Upcoming project deadline

- Project topic due on **Feb 01**.
    - Discuss with me (*and* any other PI) and read recent papers.
    - Briefly describe a project idea:
        - Title
        - Project advisor (if someone outside class)
        - 250-word abstract addressing the following 4 Qs:
            - What is the problem?
            - How is it addressed currently & what are the limitations?
            - What is your approach to addressing it & why is likely to be successful?
            - If successful, why does it matter (what is the impact)?

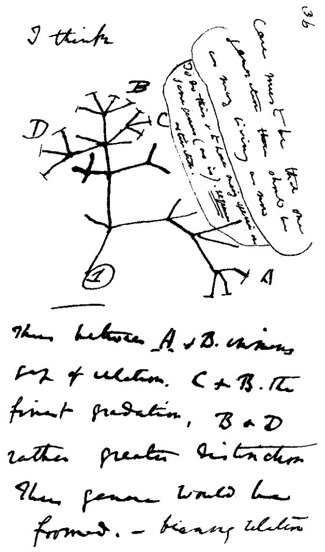
Couple more things

- You are exempt from assignment if you are scribing or presenting.
- Scribing:
  - Make use of the text in the slides (note the slide number in parentheses).
  - Incorporate the figures/images in the slides (note the slide number in parentheses)
    - If resolution is not good, go to original source.
  - Incorporate term definitions, explanations, and images from outside resources (with citation).
  - Notate math using Katex.

Lecture 6: Sequence alignment

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

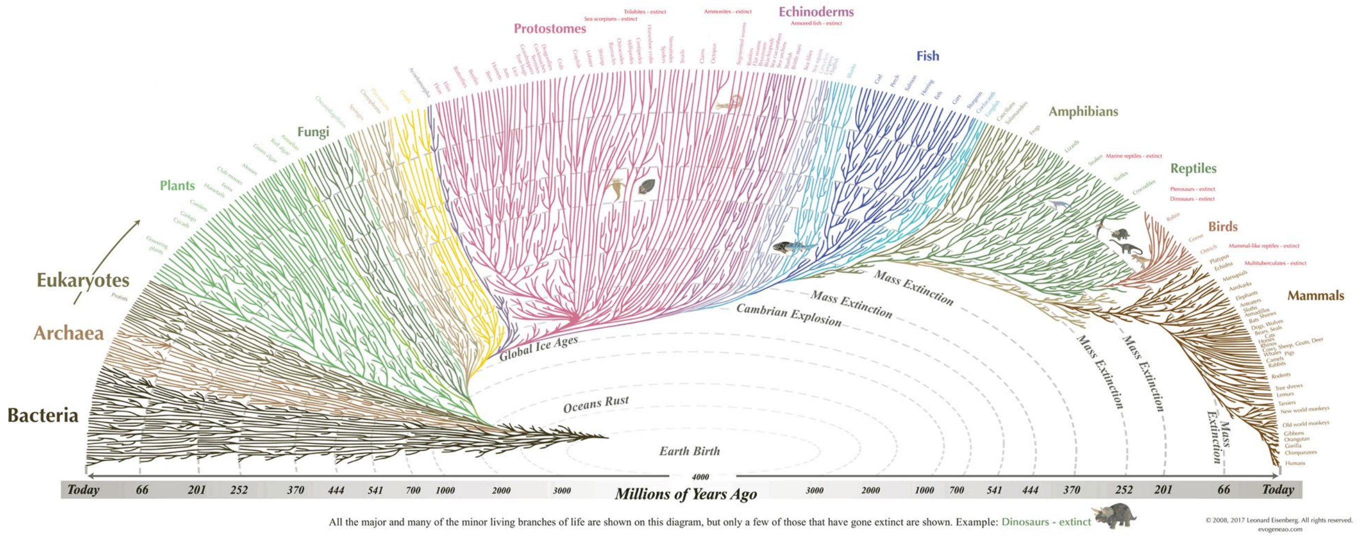
Sequence evolution


36

I think

Case must be that one generation then should be as many living as now. To do this & to have many species in same genus (as is) requires extinction.

Thus between A & B immense gap of relation. C & B the finest gradation, B & D rather greater distinction. Thus genera would be formed. — bearing relation

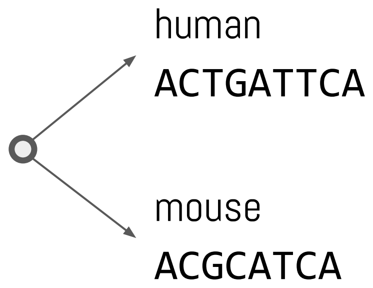


Bacteria
Archaea
Eukaryotes
Plants
Fungi
Protostomes
Echinoderms
Fish
Amphibians
Reptiles
Birds
Mammals
Global Ice Ages
Oceans Rust
Earth Birth
Cambrian Explosion
Mass Extinction
Today
66
201
252
370
444
541
700
1000
2000
3000
4000
Millions of Years Ago
All the major and many of the minor living branches 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. evogeneao.com

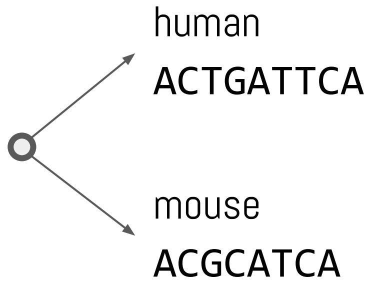
deletion — mutation — insertion

ACATGGTCA → AC*****TGGTCA → ACTG**A**TCA → ACTGAT**T**CA

Evolutionary time



Sequence evolution

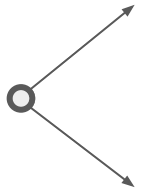


Sequences can be aligned by allowing for **gaps** and **mismatches**.

```
ACTGATTCA          ACTGATTCA          ACTG-ATTCA
ACGCA-TCA          AC-GCATCA          AC-GCAT-CA
```

Which alignment is correct?

Sequence alignment



human

ACTGATTCA

mouse

ACGCATCA

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

We will come back to this!

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 alignments?

Dynamic programming

Solve a given complex problem by:

1. Breaking it into **<u>subproblems</u>** and
2. Storing 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

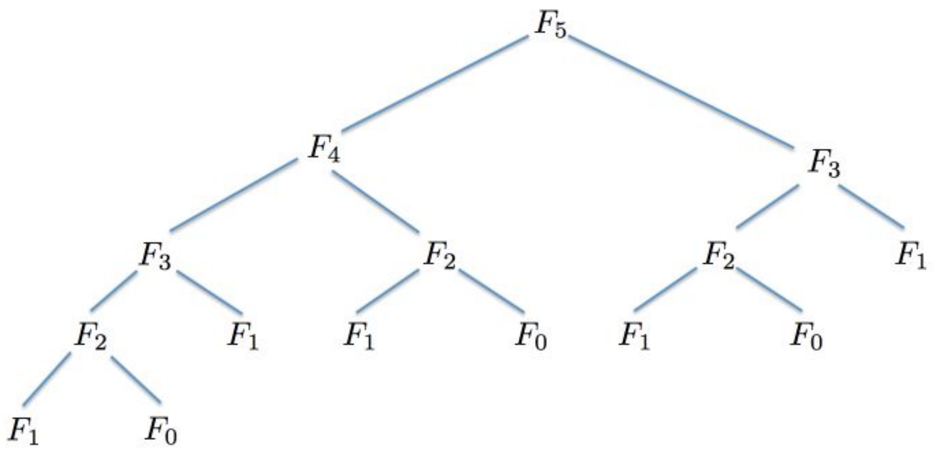
Hemachandra/Fibonacci numbers: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ……

$$F_0 := 0;\ F_1 := 1;$$

$$F_n = F_{n-1} + F_{n-2}, \text{ for all } n \geq 2.$$

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)
```



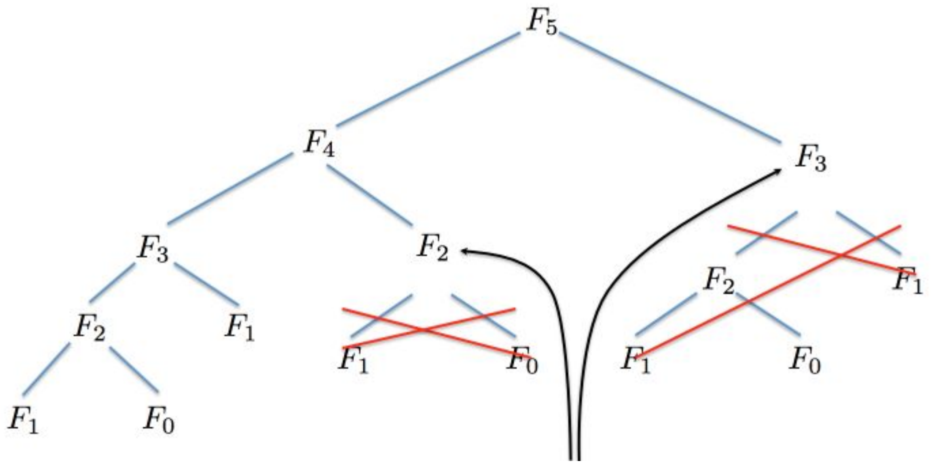
Dynamic programming

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.
Memoization: Remembering previously computed values.

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 ≈ recursion + memoization (reuse)

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

Needleman-Wunsch algorithm

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

Align **GCAT** with **GAT**

Step 1

A scoring scheme:
- Match: 1
- Mismatch: -2
- Gap: -1

	—	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**

Step 2

```
M(0, j) = j*p
M(i, 0) = i*p

M(i, j) = MAX(   M(i-1, j) + p,            top
                 M(i, j-1) + p,            left
         M(i-1, j-1) + S(A_i, B_j)   )     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**

<u>Step 2</u>

```
M(0, j) = j*p
M(i, 0) = i*p

M(i, j) = MAX(   M(i-1, j) + p,          top
                 M(i, j-1) + p,          left
         M(i-1, j-1) + S(A_i, B_j)   )   diagonal
```

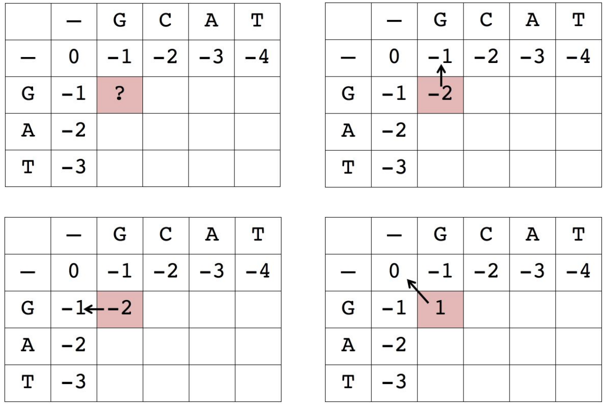
	–	G	C	A	T
–	0	–1	–2	–3	–4
G	–1				
A	–2				
T	–3				

Needleman-Wunsch algorithm

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

<u>Step 2</u>

```
M(0, j) = j*p
M(i, 0) = i*p


M(i, j) = MAX(   M(i-1, j) + p,              top
                 M(i, j-1) + p,              left
             M(i-1, j-1) + S(A_i, B_j)   )   diagonal
```

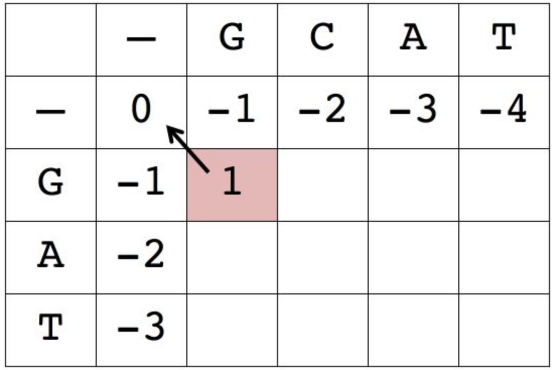


	–	G	C	A	T
–	0	-1	-2	-3	-4
G	-1	?			
A	-2				
T	-3				

	–	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				



	–	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

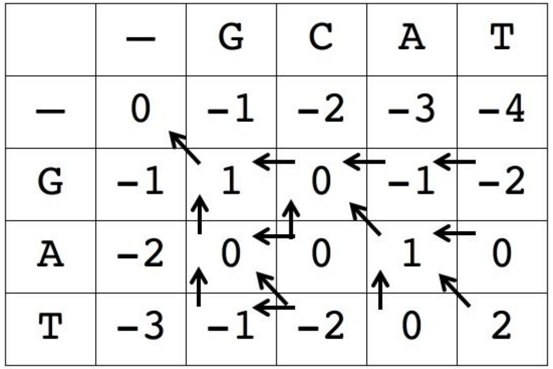
Align **GCAT** with **GAT**

<u>Step 2</u>

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

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

$$M(i, j) = \max\begin{cases} M(i-1, j) + p, & \text{top} \\ M(i, j-1) + p, & \text{left} \\ M(i-1, j-1) + S(A_i, B_j) & \text{diagonal} \end{cases}$$



	—	G	C	A	T
—	0	−1	−2	−3	−4
G	−1	1	0	−1	−2
A	−2	0	0	1	0
T	−3	−1	−2	0	2

Needleman-Wunsch algorithm

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

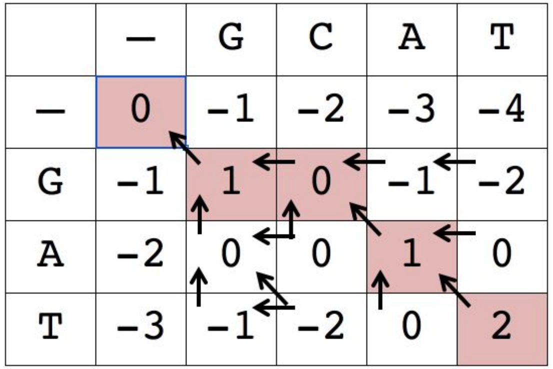
Align **GCAT** with **GAT**

```
GCAT
G-AT
```

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

$$\begin{array}{llll} M(i, j) = \mathrm{MAX}(& M(i-1, j) + p, & & \text{top} \\ & M(i, j-1) + p, & & \text{left} \\ & M(i-1, j-1) + S(A_i, B_j) &) & \text{diagonal} \end{array}$$

Step 3



	—	G	C	A	T
—	0	−1	−2	−3	−4
G	−1	1	0	−1	−2
A	−2	0	0	1	0
T	−3	−1	−2	0	2

Needleman-Wunsch algorithm

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

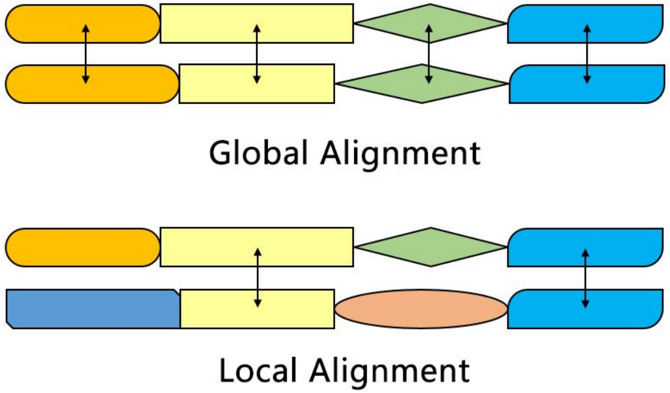
Align **ATGCT** with **ATTACA**

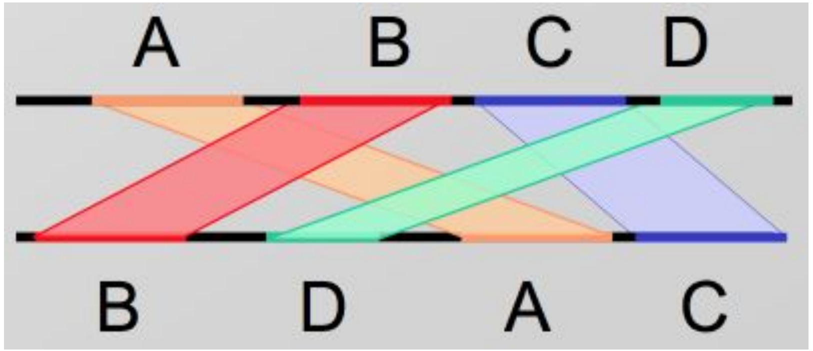
```
M(0, j) = j*p
M(i, 0) = i*p


M(i, j) = MAX(   M(i-1, j) + p,              top
                 M(i, j-1) + p,              left
         M(i-1, j-1) + S(A_i, B_j)    )      diagonal
```

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

Global & local alignment

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





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) = MAX(                   0,
                    M(i-1, j) + p,        top
                    M(i, j-1) + p,        left
          M(i-1, j-1) + S(A_i, B_j)   )   diagonal
```

	-	G	C	A	T
-					
G					
C					
T					

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.

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

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

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

Align **GCAT** with **GCT**

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