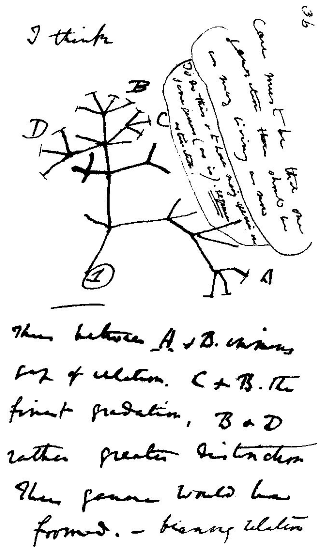
Week 2: Sequence alignment & search

- Sequence alignment problem
- Dynamic programming
- Global alignment
  - Needleman-Wunsch algorithm
- Local alignment
  - Smith-Waterman algorithm
- Substitution matrix
  - Construction & properties
- Fast sequence searches
  - BLAST; Statistics of similarity search

Sequence evolution

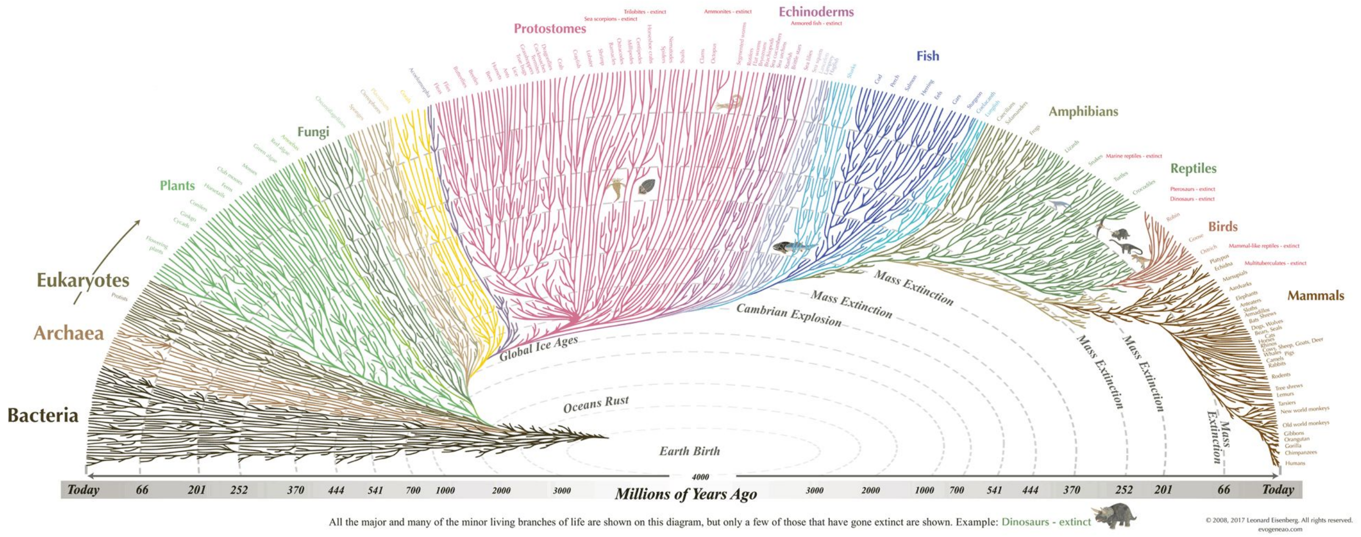


36

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

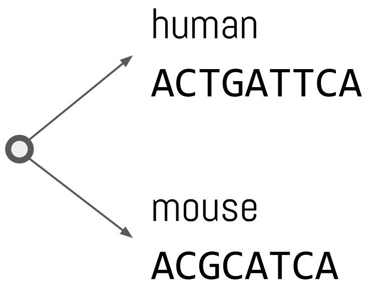


Bacteria
Archaea
Eukaryotes
Plants
Fungi
Protostomes
Echinoderms
Fish
Amphibians
Reptiles
Birds
Mammals
Global Ice Ages
Oceans Rust
Earth Birth
Cambrian Explosion
Mass Extinction
Today
66
201
252
370
444
541
700
1000
2000
3000
4000
Millions of Years Ago
All the major and many of the minor living branches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

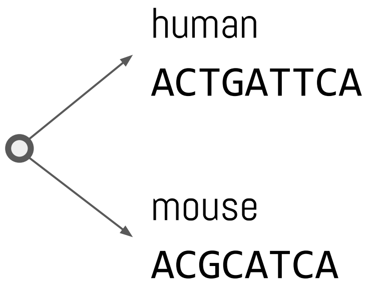
deletion — mutation — insertion

ACATGGTCA → AC*TGGTCA → ACTG<u>A</u>TCA → ACTGAT<u>T</u>CA

→ Evolutionary time



What is sequence alignment?



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

Alignment 1	Alignment 2	Alignment 3
`ACTGATTCA`	`ACTGATTCA`	`ACTG-ATTCA`
`ACGCA-TCA`	`AC-GCATCA`	`AC-GCAT-CA`

Which alignment is correct?

Alignment is gap placement.

Dynamic programming

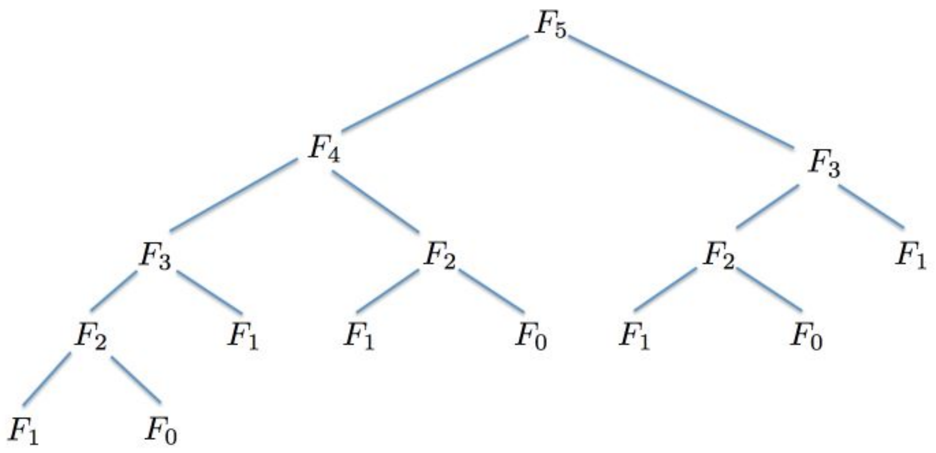
Hemachandra/Fibonacci numbers: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ……

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

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

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)
```



Needleman-Wunsch algorithm

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

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

Step 2

p: gap penalty

$s(S1_i, S2_j)$: match/mismatch score

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

$$M(i, j) = \text{MAX}(\ M(i-1, j) + p,$$ top

$$M(i, j-1) + p,$$ left

$$M(i-1, j-1) + s(S1_i, S2_j))$$ diagonal

	—	G	C	A	T
—					
G					
A					
T					

Substitution matrix to measure similarity in sequence alignments

Substitution matrix: A collection of scores for aligning nucleotides or amino acids with one another.

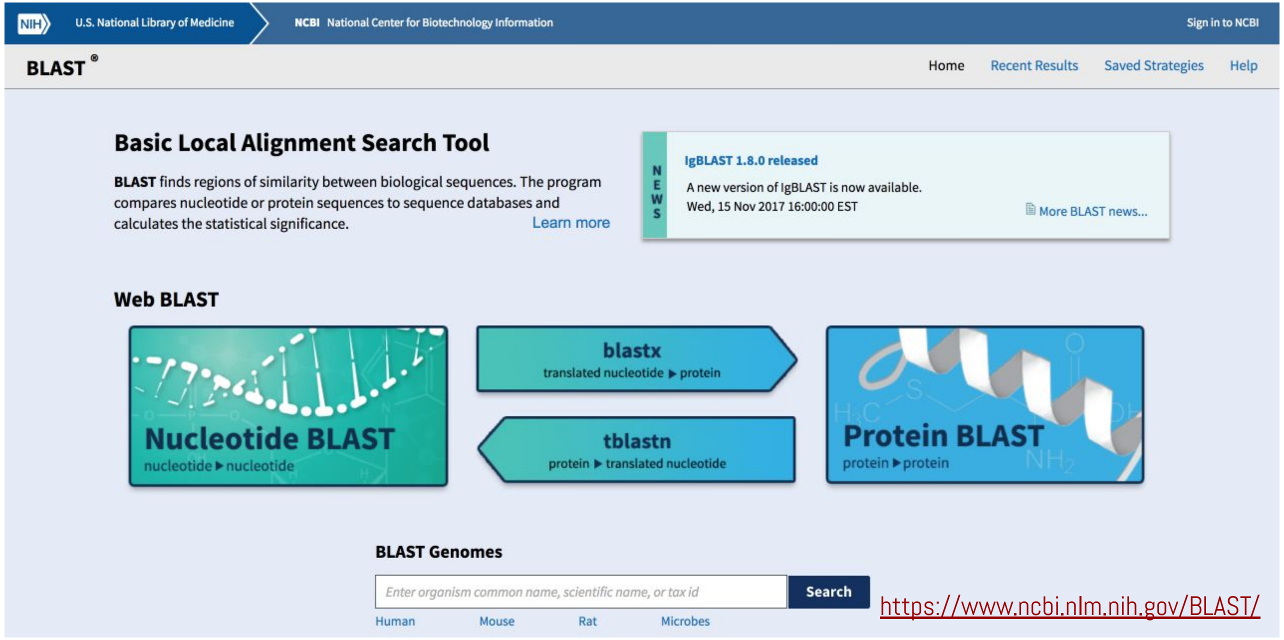
- Each score: the relative ease with which one nuc or AA may mutate into or substitute for another.
- Purely statistical, nothing directly to do with structure/biochemistry.

Ala	4																			
Arg	−1	5																		
Asn	−2	0	6																	
Asp	−2	−2	1	6																
Cys	0	−3	−3	−3	9															
Gln	−1	1	0	0	−3	5														
Glu	−1	0	0	2	−4	2	5													
Gly	0	−2	0	−1	−3	−2	−2	6												
His	−2	0	1	−1	−3	0	0	−2	8											
Ile	−1	−3	−3	−3	−1	−3	−3	−4	−3	4										
Leu	−1	−2	−3	−4	−1	−2	−3	−4	−3	2	4									
Lys	−1	2	0	−1	−3	1	1	−2	−1	−3	−2	5								
Met	−1	−1	−2	−3	−1	0	−2	−3	−2	1	2	−1	5							
Phe	−2	−3	−3	−3	−2	−3	−3	−3	−1	0	0	−3	0	6						
Pro	−1	−2	−2	−1	−3	−1	−1	−2	−2	−3	−3	−1	−2	−4	7					
Ser	1	−1	1	0	−1	0	0	0	−1	−2	−2	0	−1	−2	−1	4				
Thr	0	−1	0	−1	−1	−1	−1	−2	−2	−1	−1	−1	−1	−2	−1	1	5			
Trp	−3	−3	−4	−4	−2	−2	−3	−2	−2	−3	−2	−3	−1	1	−4	−3	−2	11		
Tyr	−2	−2	−2	−3	−2	−1	−2	−3	2	−1	−1	−2	−1	3	−3	−2	−2	2	7	
Val	0	−3	−3	−3	−1	−2	−2	−3	−3	3	1	−2	1	−1	−2	−2	0	−3	−1	4
	Ala	**Arg**	**Asn**	**Asp**	**Cys**	**Gln**	**Glu**	**Gly**	**His**	**Ile**	**Leu**	**Lys**	**Met**	**Phe**	**Pro**	**Ser**	**Thr**	**Trp**	**Tyr**	**Val**

Wikipedia; Eddy (2004)

BLAST

TITLE	CITED BY	YEAR
Basic local alignment search tool SF Altschul, W Gish, W Miller, EW Myers, DJ Lipman Journal of molecular biology 215 (3), 403-410	136003 *	1990



NIH
U.S. National Library of Medicine
NCBI 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
Enter organism common name, scientific name, or tax id
Search
Human
Mouse
Rat
Microbes

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

What to brush-up on?

Biology

1. What is DNA? What does a DNA sequence look like? What do A, T, G, and C mean?
2. What is a protein? What does a protein sequence look like? What do the individual characters in the sequence mean?

Algorithms & coding

1. What is an algorithm?
2. What is a pseudocode of an algorithm?
3. What is recursion and what are loops (for, while)?
4. What is a conditional statement (if, else) and how is it used in coding?

What to brush-up on?

Analytical concepts & techniques

1. What is a matrix?
2. How do you write a mathematical expression to refer to a particular cell in the matrix based on its row and column?

Probability & statistics

1. What does probability mean?
2. How do you write a mathematical expression for the probability that: i) event A occurs, and ii) two events A and B occur together?
3. What is a probability distribution? What do the parameters in a probability distribution mean?
4. What is the difference between a discrete and a continuous probability distribution?
5. How do you write a mathematical expression for the probability that a particular variable x is less than or equal to a particular value S?
6. What is the binomial distribution? What kinds of processes doe this distribution capture well?
7. What is the exponential distribution? What kinds of processes doe this distribution capture well?

Week 2: Sequence alignment & search

- Sequence alignment problem
- Dynamic programming
- Global alignment
  - Needleman-Wunsch algorithm
- Local alignment
  - Smith-Waterman algorithm
- Substitution matrix
  - Construction & properties
- Fast sequence searches
  - BLAST; Statistics of similarity search

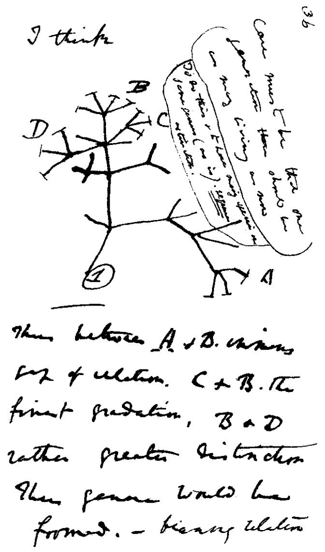
Week 2: Sequence alignment & search

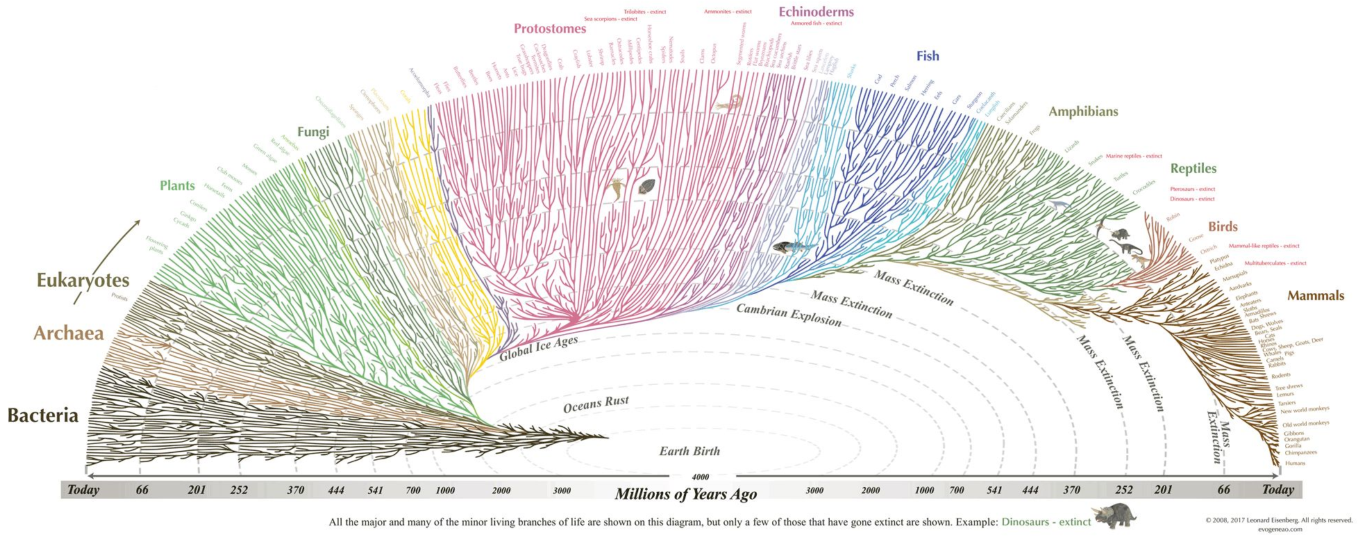
Alignment

- Sequence alignment problem
- Dynamic programming
- Global alignment
  - Needleman-Wunsch algorithm
- Local alignment
  - Smith-Waterman algorithm

- Sign into Pear Deck using the link on Slack
  - Keep a paper and a pen(cil) ready

Sequence evolution

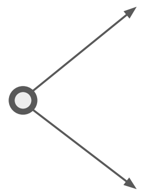

I think
B
D
C
1
A
36
Thus between A & B immense gap of relation. C & B the finest gradation, B & D rather greater distinction. Thus genera would be formed. — bearing relation


Protostomes
Echinoderms
Fish
Amphibians
Reptiles
Birds
Mammals
Fungi
Plants
Eukaryotes
Archaea
Bacteria
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 3000 2000 1000 700 541 444 370 252 201 66 Today
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

$$\text{ACATGGTCA} \rightarrow \text{AC*TGGTCA} \rightarrow \text{ACTG}\underline{\text{A}}\text{TCA} \rightarrow \text{ACTGAT}\underline{\text{T}}\text{CA}$$

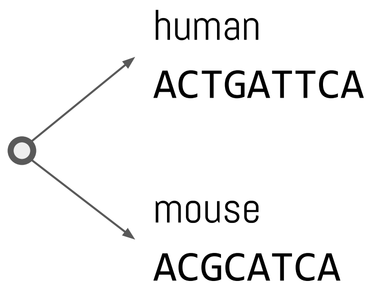
Evolutionary time



human
ACTGATTCA

mouse
ACGCATCA

What is sequence alignment?



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

Alignment 1	Alignment 2	Alignment 3
`ACTGATTCA`	`ACTGATTCA`	`ACTG-ATTCA`
`ACGCA-TCA`	`AC-GCATCA`	`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	2+2−2+2−3−3+2+2+2	2+2−2+2−2+2+2−2+2+2
= **4**	= **4**	= **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. That 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

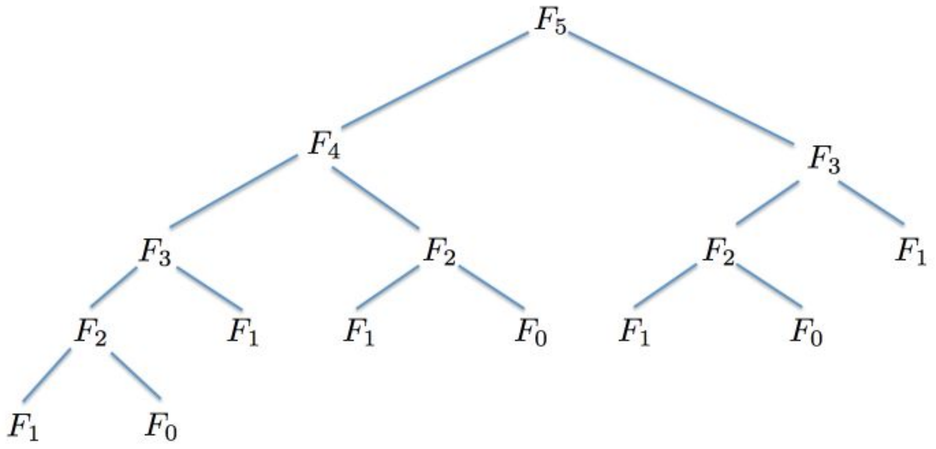
Hemachandra/Fibonacci numbers: `0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ……`

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

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

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)
```



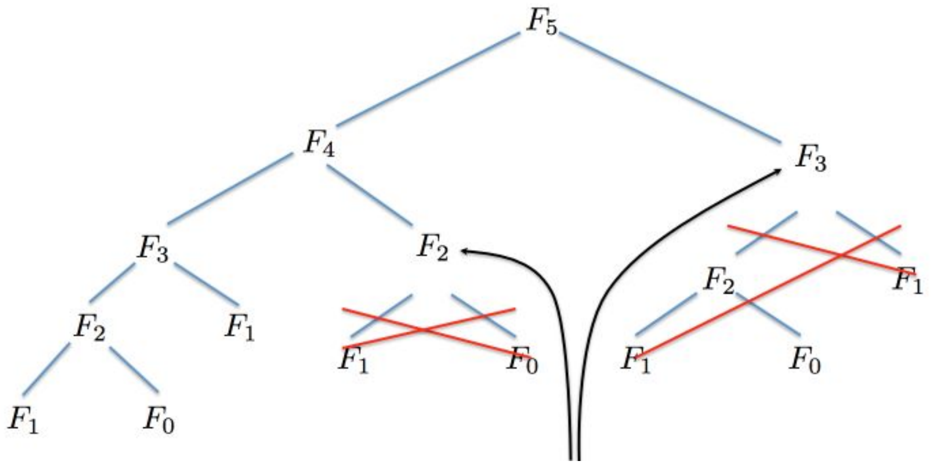
Dynamic programming

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 $F_0, F_1, \cdots, F_{n-1}$, 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

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

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

<u>**Step 2**</u>

```
p: gap penalty
s(S1_i, S2_j): match/mismatch score
```

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

$$M(i, j) = \mathrm{MAX}(\ M(i-1, j) + p, \quad \text{top}$$
$$M(i, j-1) + p, \quad \text{left}$$
$$M(i-1, j-1) + s(S1_i, S2_j)) \quad \text{diagonal}$$

	–	G	C	A	T
–					
G					
A					
T					

Needleman-Wunsch algorithm

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

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

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

Step 2

```
p: gap penalty
s(S1_i, S2_j): match/mismatch score
```

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

$$M(i, j) = \mathrm{MAX}(\ M(i-1, j) + p,\quad \text{top}$$
$$M(i, j-1) + p,\quad \text{left}$$
$$M(i-1, j-1) + s(S1_i, S2_j))\quad \text{diagonal}$$

	–	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

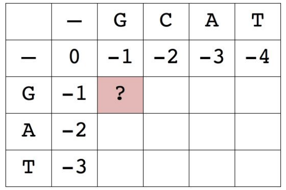
A scoring scheme:
- Match: **1**
- Mismatch: **−2**
- Gap: **−1**

Step 2

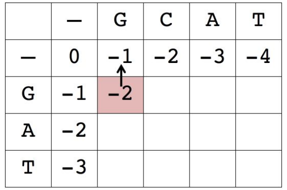
p: gap penalty
$s(S1_i, S2_j)$: match/mismatch score

$$M(0, j) = j*p; \quad M(i, 0) = i*p$$

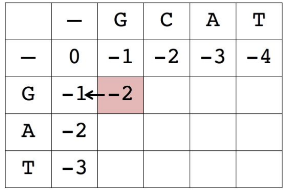
$$\begin{aligned} M(i, j) = \text{MAX}(\quad & M(i-1, j) + p, && \text{top} \\ & M(i, j-1) + p, && \text{left} \\ & M(i-1, j-1) + s(S1_i, S2_j)) && \text{diagonal} \end{aligned}$$



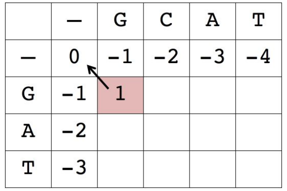
	–	G	C	A	T
–	0	−1	−2	−3	−4
G	−1	?			
A	−2				
T	−3				



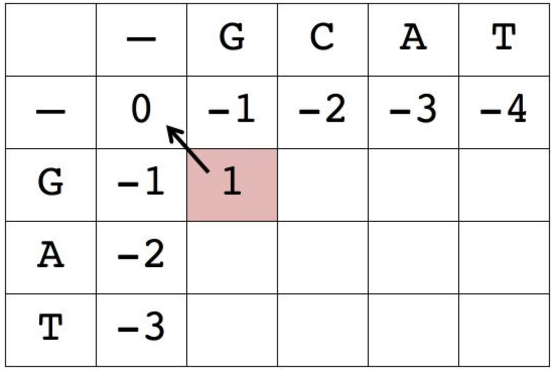
	–	G	C	A	T
–	0	−1	−2	−3	−4
G	−1	−2			
A	−2				
T	−3				



	–	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

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

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

Fill the remaining cells in this matrix.

Step 2

p: gap penalty
$s(S1_i, S2_j)$: match/mismatch score

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

$$M(i, j) = \mathrm{MAX}(\; M(i-1, j) + p, \quad \text{top}$$
$$M(i, j-1) + p, \quad \text{left}$$
$$M(i-1, j-1) + s(S1_i, S2_j)) \quad \text{diagonal}$$

	–	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

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

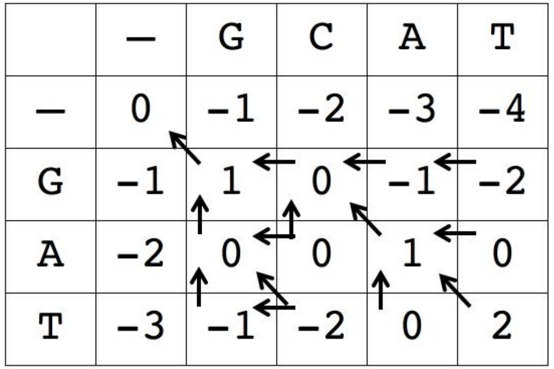
Align **GCAT** with **GAT**

Step 2

p: gap penalty
$s(S1_i, S2_j)$: match/mismatch score

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

$$M(i, j) = \text{MAX}(\; M(i-1, j) + p, \quad \text{top}$$
$$M(i, j-1) + p, \quad \text{left}$$
$$M(i-1, j-1) + s(S1_i, S2_j)) \quad \text{diagonal}$$



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

Needleman-Wunsch algorithm

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

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

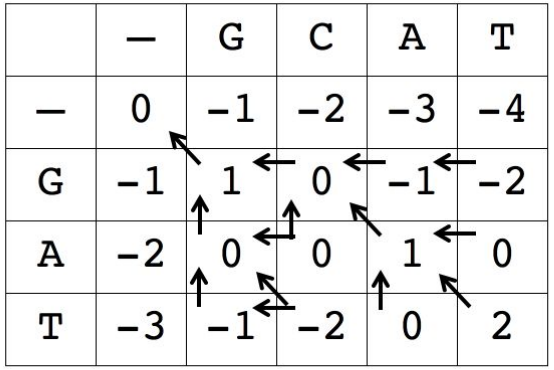
Align **GCAT** with **GAT**

```
p: gap penalty
s($S1_i$, $S2_j$): match/mismatch score

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($S1_i$, $S2_j$))    diagonal
```

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

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

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

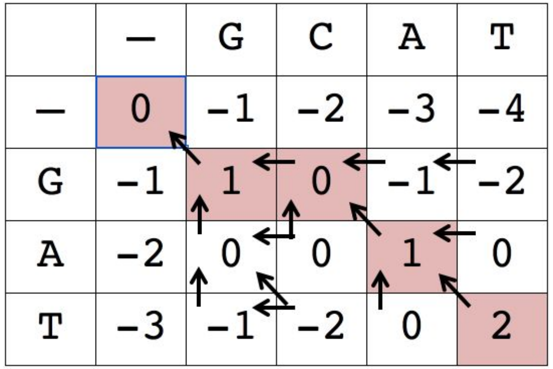
What is the alignment?

p: gap penalty
$s(S1_i, S2_j)$: match/mismatch score

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

$$M(i, j) = \mathrm{MAX}(\ M(i-1, j) + p, \quad \text{top}$$
$$M(i, j-1) + p, \quad \text{left}$$
$$M(i-1, j-1) + s(S1_i, S2_j)) \quad \text{diagonal}$$

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

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

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

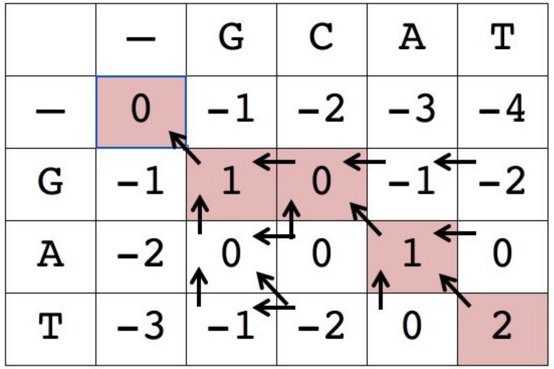
```
GCAT
G-AT
```

p: gap penalty
$s(S1_i, S2_j)$: match/mismatch score

$$M(0, j) = j*p; \quad M(i, 0) = i*p$$

$$M(i, j) = \text{MAX}(\quad M(i-1, j) + p, \qquad \text{top}$$
$$M(i, j-1) + p, \qquad \text{left}$$
$$M(i-1, j-1) + s(S1_i, S2_j)) \qquad \text{diagonal}$$

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

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

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

p: gap penalty

$s(S1_i, S2_j)$: match/mismatch score

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

$M(i, j) = \mathrm{MAX}(\ M(i-1, j) + p,$ top

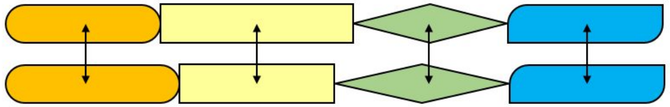
$M(i, j-1) + p,$ left

$M(i-1, j-1) + s(S1_i, S2_j))$ diagonal

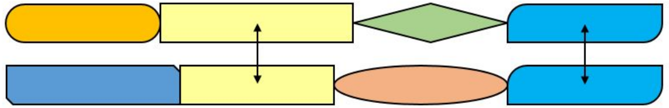
	–	A	T	T	A	C	A
–							
A							
T							
G							
C							
T							

Global & local alignment

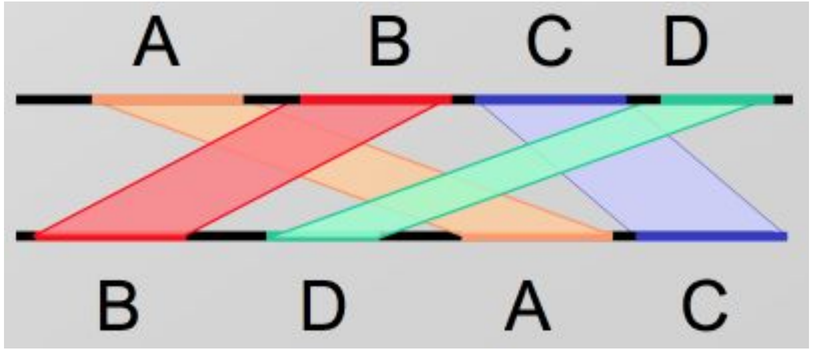
A local alignment of strings *s* and *t* is an alignment of a substring of *s* with a substring of *t*.



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. (Don't record direction.)
- Backtrack from cell with highest score & stop at 0.

```
p: gap penalty
s(S1_i, S2_j): match/mismatch score
```

$$M(0, j) = 0;\ M(i, 0) = 0$$

$$M(i, j) = \mathrm{MAX}\left(\begin{array}{ll} 0, & \\ M(i-1, j) + p, & \text{top} \\ M(i, j-1) + p, & \text{left} \\ M(i-1, j-1) + s(S1_i, S2_j)) & \text{diagonal} \end{array}\right.$$

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

What are the values in the first row and first column?

	-	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. (Don't record direction.)
- Backtrack from cell with highest score & stop at 0.

```
p: gap penalty
s(S1_i, S2_j): match/mismatch score
```

$M(0, j) = 0;\ M(i, 0) = 0$

$$M(i, j) = \mathrm{MAX}\begin{pmatrix} 0, & \\ M(i-1, j) + p, & \text{top} \\ M(i, j-1) + p, & \text{left} \\ M(i-1, j-1) + s(S1_i, S2_j) & \text{diagonal} \end{pmatrix}$$

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

Fill this matrix and enter the highest value.

	–	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. (Don't record direction.)
- Backtrack from cell with highest score & stop at 0.

p: gap penalty

$s(S1_i, S2_j)$: match/mismatch score

$$M(0, j) = 0; \; M(i, 0) = 0$$

$$M(i, j) = \mathrm{MAX}\begin{pmatrix} 0, & \\ M(i-1, j) + p, & \text{top} \\ M(i, j-1) + p, & \text{left} \\ M(i-1, j-1) + s(S1_i, S2_j) & \text{diagonal} \end{pmatrix}$$

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