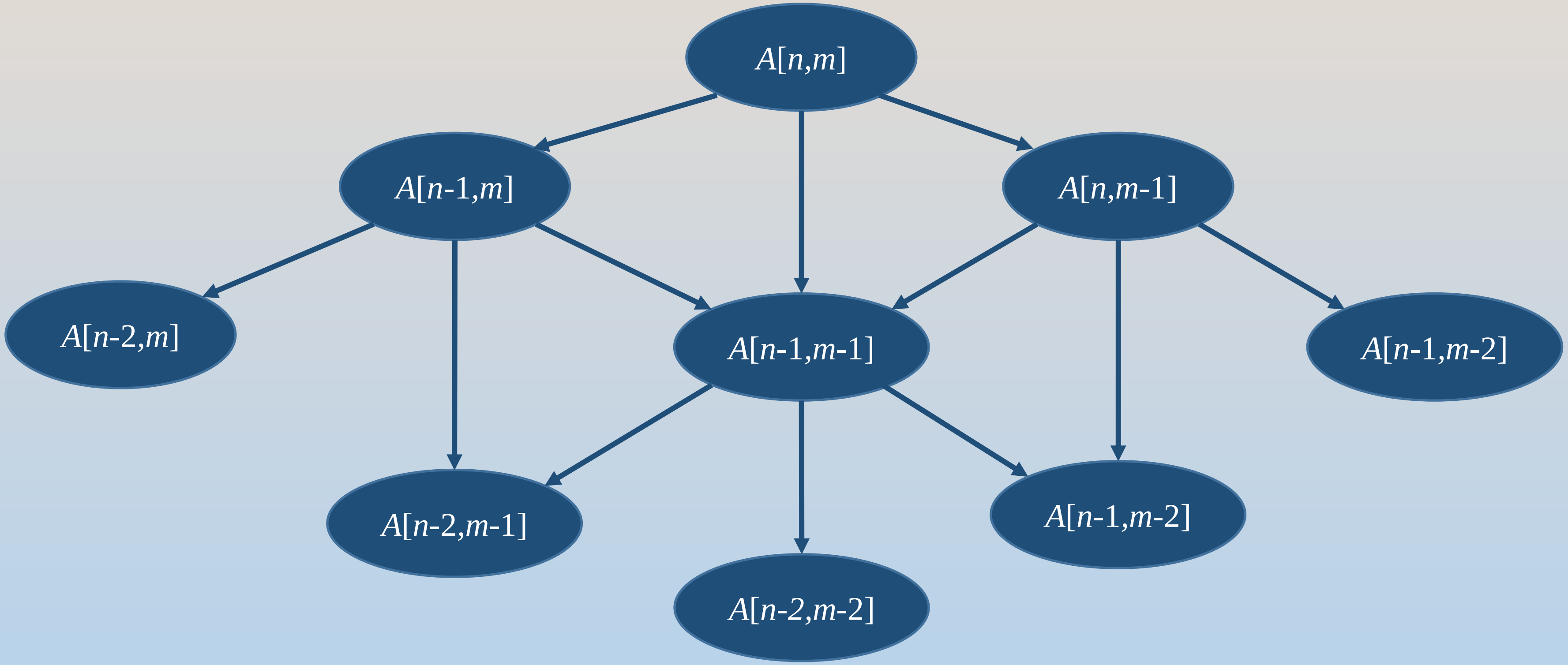
- Each recursion instance calls 3 daughter instances
- Depth of recursion tree is between $\max\{m,n\}$ and $m+n$
- Time complexity of naïve implementation is $\Omega(3^n)$



Notice:

- The paths in the recursion tree intersect
- There are only *mxn* distinct nodes in tree

Recursive formulation – intersecting call tree



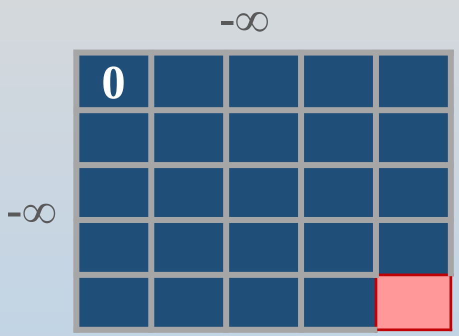
- Recursion tree can be represented by an m x n matrix
- The values can be computed using dynamic programming in $O(mn)$ time and space
- Pay in space to save in time

Needleman-Wunsch algorithm for optimal global alignment

Needleman, S.B. and Wunsch, C.D. (*Jour Mol Biol* 1970)
Sankoff D., (*PNAS* 1972)

Input: two sequences $S_{1..n}$, $T_{1..m}$ over the same alphabet Σ
scoring function: σ: ($\Sigma \cup \{-\}$) x ($\Sigma \cup \{-\}$) → R

Objective: Compute a matrix A s.t. $A[i,j]$ holds the max score alignment of $S_{1..i}$ and $T_{1..j}$



Initialization:

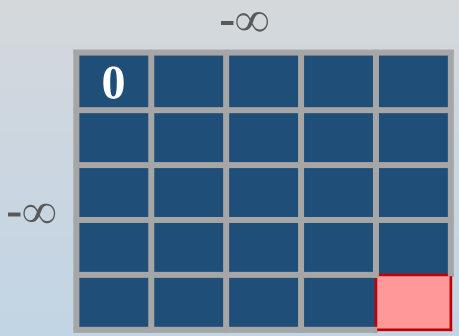
- Create empty matrix A with rows indexed -1..n and columns indexed -1..m
- Row/column 0 correspond to alignments with all-gaps in one sequence
- Row/column -1 are used to deal with edge cases and are initialized to -∞
- Initialize score of empty alignment: $A[0,0] = 0$

Needleman-Wunsch algorithm for optimal global alignment

Needleman, S.B. and Wunsch, C.D. (*Jour Mol Biol* 1970)
Sankoff D., (*PNAS* 1972)

Input: two sequences $S_{1..n}$, $T_{1..m}$ over the same alphabet Σ
scoring function: $\sigma: (\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \rightarrow R$

Objective: Compute a matrix A s.t. $A[i,j]$ holds the max score alignment of $S_{1..i}$ and $T_{1..j}$



Initialization:

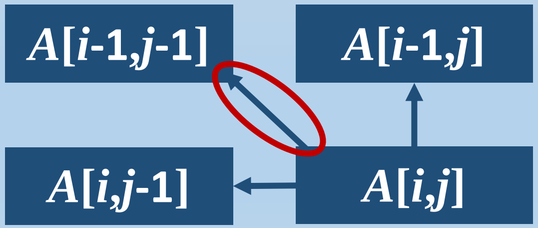
- Create empty matrix A with rows indexed -1..n and columns indexed -1..m
- Initialize $A[0,0] = 0$ and for each i=-1..n and j=-1.. n, $A[i,-1] = A[-1, j] = -\infty$

Main loop: (ascending order of columns/rows)

- For each i=0..n and j=0..m compute $A[i,j]$ as follows:

$$A[i,j] \quad = \quad \max\{ A[i-1,j-1]+\sigma(S_i ,T_j) \ ; \ A[i-1,j] +\sigma(S_i ,-) \ ; A[i,j-1] +\sigma(- ,T_j) \}$$

Keep a pointer to the cell that results in max value



Output: trace back alignment from final cell $A[n,m]$

Needleman-Wunsch algorithm example

Reichman University
Efi Arazi School of Computer Science

Input: S = GCATCGATTCCGAGC T = GCCATGATGAAC

A:

		G	C	C	A	T	G	A	T	G	A	A	C
	0	-3	-6	-9	-12								
G	-3	2											
C	-6												
A	-9												
T	-12												
C													
G													
A													
T													
T													
C													
C													
G													
A													
G													
C													

Score function:

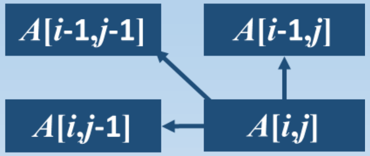
$\sigma(x,x) = 2$

$\sigma(x,y) = -2 \ (x \neq y)$

$\sigma(x,-) = \sigma(-,x) = -3$

Update step:

$$A[i,j] = \max\{ \begin{array}{l} A[i-1,j-1]+\sigma(S_i, T_j) \\ A[i-1,j]+\sigma(S_i, -) \\ A[i,j-1]+\sigma(-, T_j) \end{array} \}$$



Needleman-Wunsch algorithm example

There is an error in this matrix ☹

Input: S = GCATCGATTCCGAGC T = GCCATGATGAAC

A:

		G	C	C	A	T	G	A	T	G	A	A	C
	0	-3	-6	-9	-12	-15	-18	-21	-24	-27	-30	-33	-36
G	-3	2	-1	-4	-7	-10	-13	-16	-19	-22	-25	-28	-31
C	-6	-1	4	1	-2	-5	-8	-11	-14	-17	-20	-23	-26
A	-9	-4	1	2	3	0	-3	-6	-9	-12	-15	-18	-21
T	-12	-7	-2	-1	0	5	2	-1	-4	-7	-10	-13	-16
C	-15	-10	-5	0	-3	-2	3	0	-3	-6	-9	-12	-11
G	-18	-13	-8	-3	-6	-5	0	1	-2	-1	-4	-7	-10
A	-21	-16	-11	-6	-1	-4	-7	2	-1	-4	1	-2	-5
T	-24	-19	-14	-9	-4	1	-2	-5	4	1	-2	-1	-4
T	-27	-22	-17	-12	-7	-2	-1	-4	1	2	-1	-4	-3
C	-30	-25	-20	-15	-10	-5	-4	-3	-2	-5	0	-3	-2
C	-33	-28	-23	-18	-13	-8	-7	-6	-5	-8	-7	-2	-1
G	-36	-31	-26	-21	-16	-11	-10	-9	-8	-3	-6	-5	-4
A	-39	-34	-29	-24	-19	-14	-13	-8	-5	-6	-1	-4	-7
G	-42	-37	-32	-27	-22	-17	-16	-11	-8	-3	-6	-3	-6
C	-45	-40	-35	-30	-25	-20	-19	-14	-11	-6	-5	-6	-1

Score function:

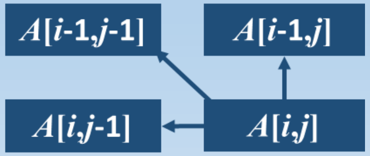
$\sigma(x,x) = 2$

$\sigma(x,y) = -2 \quad (x \neq y)$

$\sigma(x,-) = \sigma(-,x) = -3$

Update step:

$$A[i,j] = \max\left\{ \begin{array}{l} A[i-1,j-1]+\sigma(S_i, T_j) \\ A[i-1,j]+\sigma(S_i, -) \\ A[i,j-1]+\sigma(-, T_j) \end{array} \right\}$$



Needleman-Wunsch algorithm example

Reichman University — Efi Arazi School of Computer Science

Input: S = GCATCGATTCCGAGC T = GCCATGATGAAC

A:

		G	C	C	A	T	G	A	T	G	A	A	C
	0	-3	-6	-9	-12	-15	-18	-21	-24	-27	-30	-33	-36
G	-3	2	-1	-4	-7	-10	-13	-16	-19	-22	-25	-28	-31
C	-6	-1	4	1	-2	-5	-8	-11	-14	-17	-20	-23	-26
A	-9	-4	1	2	3	0	-3	-6	-9	-12	-15	-18	-21
T	-12	-7	-2	-1	0	5	2	-1	-4	-7	-10	-13	-16
C	-15	-10	-5	0	-3	2	3	0	-3	-6	-9	-12	-11
G	-18	-13	-8	-3	-6	-5	0	1	-2	-1	-4	-7	-10
A	-21	-16	-11	-6	1	-4	-7	2	-1	-4	1	-2	-5
T	-24	-19	-14	-9	-4	1	-2	-5	4	1	-2	-1	-4
T	-27	-22	-17	-12	-7	-2	1	-4	1	2	-1	-4	-3
C	-30	-25	-20	-15	-10	-5	-4	3	2	-5	0	-3	-2
C	-33	-28	-23	-18	-13	-8	-7	-6	5	-8	-7	-2	-1
G	-36	-31	-26	-21	-16	-11	-10	-9	-8	3	-6	-5	-4
A	-39	-34	-29	-24	-19	-14	-13	-8	-5	-6	1	-4	-7
G	-42	-37	-32	-27	-22	-17	-16	-11	-8	-3	-6	3	-6
C	-45	-40	-35	-30	-25	-20	-19	-14	-11	-6	-5	-6	1

Paths in the DP matrix correspond to alignments

Optimal alignments:

S' = GCATCGATTCCGAGC
T' = GC--C-ATGATGAAC

S' = GCATCGATTCCGAGC
T' = GCCATGAT---GAAC

There is an error in this matrix ☹
There is a better alignment !!

Needleman-Wunsch algorithm – complexity

Reichman University Efi Arazi School of Computer Science

Space:

$\boldsymbol{O(mn)}$ **–** matrix A has mn cells and for each cell we hold a number and a pointer

Time:

$\boldsymbol{O(mn)}$ **–** computing the value in each cell involves three arithmetic operations and maximization. The traceback operation at the end (recovering the path in the matrix) takes $\boldsymbol{O(m+n)}$ additional steps.

Food for thought:

How does the scoring scheme affect the global alignment:

1. Does scaling all scores by a positive multiplicative factor change the optimal alignment?
   $\sigma'(x,y) = a \times \sigma(x,y)$ $(a > 0)$

No. The score of alignments is just multiplied by a constant factor of a:

$$\sigma'(\mathrm{A}) = \sum_{i=1}^{len(\mathrm{A})} \sigma'(\mathrm{A}_i) = \sum_{i=1}^{len(\mathrm{A})} a \times \sigma(\mathrm{A}_i) = a \times \sum_{i=1}^{len(\mathrm{A})} \sigma(\mathrm{A}_i) = a \times \sigma(\mathrm{A})$$

So, if A and A' are two alignments and A has higher score under σ: $\sigma(\mathrm{A}) > \sigma(\mathrm{A}')$,

then and A also has higher score under σ': $\sigma'(\mathrm{A}) = a \times \sigma(\mathrm{A}) > a \times \sigma(\mathrm{A}') = \sigma'(\mathrm{A}')$

Food for thought:

How does the scoring scheme affect the global alignment:

2. Does shifting all scores by an additive factor change the optimal alignment?
   $\sigma'(x,y) = a + \sigma(x,y)$

[See Problem 2 in HW #1]

Food for thought:

Reichman University Efi Arazi School of Computer Science

How does the scoring scheme affect the global alignment:

3. In biologically-motivated scoring schemes, it is common to assume that the gap score is smaller than half of the mismatch score: $\sigma(x,-) < \frac{1}{2}\sigma(x,y)$.
   Why does this make sense?

[See Problem 2 in HW #1]

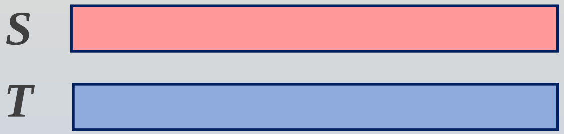
Lecture overview

- Formulation of the alignment problem
- An efficient algorithm for global alignment
- An efficient algorithm for local alignment ←
- Other variants of the alignment problem

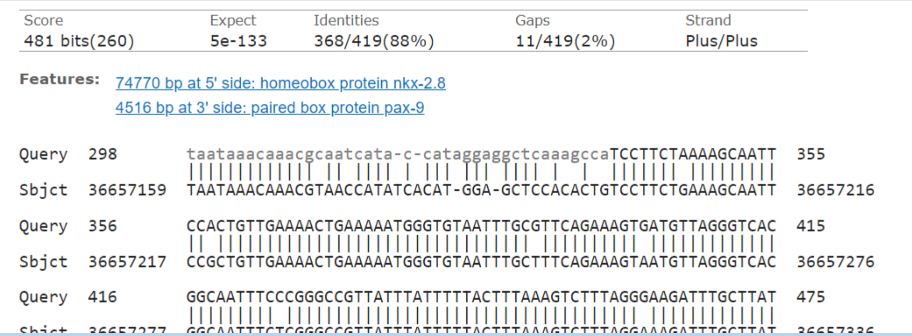
Global vs. local alignment

Global alignment:

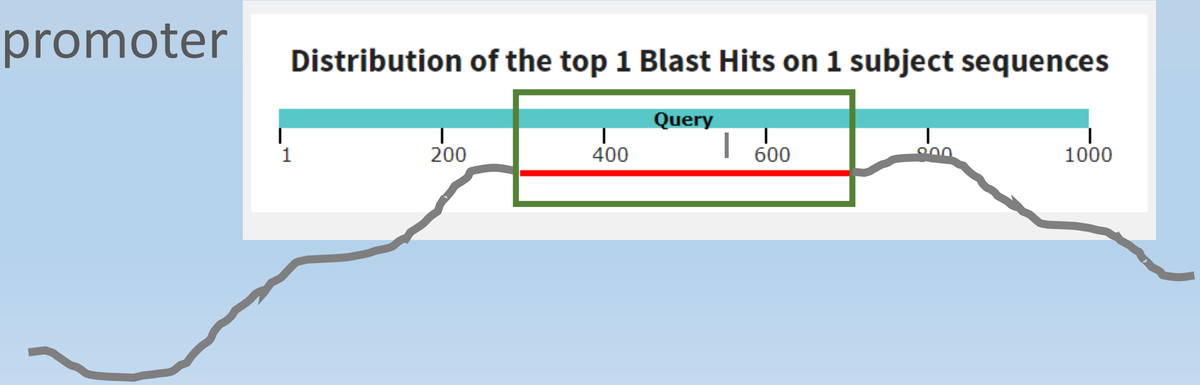
- Find the best scoring alignment between two sequences
- Allows us to quantify the level of evolutionary match (homology) between sequences



However, in many cases we are interested in finding segments of the two sequences that have a good match



Recall the demonstration we did for the 1000 base promoter of the PAX9 gene:

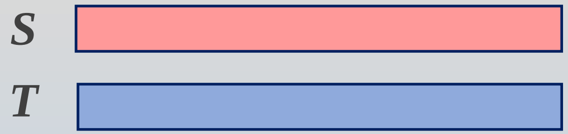


Global vs. local alignment

Reichman University Efi Arazi School of Computer Science

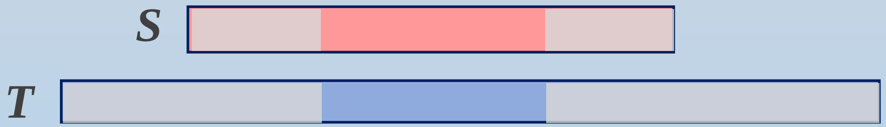
Global alignment:

- Find the best scoring alignment between two sequences
- Allows us to quantify the level of evolutionary match (homology) between sequences



Local alignment:

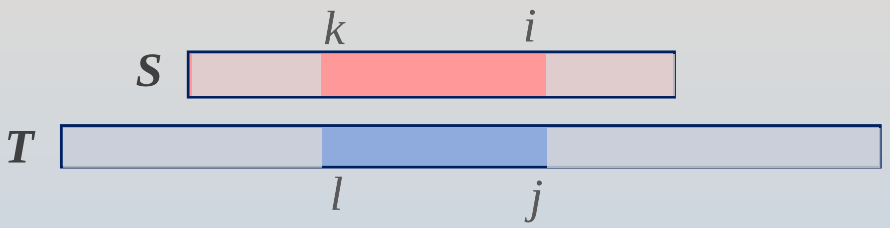
- Find the best scoring alignment of **subsequences** of two given sequences
- Allows us to detect possible regions of homology between two (long) sequences



Local alignment

Local alignment:

- Find the best scoring alignment of **subsequences** of two given sequences
- Allows us to detect possible regions of homology between two (long) sequences



Formulation:

Input: two sequences $S_{1..n}$, $T_{1..m}$ over the same alphabet Σ

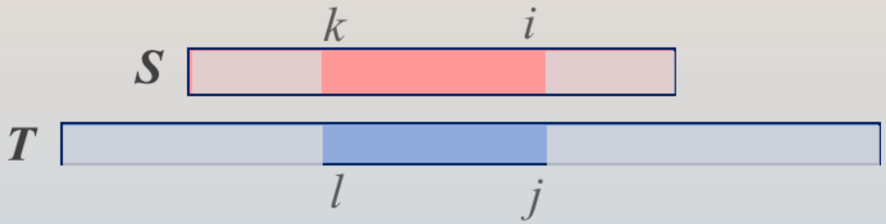
Output: two sequences S' and T' over the alphabet $\Sigma \cup \{-\}$ that satisfy:

- Removing the gap labels ('-') from S' and T' gives $S_{k..i}$ and $T_{l..j}$ for some $1 \leq k \leq i \leq n$ and $1 \leq l \leq j \leq m$.
- S' and T' have the same length.

Maximum score local alignment

Input: two sequences $S_{1..n}$, $T_{1..m}$ over the same alphabet Σ

+ score function σ: $(\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \rightarrow R$



Output: local alignment $(S'_{k..i}, T'_{l..j})$ with maximum score (sum of column scores)

Example:

S = AGTTCTTGCGCATCGATTCCGAGCAGGCGTAAT

T = AGTCCTTGCGCCATGATGAACAGGCTTAAT

$\sigma(x,x) = 2$

$\sigma(x,y) = -2$ $(x \neq y)$

$\sigma(x,-) = \sigma(-,x) = -3$

Possible local alignments:

```
AGTTCTTGCGC
AGTCCTTGCGC
```

$\sigma(S'_{1,11}, T'_{1,11}) = 18$

```
C-ATCGATTCCG
CCAT-GAT---G
```

$\sigma(S'_{11,21}, T'_{11,17}) = -1$

```
GAGCAGGCGTAAT
GAACAGGCTTAAT
```

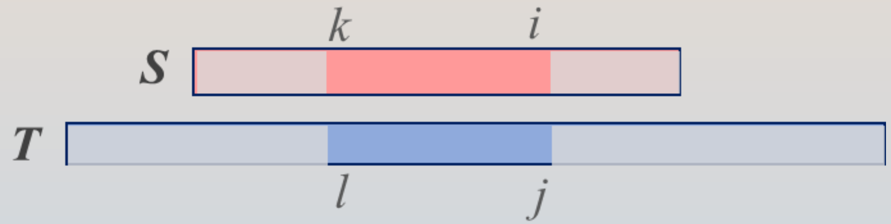
$\sigma(S'_{21,33}, T'_{17,29}) = 18$

Maximum score local alignment

Reichman University · Efi Arazi School of Computer Science

Input: two sequences $S_{1..n}$, $T_{1..m}$ over the same alphabet Σ

+ score function σ: $(\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \rightarrow R$



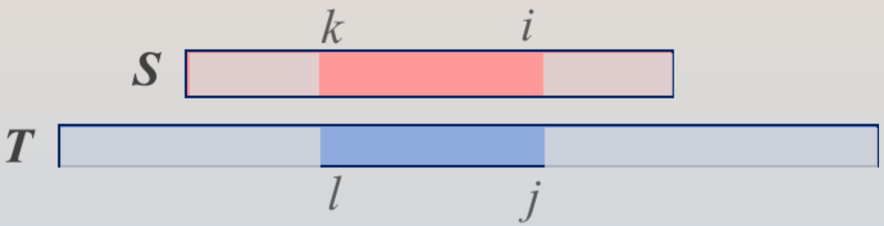
Output: local alignment $(S'_{k..i}, T'_{l..j})$ with maximum score (sum of column scores)

1st try: local alignment is like global alignment for sub-sequences

- Run the NW algorithm for global alignment for every two subsequences of S and T ($\frac{1}{4}m(m+1)n(n+1)$ times)
- Return the alignment with maximum score
- Complexity: $O(m^3n^3)$

Maximum score local alignment

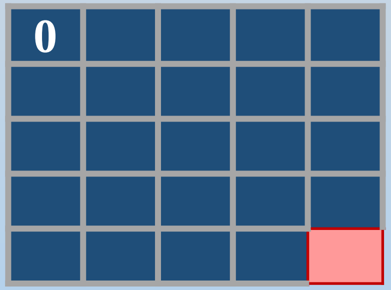
Input: two sequences $S_{1..n}$, $T_{1..m}$ over the same alphabet Σ
+ score function σ: $(\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \rightarrow R$



Output: local alignment $(S'_{k..i}, T'_{l..j})$ with maximum score (sum of column scores)

2nd try: the NW DP matrix represents optimal alignments for all prefixes of S and T

- Run NW for all suffixes $S_{k..n}$ and $T_{l..m}$ ($\boldsymbol{mn}$ times)
- In each run (k, l) keep the alignment starting from the cell (i, j) in the DP matrix with highest score (of $S_{k..k+i}$ and $T_{l..l+j}$)
- Return the alignment with highest score across all runs
- Complexity: $\boldsymbol{O(m^2n^2)}$



Can we do better by defining a DP matrix for max score alignment of suffixes?

Revisiting the Needleman-Wunsch DP matrix:

We would like to redefine the DP matrix s.t. $A[i,j]$ represents the max-score of any alignment of a suffix of $S_{1..i}$ and a suffix of $T_{1..j}$ (denote as alignment ending at [i,j])

To do that we have to modifying the recursive claim:

If an alignment has maximum score of all alignments that end at [i,j], then one of the following must hold:

- Last column (S_i ,T_j) preceded by a max-score alignment ending at [i-1 , j-1]
- Last column (- ,T_j) preceded by a max-score alignment ending at [i , j-1]
- Last column (S_i ,-) preceded by a max-score alignment ending at [i-1 , j]
- The alignment is empty (score 0) ← **added case !!**

➔ $A[i,j] = \max\{ A[i-1,j-1]+\sigma(S_i ,T_j) \;;\; A[i-1,j] +\sigma(S_i ,-) \;;\; A[i,j-1] +\sigma(- ,T_j) \}$

Revisiting the Needleman-Wunsch DP matrix:

We would like to redefine the DP matrix s.t. $A[i,j]$ represents the max-score of any alignment of a suffix of $S_{1..i}$ and a suffix of $T_{1..j}$ (denote as alignment ending at [i,j])

To do that we have to modifying the recursive claim:

If an alignment has maximum score of all alignments that end at [i,j], then one of the following must hold:

- Last column (S_i, T_j) preceded by a max-score alignment ending at $[i-1, j-1]$
- Last column $(-, T_j)$ preceded by a max-score alignment ending at $[i, j-1]$
- Last column $(S_i, -)$ preceded by a max-score alignment ending at $[i-1, j]$
- The alignment is empty (score 0) **← added case !!**

Proof: similar to proof of recursive claim for global alignment **[left as self exercise]**

$$\Rightarrow A[i,j] = \max\{ 0 \; ; A[i-1,j-1]+\sigma(S_i, T_j) \; ; A[i-1,j]+\sigma(S_i, -) \; ; A[i,j-1]+\sigma(-, T_j) \}$$

Smith-Waterman algorithm for optimal local alignment

Smith , T. F. and Waterman, M.S. (*Jour Mol Biol* 1970)

Input: two sequences $S_{1..n}$, $T_{1..m}$ over the same alphabet Σ
scoring function: $\sigma: (\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \rightarrow R$

Objective: Compute a matrix A s.t. $A[i,j]$ holds the max score alignment of a suffix of $S_{1..i}$ and a suffix of $T_{1..j}$

Initialization:

- Create empty matrix A with rows indexed $0..n$ and columns indexed $0..m$
- Initialize $A[i,0] = A[0, j] = 0$ and for each i=0..n and j=0.. n

(assuming non-positive scores for gaps)

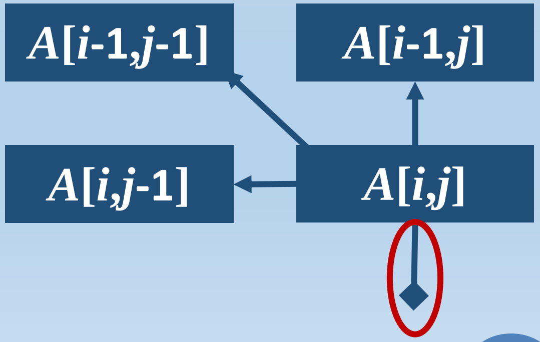
0	0	0	0	0
0				
0				
0				
0				

Main loop: (ascending order of columns/rows)

- For each i=1..n and j=1..m compute $A[i,j]$ as follows:

$$A[i,j] = \max\{0 ; A[i-1,j-1]+\sigma(S_i ,T_j) ; A[i-1,j] +\sigma(S_i ,-) ; A[i,j-1] +\sigma(- ,T_j) \}$$

Keep a pointer to the cell that results in max value



Output: find cell $A[i,j]$ with maximum score and trace back from there

Smith-Waterman algorithm – example

A:

		G	C	C	A	T	G	A	T	G	A	A	C
	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	2	0	0	0	0	2	0	0	2	0	0	0
C	0	0	4	2	0	0	0	0	0	0	0	0	2
A	0	0	1	2	4	1	0	2	0	0	2	2	0
T	0	0	0	0	1	6	3	0	4	1	0	0	0
C	0	0	2	2	0	3	4	1	1	2	0	0	2
G	0	2	0	0	0	0	5	2	0	3	0	0	0
A	0	0	0	0	2	0	2	7	4	1	5	2	0
T	0	0	0	0	0	4	1	4	9	6	3	3	0
T	0	0	0	0	0	2	2	1	6	7	4	1	1
C	0	0	2	2	0	0	0	0	3	4	5	2	3
C	0	0	2	4	1	0	0	0	0	1	2	3	4
G	0	2	0	1	2	0	2	0	0	2	0	0	1
A	0	0	0	0	3	0	0	4	1	0	4	2	0
G	0	2	0	0	0	1	2	1	2	3	1	2	0
C	0	0	4	2	0	0	0	0	0	0	1	0	4

Score function:

$\sigma(x,x) = 2$

$\sigma(x,y) = -2 \quad (x \neq y)$

$\sigma(x,-) = \sigma(-,x) = -3$

Update step:

$$A[j,l] = \max\begin{cases} 0 \\ A[j-1,l-1]+\sigma(S_j,T_l) \\ A[j-1,l]+\sigma(S_i,-) \\ A[j,l-1]+\sigma(-,T_l) \end{cases}$$

Smith-Waterman algorithm – example

A:

		G	C	C	A	T	G	A	T	G	A	A	C
	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	2	0	0	0	0	2	0	0	2	0	0	0
C	0	0	4	2	0	0	0	0	0	0	0	0	2
A	0	0	1	2	4	1	0	2	0	0	2	2	0
T	0	0	0	0	1	6	3	0	4	1	0	0	0
C	0	0	2	2	0	3	4	1	1	2	0	0	2
G	0	2	0	0	0	0	5	2	0	3	0	0	0
A	0	0	0	0	2	0	2	7	4	1	5	2	0
T	0	0	0	0	0	4	1	4	9	6	3	3	0
T	0	0	0	0	0	2	2	1	6	7	4	1	1
C	0	0	2	2	0	0	0	0	3	4	5	2	3
C	0	0	2	4	1	0	0	0	0	1	2	3	4
G	0	2	0	1	2	0	2	0	0	2	0	0	1
A	0	0	0	0	3	0	0	4	1	0	4	2	0
G	0	2	0	0	0	1	2	1	2	3	1	2	0
C	0	0	4	2	0	0	0	0	0	0	1	0	4

- Find top score in matrix
- Trace back path and alignment

S' = CATCGAT
T' = CAT–GAT

Smith-Waterman algorithm – complexity

Space:

$O(mn)$ – matrix A has mn cells and for each cell we hold a number and a pointer

Time:

$O(mn)$ – computing the value in each cell involves three arithmetic operations and maximization. The traceback operation at the end (recovering the path in the matrix) takes $O(m+n)$ additional steps.

Same as Needleman-Wunsch algorithm for global alignment

Food for thought (II):

Reichman University Efi Arazi School of Computer Science

How does the scoring scheme affect the local alignment:

1. Does scaling all scores by a positive multiplicative factor change the optimal alignment? $\sigma'(x,y) = a \times \sigma(x,y)$ $(a > 0)$

No. The score of alignments is just multiplied by a constant factor of *a*

(same arguments for global alignment apply here)

Food for thought (II):

How does the scoring scheme affect the local alignment:

2. Does shifting all scores by an additive factor change the optimal alignment?
   $\sigma'(x,y) = a + \sigma(x,y)$

[See Problem 2 in HW #1]

Food for thought (II):

How does the scoring scheme affect the local alignment:

3. In scoring schemes for local alignment, it's typically assumed that some pairs have a positive score and others have a negative score. Why does this make sense?

- If all pairs have a negative score, then the optimal local alignment is an empty alignment (with score 0). Any other alignment has a negative score.
- If all pairs have a positive score, then the optimal local alignment is a global alignment. This is because we never discard "overhangs", since they can be arbitrarily aligned and contribute positively to the score.

Lecture overview

- Formulation of the alignment problem
- An efficient algorithm for global alignment
- An efficient algorithm for local alignment
- Other variants of the alignment problem ←

  to be continued… next week!