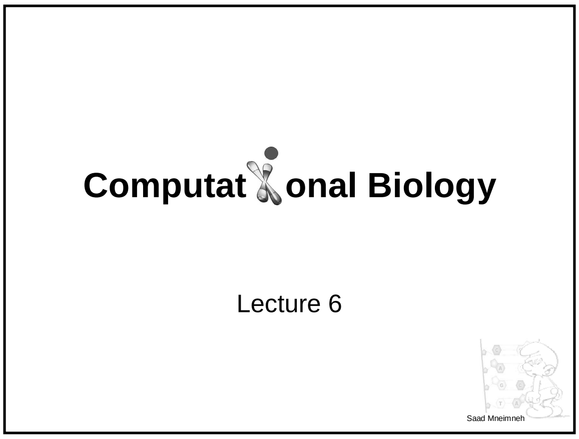
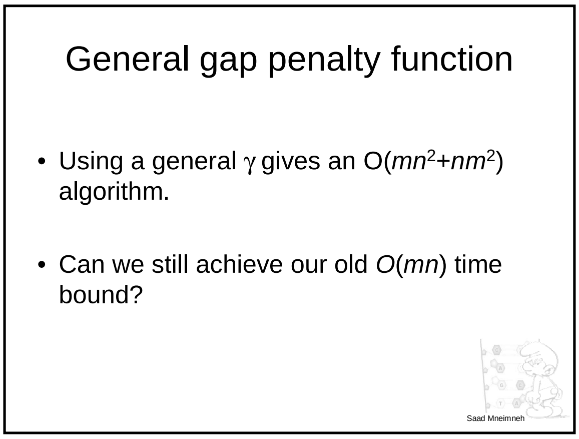


Computational Biology

Lecture 6

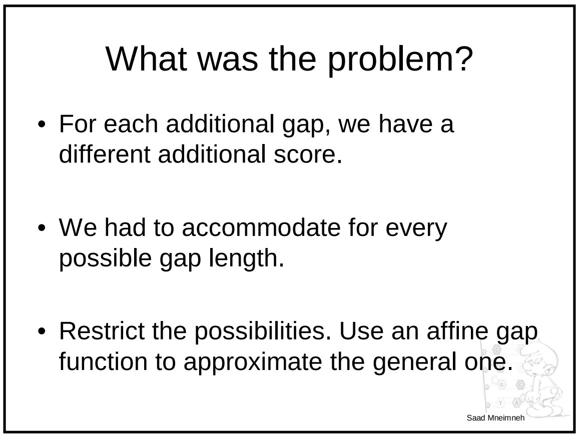
Saad Mneimneh



General gap penalty function

- Using a general γ gives an O(mn^2+nm^2) algorithm.
- Can we still achieve our old $O(mn)$ time bound?

Saad Mneimneh

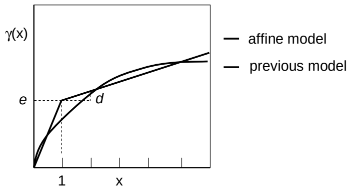


What was the problem?

- For each additional gap, we have a different additional score.
- We had to accommodate for every possible gap length.
- Restrict the possibilities. Use an affine gap function to approximate the general one.

Saad Mneimneh

Affine gap penalty function



$$\gamma(x) = e + d(x - 1) \quad x \geq 1.$$

How can this help

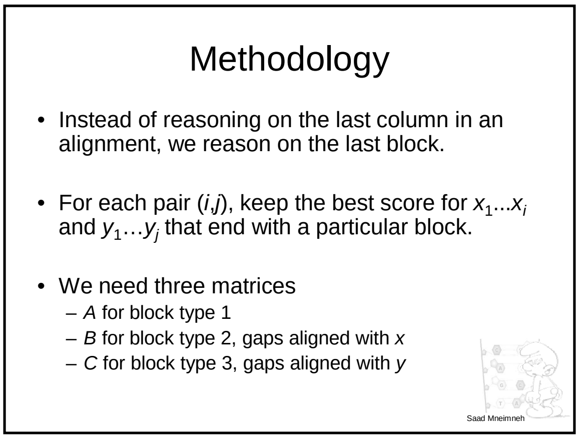
- Any gap length greater than one is penalized linearly.
- Need to distinguish only between 1 and more than 1.
- If 1
  - penalize by *e*.
- If $x > 1$
  - penalize linearly by increments of *d*.
- Use more than one matrix to detect that, depends on where we stopped last time.

Looking for additivity

- Break the alignment into 3 kinds of blocks:
  - Two aligned characters
  - Consecutive gaps aligned with *x*
  - Consecutive gaps aligned with *y*

```
A|A|C|---|A|ATTCCG|A|C|T|AC
A|C|T|ACC|T|------|C|G|C|--
```

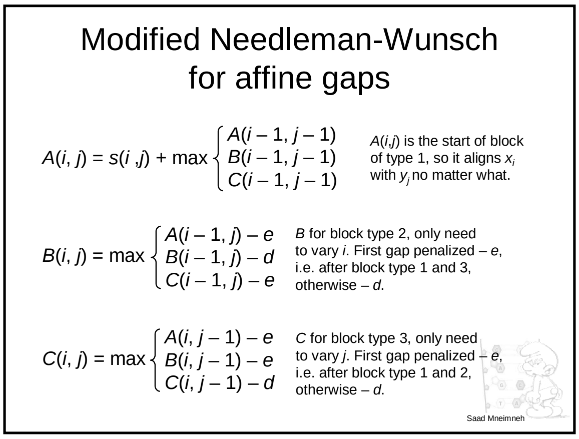
- Score is additive across block boundaries, this is true regardless of the gap penalty function.



Methodology

- Instead of reasoning on the last column in an alignment, we reason on the last block.
- For each pair (i,j), keep the best score for $x_1 \ldots x_i$ and $y_1 \ldots y_j$ that end with a particular block.
- We need three matrices
  - *A* for block type 1
  - *B* for block type 2, gaps aligned with *x*
  - *C* for block type 3, gaps aligned with *y*

Saad Mneimneh



Modified Needleman-Wunsch for affine gaps

$$A(i, j) = s(i, j) + \max \begin{cases} A(i-1, j-1) \\ B(i-1, j-1) \\ C(i-1, j-1) \end{cases}$$

$A(i,j)$ is the start of block of type 1, so it aligns x_i with y_j no matter what.

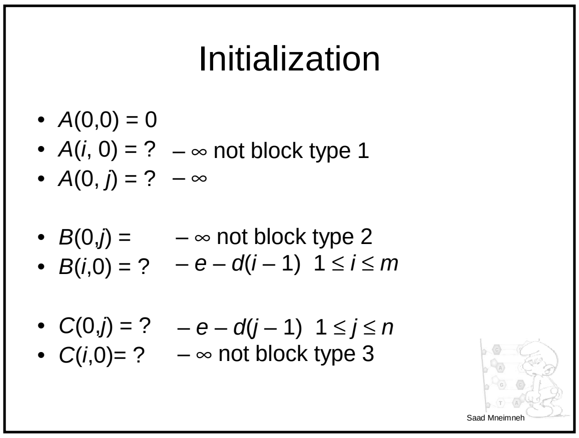
$$B(i, j) = \max \begin{cases} A(i-1, j) - e \\ B(i-1, j) - d \\ C(i-1, j) - e \end{cases}$$

B for block type 2, only need to vary *i*. First gap penalized – *e*, i.e. after block type 1 and 3, otherwise – *d*.

$$C(i, j) = \max \begin{cases} A(i, j-1) - e \\ B(i, j-1) - e \\ C(i, j-1) - d \end{cases}$$

C for block type 3, only need to vary *j*. First gap penalized – *e*, i.e. after block type 1 and 2, otherwise – *d*.

Saad Mneimneh



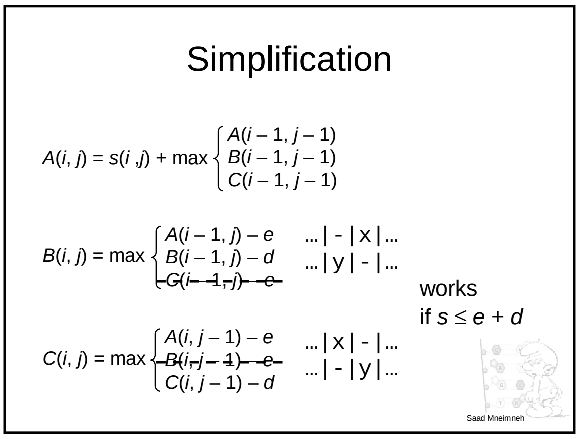
Initialization

- $A(0,0) = 0$
- $A(i, 0) = ?$ $-\infty$ not block type 1
- $A(0, j) = ?$ $-\infty$

- $B(0,j) =$ $-\infty$ not block type 2
- $B(i,0) = ?$ $-e - d(i-1)$ $1 \le i \le m$

- $C(0,j) = ?$ $-e - d(j-1)$ $1 \le j \le n$
- $C(i,0) = ?$ $-\infty$ not block type 3

Saad Mneimneh



Simplification

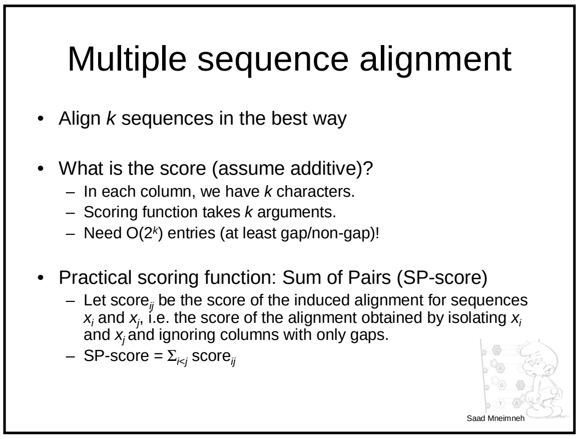
$$A(i, j) = s(i, j) + \max \begin{cases} A(i-1, j-1) \\ B(i-1, j-1) \\ C(i-1, j-1) \end{cases}$$

$$B(i, j) = \max \begin{cases} A(i-1, j) - e & \ldots|\,\text{-}\,|\,\text{x}\,|\ldots \\ B(i-1, j) - d & \ldots|\,\text{y}\,|\,\text{-}\,|\ldots \\ \cancel{C(i-1, j) - e} & \end{cases}$$

works
if $s \leq e + d$

$$C(i, j) = \max \begin{cases} A(i, j-1) - e & \ldots|\,\text{x}\,|\,\text{-}\,|\ldots \\ \cancel{B(i, j-1) - e} & \ldots|\,\text{-}\,|\,\text{y}\,|\ldots \\ C(i, j-1) - d & \end{cases}$$

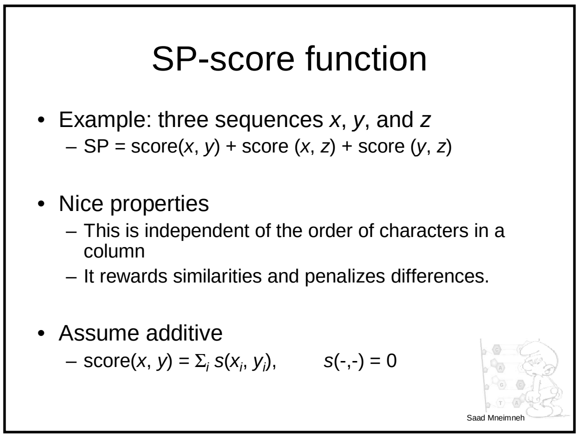
Saad Mneimneh



Multiple sequence alignment

- Align *k* sequences in the best way
- What is the score (assume additive)?
  - In each column, we have *k* characters.
  - Scoring function takes *k* arguments.
  - Need O(2^k) entries (at least gap/non-gap)!
- Practical scoring function: Sum of Pairs (SP-score)
  - Let score_{ij} be the score of the induced alignment for sequences x_i and x_j, i.e. the score of the alignment obtained by isolating x_i and x_j and ignoring columns with only gaps.
  - SP-score = $\Sigma_{i<j}\, \text{score}_{ij}$

Saad Mneimneh



SP-score function

- Example: three sequences *x*, *y*, and *z*
  - SP = score(*x*, *y*) + score (*x*, *z*) + score (*y*, *z*)
- Nice properties
  - This is independent of the order of characters in a column
  - It rewards similarities and penalizes differences.
- Assume additive
  - score(x, y) = $\Sigma_i\, s(x_i, y_i)$, s(-,-) = 0

Saad Mneimneh

Example

- w = ATG, x = ATG, y = A, z = T

```
ATG
ATG
A--                A-
-T-                 -T
```

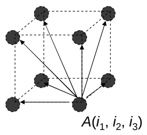
- Score = 3 -3 -3 -3 -3 -4 = - 13
- The *induced* alignment between a pair of sequences is not necessarily an optimal one, e.g. y and z.

Dynamic Programming

- k sequences of length n_i each.
- k dimensional array A of length $n_i + 1$ in each direction.
- $A(i_1, \ldots, i_k)$ holds the score of the optimal alignment involving $x_1[1...i_1], \ldots, x_k[1...i_k]$.
- A now requires $O(n^k)$ space.

Dynamic Programming (cont.)

- Now each entry in A depends on $2^k - 1$ entries (note: $k = 2$ => 3) (why?)
  - e.g. $A(i_1, i_2, i_3)$ depends on:



$A(i_1, i_2, i_3 - 1)$	001
$A(i_1, i_2 - 1, i_3)$	010
$A(i_1, i_2 - 1, i_3 - 1)$	011
$A(i_1 - 1, i_2, i_3)$	100
$A(i_1 - 1, i_2, i_3 - 1)$	101
$A(i_1 - 1, i_2 - 1, i_3)$	110
$A(i_1 - 1, i_2 - 1, i_3 - 1)$	111

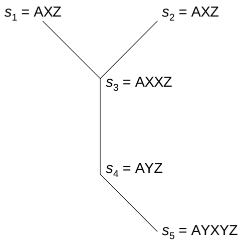
- Computing the SP-score in each case requires $O(k^2)$ time.
- Total running time is $O(k^2 2^k n^k)$

Multiple sequence alignment is NP-complete

Heuristic: Star Alignment

- Star alignment is a special case of tree alignments.
- What is a tree alignment?
- Given a tree with *k* nodes representing *k* sequences, a multiple alignment of the *k* sequences *consistent* with the tree is such that the induced alignment between x_i and x_j is optimal if there is an edge (x_i, x_j).

Example

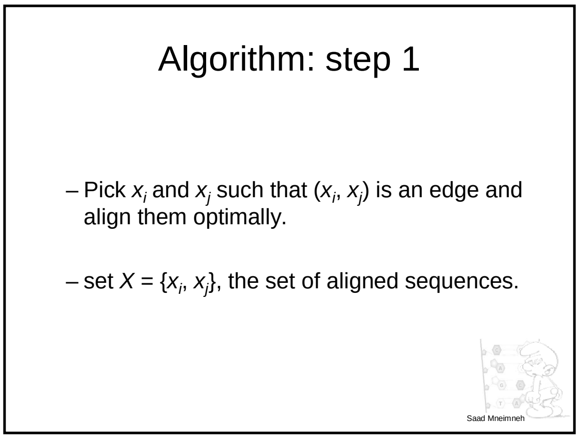


A multiple alignment consistent with the tree

s_1	A	X	-	-	Z
s_2	A	-	X	-	Z
s_3	A	X	X	-	Z
s_4	A	Y	-	-	Z
s_5	A	y	X	X	Z

Tree alignment

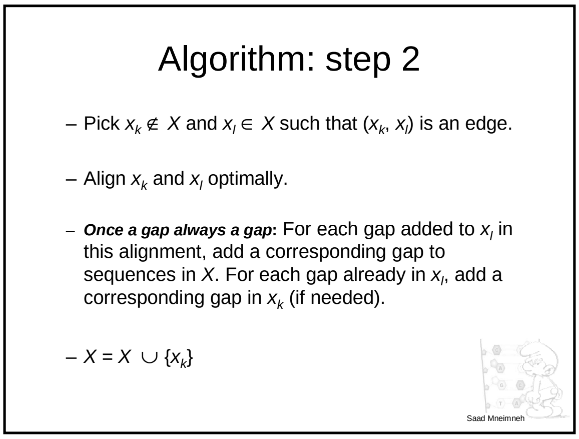
- Given a tree, is it always possible to obtain a multiple alignment consistent with the tree?
  - YES
- How?



Algorithm: step 1

- Pick x_i and x_j such that (x_i, x_j) is an edge and align them optimally.
- set $X = \{x_i, x_j\}$, the set of aligned sequences.

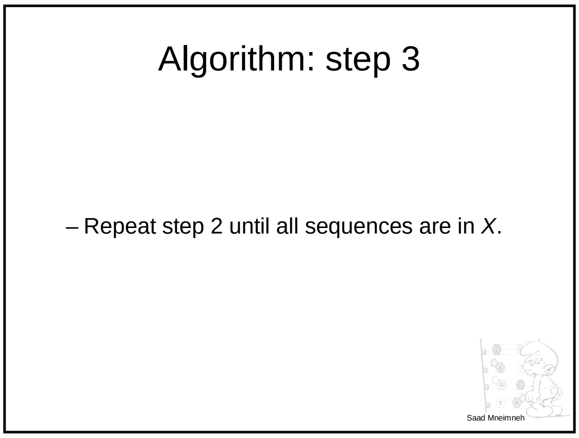
Saad Mneimneh



Algorithm: step 2

- Pick $x_k \notin X$ and $x_l \in X$ such that (x_k, x_l) is an edge.
- Align x_k and x_l optimally.
- ***Once a gap always a gap*:** For each gap added to x_l in this alignment, add a corresponding gap to sequences in X. For each gap already in x_l, add a corresponding gap in x_k (if needed).
- $X = X \cup \{x_k\}$

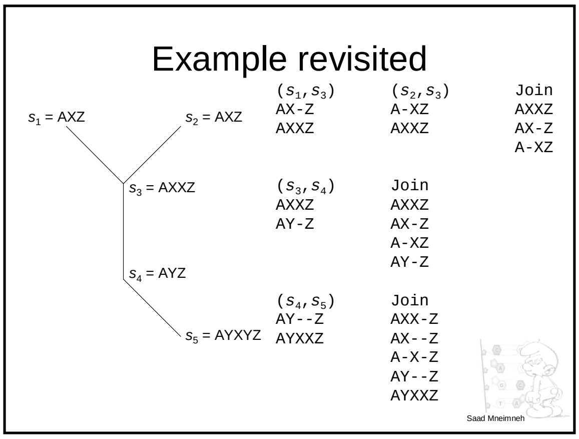
Saad Mneimneh



Algorithm: step 3

- Repeat step 2 until all sequences are in X.

Saad Mneimneh



Example revisited

s_1 = AXZ

s_2 = AXZ

s_3 = AXXZ

s_4 = AYZ

s_5 = AYXYZ

(s_1,s_3)
```
AX-Z
AXXZ
```

(s_2,s_3)
```
A-XZ
AXXZ
```

Join
```
AXXZ
AX-Z
A-XZ
```

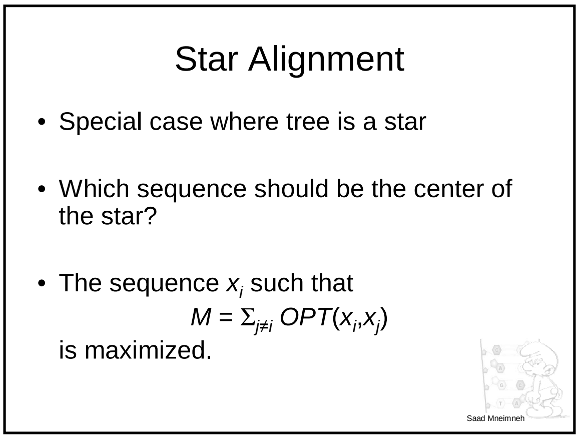
(s_3,s_4)
```
AXXZ
AY-Z
```

Join
```
AXXZ
AX-Z
A-XZ
AY-Z
```

(s_4,s_5)
```
AY--Z
AYXXZ
```

Join
```
AXX-Z
AX--Z
A-X-Z
AY--Z
AYXXZ
```

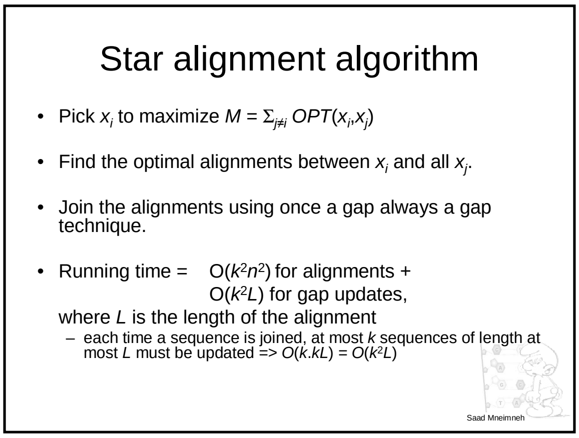
Saad Mneimneh



Star Alignment

- Special case where tree is a star
- Which sequence should be the center of the star?
- The sequence x_i such that
$$M = \Sigma_{j \neq i}\ OPT(x_i,x_j)$$
is maximized.

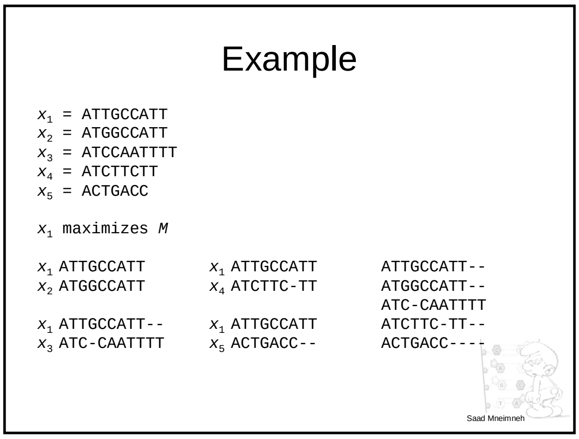
Saad Mneimneh



Star alignment algorithm

- Pick x_i to maximize $M = \Sigma_{j \neq i}\ OPT(x_i,x_j)$
- Find the optimal alignments between x_i and all x_j.
- Join the alignments using once a gap always a gap technique.
- Running time = O(k^2n^2) for alignments +
O(k^2L) for gap updates,
where L is the length of the alignment
  - each time a sequence is joined, at most k sequences of length at most L must be updated => $O(k.kL) = O(k^2L)$

Saad Mneimneh



Example

x_1 = ATTGCCATT
x_2 = ATGGCCATT
x_3 = ATCCAATTTT
x_4 = ATCTTCTT
x_5 = ACTGACC

x_1 maximizes M

x_1 ATTGCCATT	x_1 ATTGCCATT	ATTGCCATT--
x_2 ATGGCCATT	x_4 ATCTTC-TT	ATGGCCATT--
		ATC-CAATTTT
x_1 ATTGCCATT--	x_1 ATTGCCATT	ATCTTC-TT--
x_3 ATC-CAATTTT	x_5 ACTGACC--	ACTGACC----

Saad Mneimneh