

# Pre-class assignment

- Submit a PDF (not word doc, please), preferably via Slack
  - [YYYY-MM-DD]\_[LastName]\_report.pdf
- No need for elaborate headings at the beginning of the report.
  - [Name - Due-Date - Paper title] is enough.
  - The point is to not encroach Page1.
- Page settings:
  - Font: Arial 10pt or Times New Roman 11pt.
  - Margins & spacing: 1-inch margins on all sides; 1 or 1.15 spacing between lines.
- Read the paper & write the report critically but your report is not a peer-review.

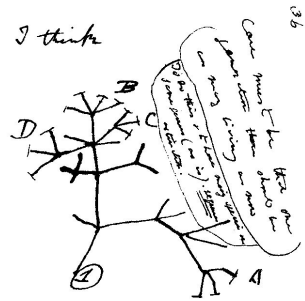
# Paper discussion

- Use other resources beyond the paper to illustrate the problem.
  - Provide citation on the slide.
- Split complex multi-part figures into several slides.
  - Capture/zoom-into parts of figure & annotate with boxes/arrows.
- Convert methods to flowcharts & annotate flowcharts.
  - You're welcome to draw neatly on paper & photograph it.
- Be prepared to define phrases/terminology on your slide.
  - Google it, read-up; No issues if it's still unclear. Please raise it in class & we can discuss.
- Look-up other papers (esp. recent ones that cite this paper to see what's said).
  - Use that to final discussion about limitations & future directions.

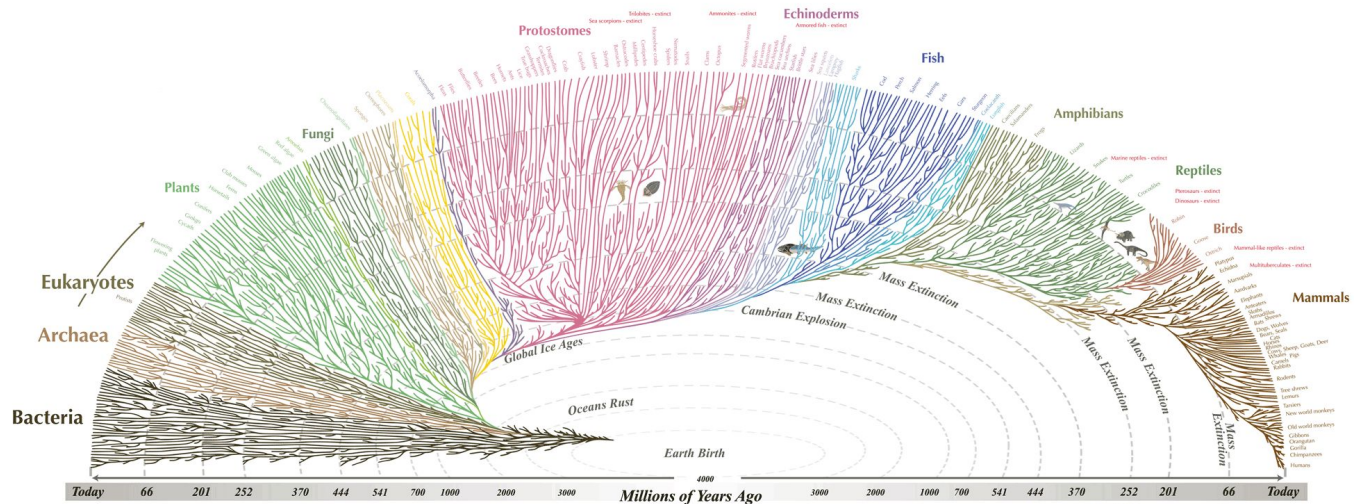
# Lecture 4: Sequence alignment

- Global alignment
  - Dynamic programming
  - Needleman-Wunsch algorithm
- Local alignment
  - Smith-Waterman algorithm
  - BLAST

# Sequence evolution



Then between A & B. various  
 loss of relation. C & B. the  
 first predation, B & D  
 rather greater distance than  
 then former would have  
 formed. - heavy relation



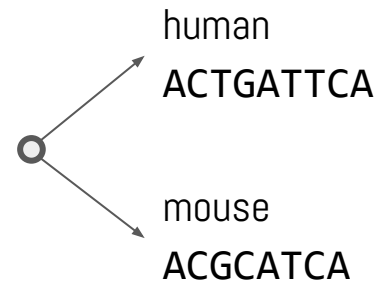
All the major and many of the minor living branches of life are shown on this diagram, but only a few of those that have gone extinct are shown. Example: Dinosaurs - extinct

© 2008, 2017 Leonard Eisenberg. All rights reserved.  
 evo4gram.com

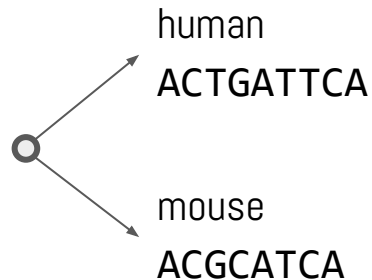
deletion      mutation      insertion

ACATGGTCA → AC\*TGGTCA → ACTGATCA → ACTGATTCA

Evolutionary time



# Sequence evolution



Sequences can be aligned by allowing for **gaps** and **mismatches**.

ACTGATTCA

ACGCA-TCA

ACTGATTCA

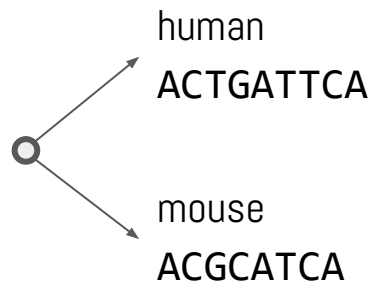
AC-GCATCA

ACTG-ATTCA

AC-GCAT-CA

Which alignment is correct?

# Sequence evolution



Sequences can be aligned by allowing for **gaps** and **mismatches**.

ACTGATTCA

ACGCA-TCA

ACTGATTCA

AC-GCATCA

ACTG-ATTCA

AC-GCAT-CA

Which alignment is correct?

A scoring scheme:

- Match: 2
- Mismatch: -3
- Gap: -2

$$2+2-3-3+2-2+2+2+2 \\ = 4$$

$$2+2-2+2-3-3+2+2+2 \\ = 4$$

$$2+2-2+2-2+2+2-2+2+2 \\ = 8$$

Alignment is gap placement.

How many possible alignment?

# Dynamic programming

Solve a given complex problem by breaking it into subproblems and store the results of subproblems to avoid computing the same results again.

Two key properties of a problem that suggest that the given problem can be solved using DP.

## 1. Overlapping Subproblems

- Given problem can be recursively broken down into subproblems that can be related to each other. This is total no. of subproblems is polynomial.

## 2. Optimal Substructure

- The optimal solution can be produced by combining optimal solutions of subproblems.



Richard Bellman

Optimal decision processes, involved time series & planning - thus 'dynamic' & 'programming'.

"It's impossible to use the word dynamic in a pejorative sense"; DP was "something not even a Congressman could object to."

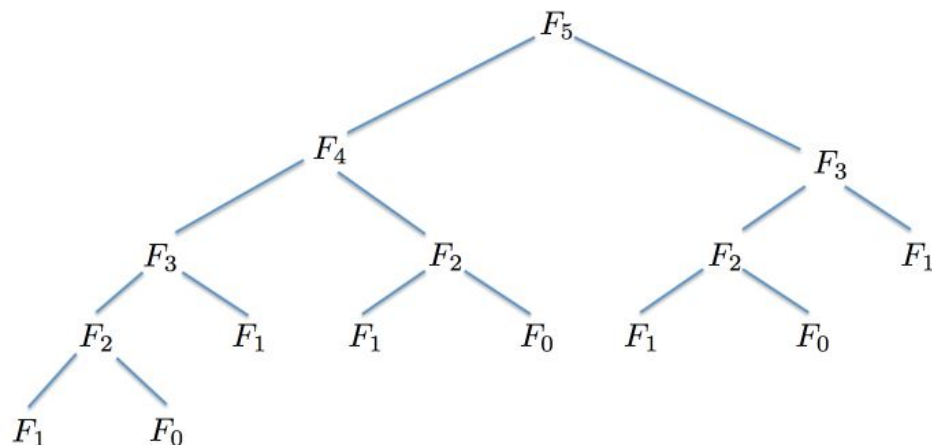
# Dynamic programming

Hemachandra/Fibonacci numbers: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, .....

$$\begin{aligned} F_0 &:= 0; F_1 := 1; \\ F_n &= F_{n-1} + F_{n-2}, \text{ for all } n \geq 2. \end{aligned}$$

A trivial algorithm for computing  $F_n$ :

```
naive_fib(n):  
    if n ≤ 1: return n  
    else: return naive_fib(n - 1) +  
           naive_fib(n - 2)
```





# Dynamic programming

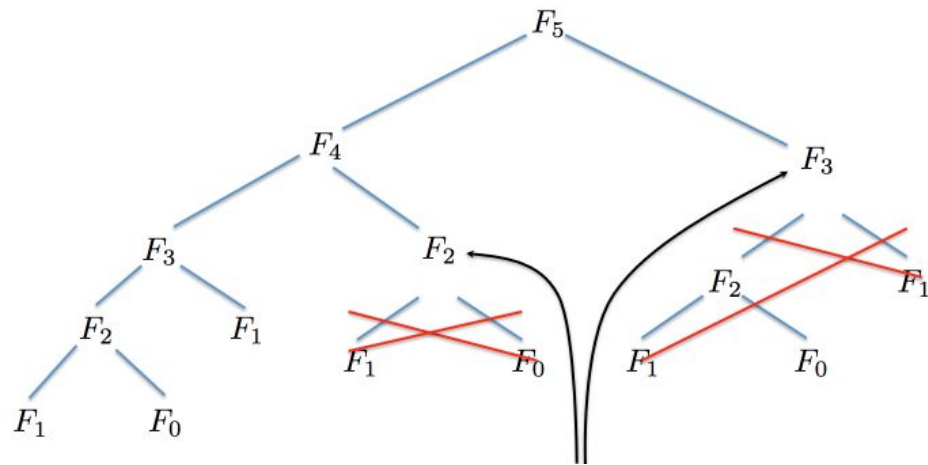
Hemachandra/Fibonacci numbers:  $F_0 := 0$ ;  $F_1 := 1$ ;  $F_n = F_{n-1} + F_{n-2}$ , for all  $n \geq 2$ .

Never recompute a subproblem  $F(k)$ ,  $k \leq n$ , if it has been computed before.

Remembering previously computed values: memoization.

Improved algorithm for computing  $F_n$ :

```
memo = { }  
  
fib(n):  
    if n in memo: return memo[n]  
    else if n = 0: return 0  
    else if n = 1: return 1  
    else: f = fib(n - 1) + fib(n - 2)  
    memo[n] = f  
    return f
```



These values are already computed and stored in memo when runtime processes these nodes of the recursion.

# Dynamic programming

1. Overlapping Subproblems
2. Optimal Substructure

DP  $\approx$  recursion + memoization (reuse)

- Remember (memoize) previously solved “subproblems”; e.g., in Fibonacci, we memoized the solutions to the subproblems  $F_0, F_1, \dots, F_{n-1}$ , while unraveling the recursion.
- If we encounter a subproblem that has already been solved, reuse solution.
- Runtime  $\approx$  (no. of subproblems) \* (time per subproblem)

# Needleman-Wunsch algorithm

1. Scoring function: substitution matrix & gap penalty
2. Matrix initialization & filling
3. Traceback

A scoring scheme:

- Match: 1
- Mismatch: -1
- Gap: -1

Align GCAT with GAT

	-	G	C	A	T
-					
G					
A					
T					

# Needleman-Wunsch algorithm

1. Scoring function: substitution matrix & gap penalty
2. Matrix initialization & filling
3. Traceback

Align GCAT with GAT

$$M(0, j) = j * p$$

$$M(i, 0) = i * p$$

$$M(i, j) = \text{MAX} \left( \begin{array}{l} M(i-1, j) + p, \\ M(i, j-1) + p, \\ M(i-1, j-1) + S(A_i, B_j) \end{array} \right)$$

top  
left  
diagonal

	-	G	C	A	T
-					
G					
A					
T					

# Needleman-Wunsch algorithm

1. Scoring function: substitution matrix & gap penalty
2. Matrix initialization & filling
3. Traceback

Align GCAT with GAT

$$M(0, j) = j * p$$

$$M(i, 0) = i * p$$

$$M(i, j) = \text{MAX} \left( \begin{array}{l} M(i-1, j) + p, \\ M(i, j-1) + p, \\ M(i-1, j-1) + S(A_i, B_j) \end{array} \right)$$

top  
left  
diagonal

	-	G	C	A	T
-	0	-1	-2	-3	-4
G	-1				
A	-2				
T	-3				

# Needleman-Wunsch algorithm

1. Scoring function: substitution matrix & gap penalty
2. Matrix initialization & filling
3. Traceback

$$M(0, j) = j * p$$

$$M(i, 0) = i * p$$

$$M(i, j) = \text{MAX} \left( \begin{array}{l} M(i-1, j) + p, \\ M(i, j-1) + p, \\ M(i-1, j-1) + S(A_i, B_j) \end{array} \right)$$

top

left

diagonal

	-	G	C	A	T
-	0	-1	-2	-3	-4
G	-1	?			
A	-2				
T	-3				

	-	G	C	A	T
-	0	-1	-2	-3	-4
G	-1	-2			
A	-2				
T	-3				

	-	G	C	A	T
-	0	-1	-2	-3	-4
G	-1	-2			
A	-2				
T	-3				

	-	G	C	A	T
-	0	-1	-2	-3	-4
G	-1	1			
A	-2				
T	-3				

# Needleman-Wunsch algorithm

1. Scoring function: substitution matrix & gap penalty
2. Matrix initialization & filling
3. Traceback

Align GCAT with GAT

$$M(0, j) = j * p$$

$$M(i, 0) = i * p$$

$$M(i, j) = \text{MAX} \left( \begin{array}{l} M(i-1, j) + p, \\ M(i, j-1) + p, \\ M(i-1, j-1) + S(A_i, B_j) \end{array} \right)$$

top  
left  
diagonal

	-	G	C	A	T
-	0	-1	-2	-3	-4
G	-1	1	0	-1	-2
A	-2	0	0	1	0
T	-3	-1	-1	0	2

# Needleman-Wunsch algorithm

1. Scoring function: substitution matrix & gap penalty
2. Matrix initialization & filling
3. Traceback

$$M(0, j) = j * p$$

$$M(i, 0) = i * p$$

$$M(i, j) = \text{MAX} \left( \begin{array}{l} M(i-1, j) + p, \\ M(i, j-1) + p, \\ M(i-1, j-1) + S(A_i, B_j) \end{array} \right)$$

top

left

diagonal

Align GCAT with GAT

GCAT  
G-AT

	-	G	C	A	T
-	0	-1	-2	-3	-4
G	-1	1	0	-1	-2
A	-2	0	0	1	0
T	-3	-1	-1	0	2



# Needleman-Wunsch algorithm

1. Scoring function: substitution matrix & gap penalty
2. Matrix initialization & filling
3. Traceback

Align **ATGCT** with **ATTACA**

$$M(0, j) = j * p$$

$$M(i, 0) = i * p$$

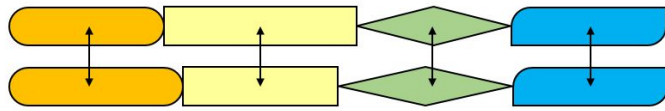
$$M(i, j) = \text{MAX} \left( \begin{array}{l} M(i-1, j) + p, \\ M(i, j-1) + p, \\ M(i-1, j-1) + S(A_i, B_j) \end{array} \right)$$

top  
left  
diagonal

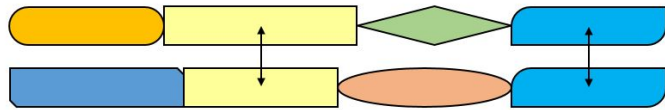
	-	A	T	T	A	C	A
-							
A							
T							
G							
C							
T							

# Local alignment

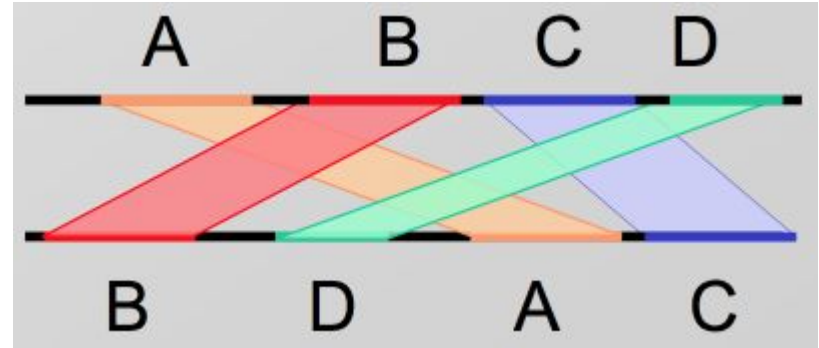
A local alignment of strings  $s$  and  $t$  is an alignment of a substring of  $s$  with a substring of  $t$ .



Global Alignment



Local Alignment



# Smith-Waterman algorithm

Similar to Needleman-Wunsch, with 3 changes:

- First row/column set to 0.
- No negative scores, set to 0.
- Backtrack from cell with highest score, stop at 0.

Align GCAT with GCT

$$M(0, j) = 0$$

$$M(i, 0) = 0$$

$$M(i, j) = \text{MAX}(\begin{array}{l} 0, \\ M(i-1, j) + p, \\ M(i, j-1) + p, \\ M(i-1, j-1) + S(A_i, B_j) \end{array})$$

top  
left  
diagonal

# Smith-Waterman algorithm

Similar to Needleman-Wunsch, with 3 changes:

- First row/column set to 0.
- No negative scores, set to 0.
- Backtrack from cell with highest score, stop at 0.

$$M(0, j) = 0$$

$$M(i, 0) = 0$$

$$M(i, j) = \text{MAX} \left( \begin{array}{l} 0, \\ M(i-1, j) + p, \\ M(i, j-1) + p, \\ M(i-1, j-1) + S(A_i, B_j) \end{array} \right)$$

top

left

diagonal

Align GCAT with GAT

GC  
GC

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

# How do we scale this up to search an entire sequence database?

Given a query sequence, and a large set of target sequences (millions), which target sequences (if any) are related to the query?

- Individual alignments need not be perfect: Once initial matches are found, they can fine-tune them later.
- Must be very fast.

Exploit the nature of the problem (most sequences will be unrelated to the query):

- If any match with % identity  $\leq 90$  is going to be rejected, can ignore sequences which don't have a stretch of 10 nucleotides in a row.
- Pre-screen sequences for common long stretches.
- Pre-process the database offline and index k-mers.

TITLE	CITED BY	YEAR
<b>Basic local alignment search tool</b> SF Altschul, W Gish, W Miller, EW Myers, DJ Lipman Journal of molecular biology 215 (3), 403-410	69248	1990
<b>Gapped BLAST and PSI-BLAST: a new generation of protein database search programs</b> SF Altschul, TL Madden, AA Schäffer, J Zhang, Z Zhang, W Miller, ... Nucleic acids research 25 (17), 3389-3402	64876	1997

U.S. National Library of Medicine

National Center for Biotechnology Information

Sign in to NCBI

**BLAST®**

Home
 Recent Results
 Saved Strategies
 Help

### Basic Local Alignment Search Tool

BLAST finds regions of similarity between biological sequences. The program compares nucleotide or protein sequences to sequence databases and calculates the statistical significance.

[Learn more](#)

NEWS

**IgBLAST 1.8.0 released**  
 A new version of IgBLAST is now available.  
 Wed, 15 Nov 2017 16:00:00 EST

More BLAST news...

### Web BLAST

**Nucleotide BLAST**  
 nucleotide ► nucleotide

**blastx**  
 translated nucleotide ► protein

**tblastn**  
 protein ► translated nucleotide

**Protein BLAST**  
 protein ► protein

### BLAST Genomes

Human
 Mouse
 Rat
 Microbes

<https://www.ncbi.nlm.nih.gov/BLAST/>

# BLAST

Query sequence: PQGEFG

Word 1: PQG

Word 2: QGE

Word 3: GEF

Word 4: EFG

query word ( $W = 3$ )

Query: GSVEDTTGSQSLAALLNKCKT**PQG**QRLVNQWIKQPLMDKNRIEERLNLVEAFVEDAELRQTLQEDL

neighborhood  
words

PQG	18
PEG	15
PRG	14
PKG	14
PNG	13
PDG	13
PHG	13
<b>PMG</b>	13
PSG	13
PQA	12
PQN	12
etc...	

neighborhood  
score threshold  
( $T = 13$ )

Query: 325 SLAALLNKCKT**PQG**QRLVNQWIKQPLMDKNRIEERLNLVEA 365  
+LA++L+ TP G R++ +U+ P+ D + ER + A  
Sbjct: 290 TLASVLDCTV**PMG**SRMLKRWLHMPVRDTRVLLERQQTIGA 330

High-scoring Segment Pair (HSP)