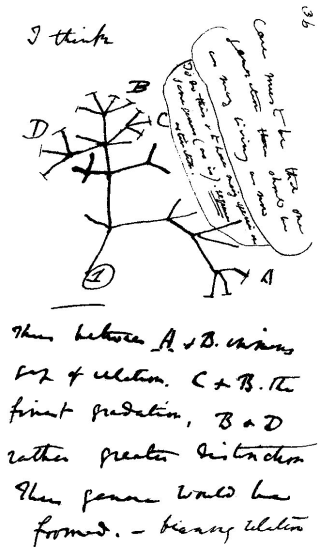
Week 2: Sequence alignment & search

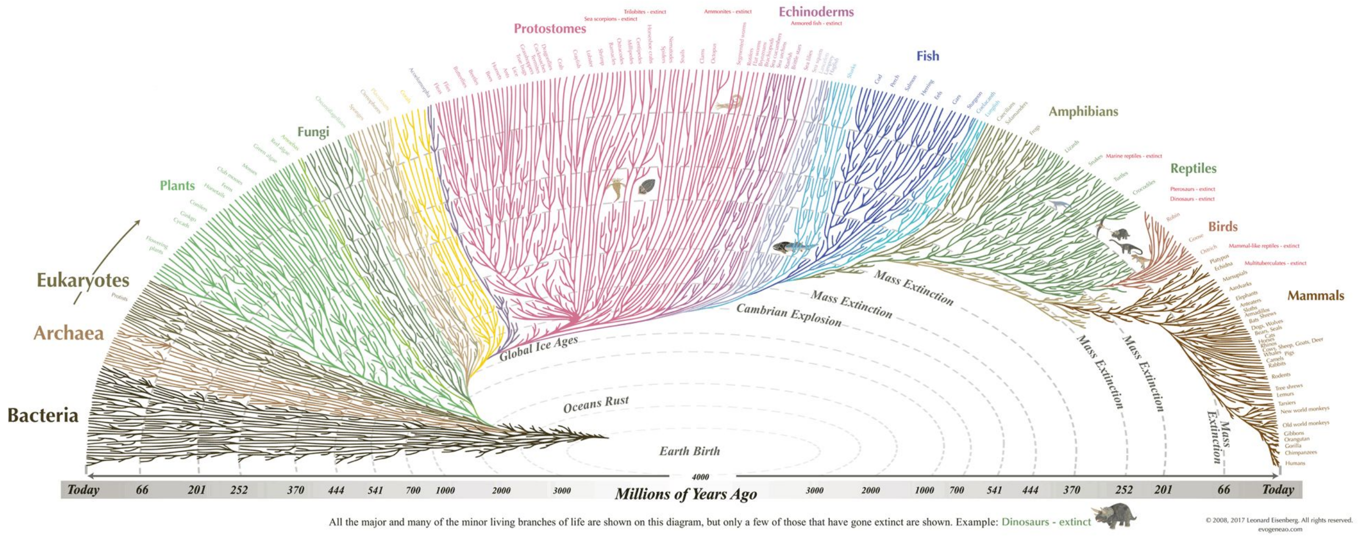
- Sequence alignment problem
- Dynamic programming
- Global alignment
  - Needleman-Wunsch algorithm
- Local alignment
  - Smith-Waterman algorithm
- Substitution matrix
  - Construction & properties
- Fast sequence searches
  - BLAST; Statistics of similarity search

Sequence evolution


36
I think
B
C
D
1
A

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


Protostomes
Echinoderms
Fish
Amphibians
Reptiles
Birds
Mammals
Fungi
Plants
Eukaryotes
Archaea
Bacteria
Global Ice Ages
Oceans Rust
Earth Birth
Mass Extinction
Cambrian Explosion
Today
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

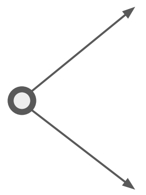
ACATGGTCA → AC*TGGTCA → ACTGATCA → ACTGATTCA

Evolutionary time →

human
ACTGATTCA

mouse
ACGCATCA

What is sequence alignment?



human

ACTGATTCA

mouse

ACGCATCA

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.

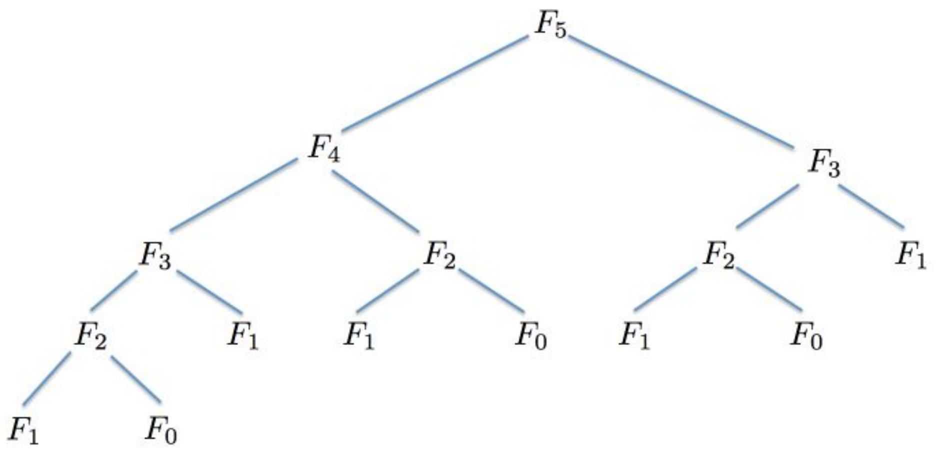
Dynamic programming

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

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

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

	—	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.

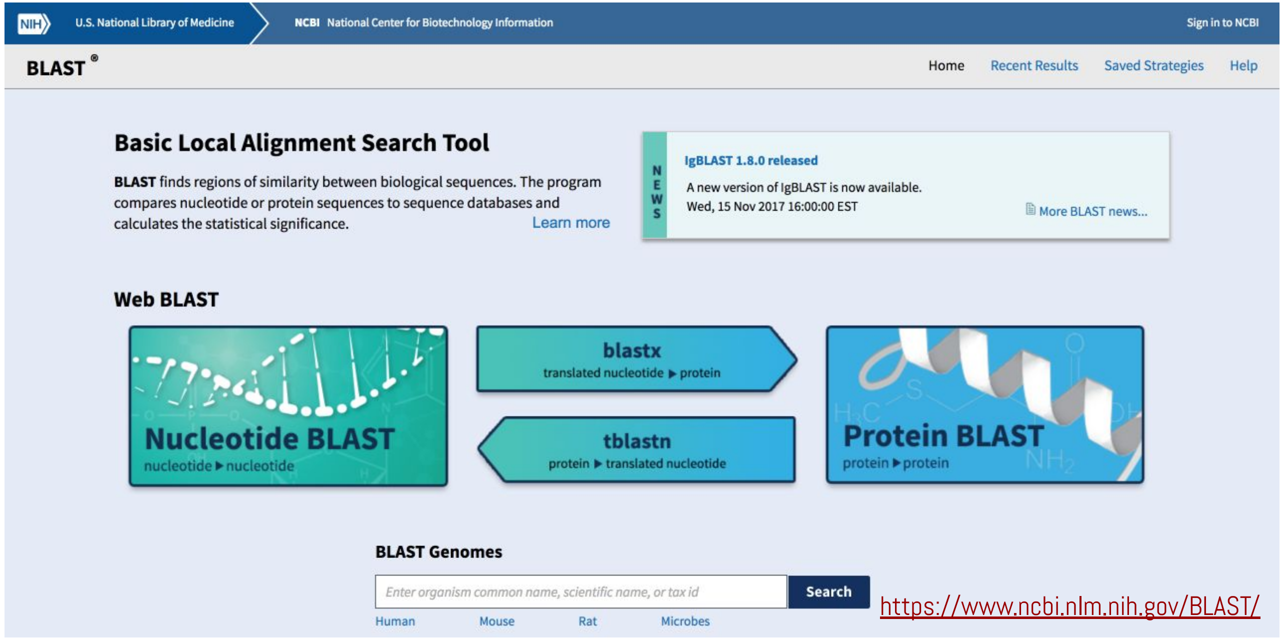
- Each score: the relative ease with which one nuc or AA may mutate into or substitute for another.
- Purely statistical, nothing directly to do with structure/biochemistry.

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

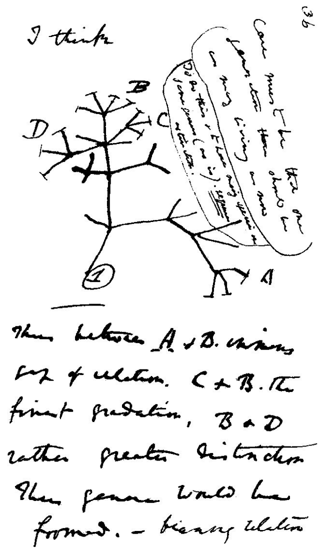
Week 2: Sequence alignment & search

Alignment

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

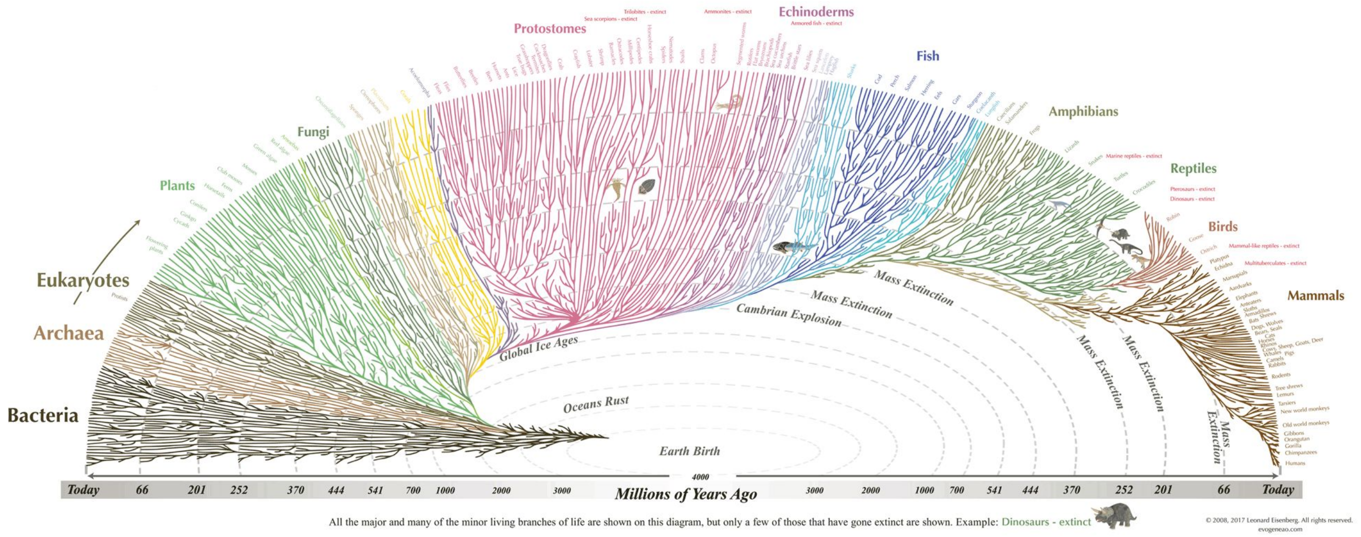
- Sign into Pear Deck using the link on Slack
    - Keep a paper and a pen(cil) ready

Sequence evolution


I think

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.

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

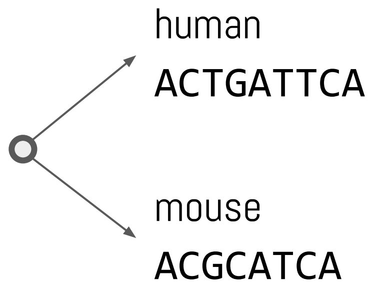


Protostomes
Echinoderms
Fish
Amphibians
Reptiles
Birds
Mammals
Fungi
Plants
Eukaryotes
Archaea
Bacteria
Global Ice Ages
Oceans Rust
Earth Birth
Cambrian Explosion
Mass Extinction
Today
66
201
252
370
444
541
700
1000
2000
3000
4000
Millions of Years Ago
All the major and many of the minor living branches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

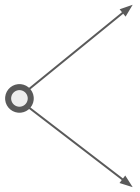
deletion — mutation — insertion

ACATGGTCA → AC*TGGTCA → ACTGATCA → ACTGATTCA

Evolutionary time



What is sequence alignment?



human
ACTGATTCA

mouse
ACGCATCA

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!

Alignment 1	Alignment 2	Alignment 3
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. 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

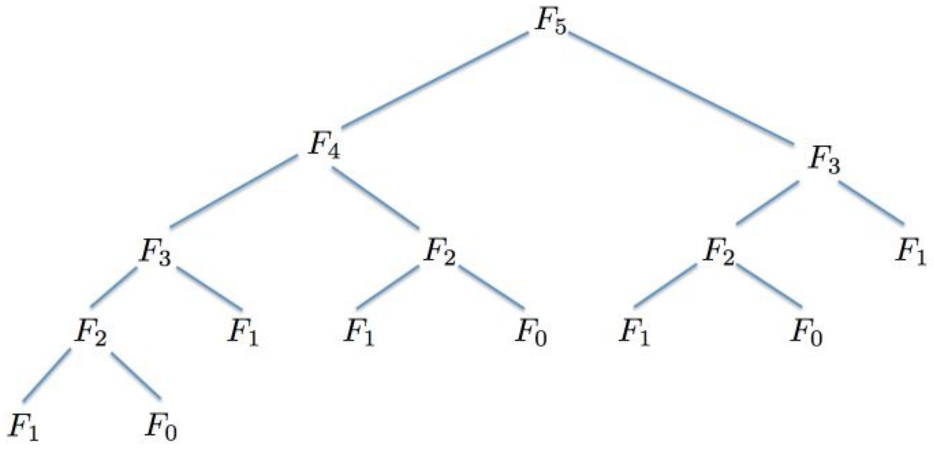
Hemachandra/Fibonacci numbers: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ……

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

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

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)
```



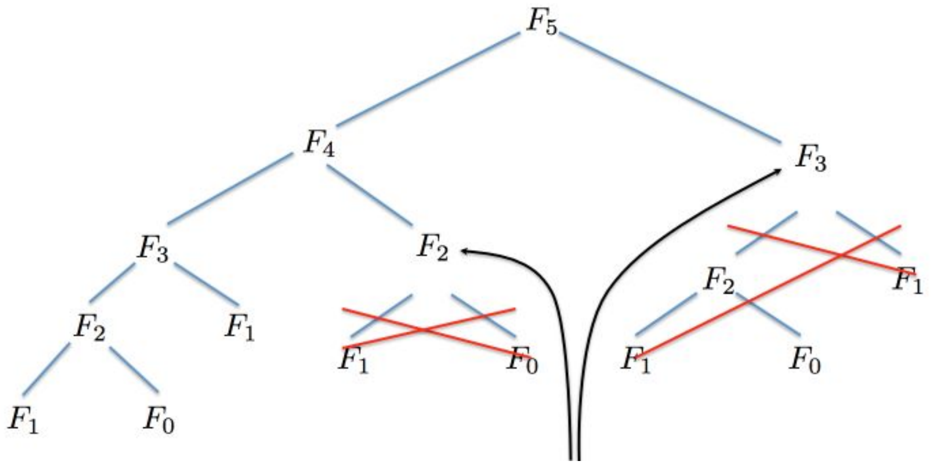
Dynamic programming

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

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

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

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

	–	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**

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

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

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

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

	–	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**

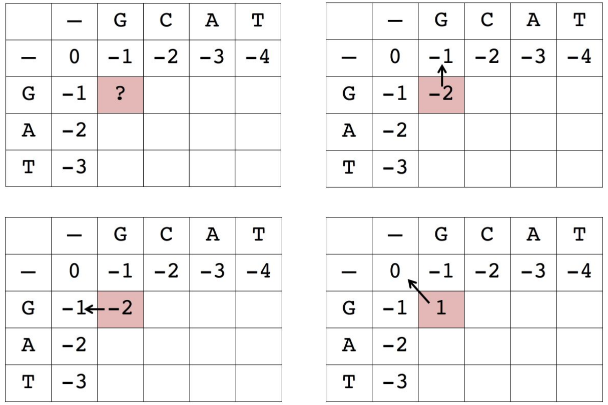
Step 2

p: gap penalty

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

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

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

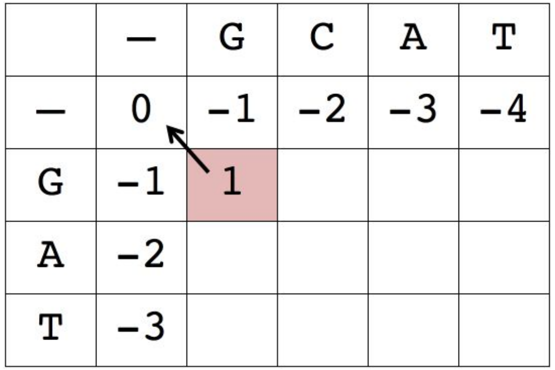


	—	G	C	A	T
—	0	−1	−2	−3	−4
G	−1	?			
A	−2				
T	−3				

	—	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.

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

p: gap penalty

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

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

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

	—	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**

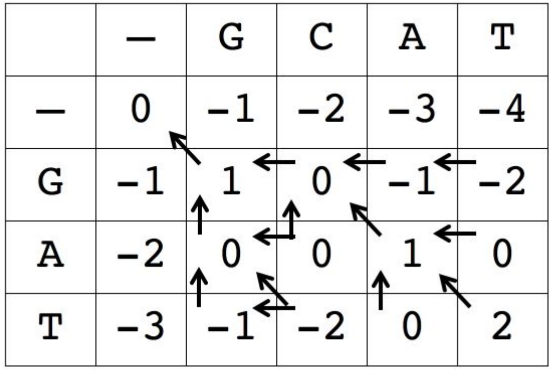
Align **GCAT** with **GAT**

Step 2

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

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

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



	–	G	C	A	T
–	0	−1	−2	−3	−4
G	−1	1	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**

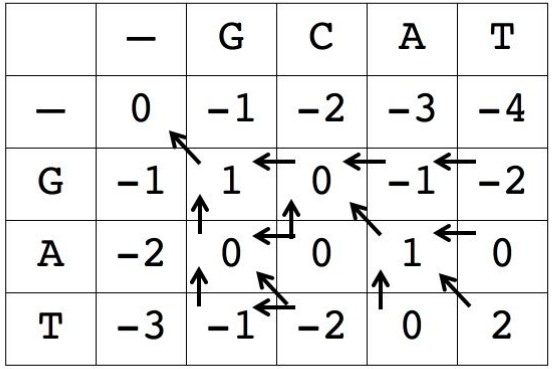
p: gap penalty

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

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

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

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**

What is the alignment?

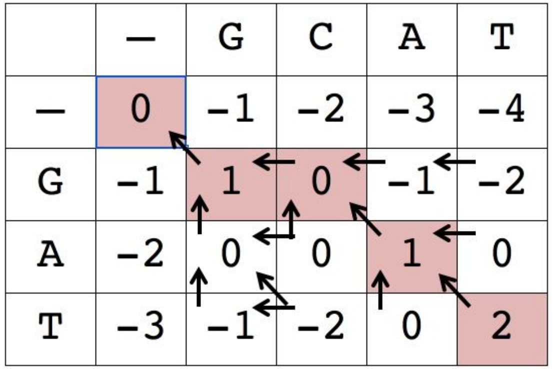
p: gap penalty

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

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

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

<u>Step 3</u>



	–	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**

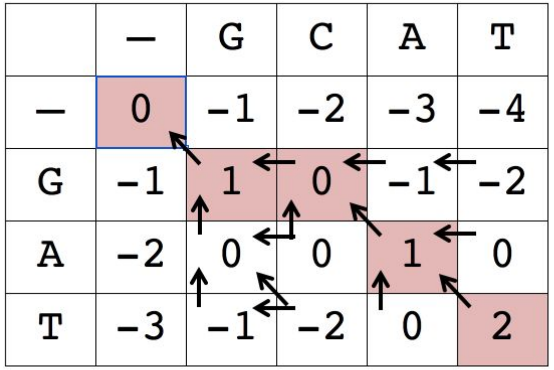
```
GCAT
G-AT
```

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

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

$M(i, j) = \text{MAX}($	$M(i-1, j) + p,$	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 **ATGCT** with **ATTACA**

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

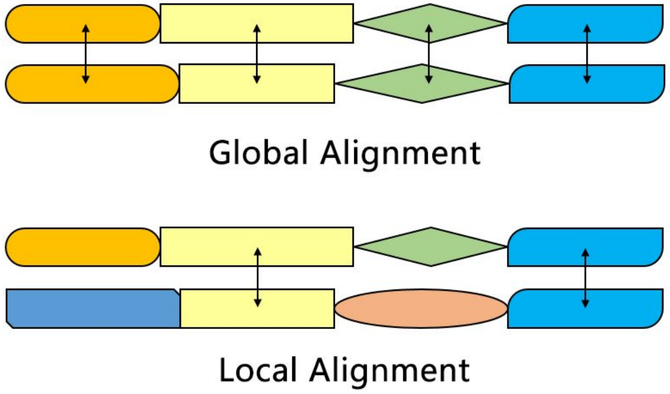
$M(0, j) = j*p;\ M(i, 0) = i*p$

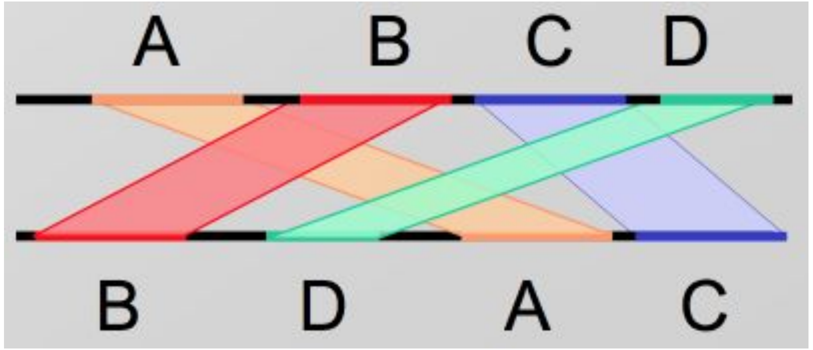
$$M(i, j) = \text{MAX}(\ M(i-1, j) + p, \quad \text{top}$$
$$M(i, j-1) + p, \quad \text{left}$$
$$M(i-1, j-1) + s(S1_i, S2_j)) \quad \text{diagonal}$$

	–	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. (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) = MAX(                  0,
                    M(i-1, j) + p,         top
                    M(i, j-1) + p,         left
          M(i-1, j-1) + s(S1_i, S2_j))     diagonal
```

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) = MAX(                  0,
                    M(i-1, j) + p,        top
                    M(i, j-1) + p,        left
          M(i-1, j-1) + s(S1_i, S2_j))    diagonal
```

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.

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

```
GC
GC
```

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

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(S1_i, S2_j))       diagonal
```

	-	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

Week 2: Sequence alignment & search

Substitution, BLAST

- Substitution matrix
  - Construction & properties
- Fast sequence searches
  - BLAST; Statistics of similarity search

Substitution matrix to measure similarity in sequence alignments

Dr. Margaret Dayhoff

Applying math & computational techniques to the sequencing of proteins and nucleic acids.

- 1965: First collection of protein seqs.

- Single-letter code for amino acids.

- 1966: ‘Evolutionary trees’.

- 1978: First AA similarity-scoring matrix.

- 1980: Launched the Protein Information Resource, the first online database system that could be accessed by telephone line.

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

- The scores represent the relative ease with which one nucleotide or amino acid 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)

Substitution matrix to measure similarity in sequence alignments

Substitution matrix: Each score is a log-odds score equal to the logarithm of the ratio of the likelihoods of two hypotheses: i) the residues can substitute for one another, or ii) not.

$$s(a,b) = \frac{1}{\lambda} \log \frac{p_{ab}}{f_a f_b}$$

- p_{ab}: likelihood of these two residues being correlated because they're homologous.
  - p_{ab} are the target frequencies: the probability that we expect to observe residues *a* and *b* aligned in homologous sequence alignments.
- $f_a f_b$: likelihood of these two residues being uncorrelated and unrelated, occurring independently.
  - f_a and f_b are background frequencies: the probabilities that we expect to observe amino acids *a* and *b* on average in any protein sequence.
- λ: a scaling factor, usually set to something that lets helps round off all the terms in the score matrix to sensible integers.

Eddy (2004)

Substitution matrix to measure similarity in sequence alignments

Substitution matrix: Each score is a log-odds score equal to the logarithm of the ratio of the likelihoods of two hypotheses: i) the residues can substitute for one another, or ii) not.

$$s(a,b) = \frac{1}{\lambda} \log \frac{p_{ab}}{f_a f_b}$$

- p_{ab}: likelihood of these two residues being correlated because they're homologous.
- $f_a f_b$: likelihood of these two residues being uncorrelated and unrelated, occurring independently.
- λ: a scaling factor

Assuming that each aligned residue pair is statistically independent of the others (biologically dubious, but mathematically convenient):

- The score of an alignment ("**alignment score**") = sum of individual log-odds scores for each aligned residue pair.

Eddy (2004)

Substitution matrix to measure similarity in sequence alignments

BLOSUM (BLOcks SUbstitution Matrix) for protein sequence alignment.

- Scan the BLOCKS database for very conserved regions of protein families (w/o gaps in the alignment).
- Count the relative frequencies of AA and their substitution probabilities.
- Calculate a log-odds score for each of the 210 possible substitution pairs of the 20 standard amino acids.

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**

Henikoff & Henikoff (1992)

Substitution matrix to measure similarity in sequence alignments

BLOSUM (BLOcks SUbstitution Matrix) for protein sequence alignment.

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
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	$\boxed{4}$									
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	$\boxed{11}$		
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

- The rarer the amino acid is, the more surprising it would be to see two of them align together by chance.
- L/L more common than WW:
  - $p_{LL} = 0.0371$, $p_{WW} = 0.0065$
- W is a much rarer amino acid:
  - $f_L = 0.099$, $f_W = 0.013$.

Check with $\lambda = 0.347$.

Eddy (2004)

Substitution matrix to measure similarity in sequence alignments

BLOSUM (BLOcks SUbstitution Matrix) for protein sequence alignment.

- A/L pairs are slightly more frequent in homologous alignments than K/E pairs:
  - $p_{AL} = 0.0044$, $p_{KE} = 0.0041$.
- But, A and L are more common amino acids:
  - $f_A = 0.074$, $f_L = 0.099$, $f_K = 0.058$, $f_E = 0.054$.

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
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	$\boxed{-1}$	−2	−3	−4	−1	−2	−3	−4	−3	2	4									
Lys	−1	2	0	−1	−3	1	$\boxed{1}$	−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

Check with $\lambda = 0.347$.

Eddy (2004)

Substitution matrix to measure similarity in sequence alignments

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**

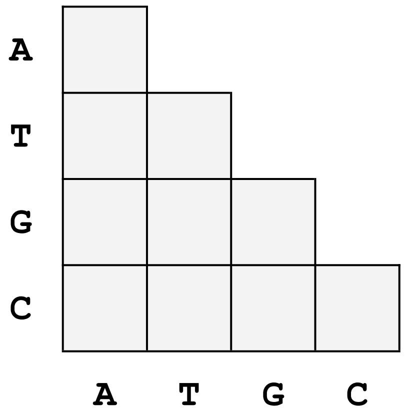
BLOSUMr: the matrix built from blocks with less than r% of similarity.

- E.g., BLOSUM62: built using sequences with less than 62% similarity (sequences with ≥ 62% identity were clustered)
  - BLOSUM62 among the best for detecting most weak protein similarities.
  - Default matrix for protein BLAST.

Eddy (2004)

Substitution matrix to measure similarity in sequence alignments

Substitution matrix for DNA



Making-up an arbitrary matrix by fixing the p_{ab} values ⟶ directly describes what homologous alignments are expected to look like.

- The resulting score matrix is optimal for detecting alignments that match these target frequencies.

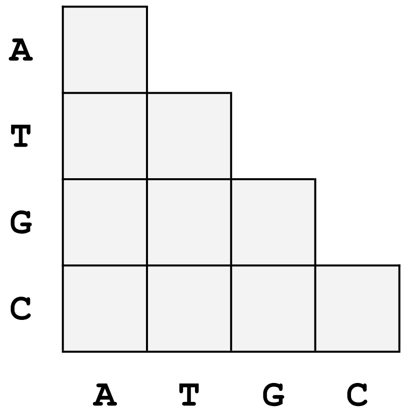
Say, the matrix should be optimized for finding 88% identity alignments.

- Assume that all mismatches are equiprobable, and composition of both alignments and background sequences is uniform at 25% for each nucleotide (f_a, f_b = 0.25 for all a,b). Then,
  - Four identities: $p_{aa} = 0.22$
  - 12 types of mismatch: $p_{ab} = 0.01$.
- If we set $\lambda = 1$, this gives +1.26 for a match and −1.83 for a mismatch.
- Setting $\lambda = 0.25$ and round off: we have a new scoring system of +4/−7.

Eddy (2004)

Substitution matrix to measure similarity in sequence alignments

Substitution matrix for DNA



Given a scoring matrix, we can back calculate target frequencies if two conditions are met:

$$s(a,b) = \frac{1}{\lambda} \log \frac{p_{ab}}{f_a f_b}$$

1. It must have at least one positive score, and
2. The expected score for random sequence alignments must be negative.

True for most score matrices:

- These properties are necessary to make local sequence alignment algorithms like BLAST and Smith-Waterman work.
- Both conditions are met by definition for matrices derived as log-odds scores, except for the useless case of $p_{ab} = f_a f_b$ for all a,b.

Examples:

- FASTA & WU-BLASTN: arbitrary +5/−4 scoring system;
  Optimal for detecting alignments that are 65% identical.
- NCBI BLASTN: +1/−2 scoring system; Optimal for detecting alignments that are 95% identical.

Eddy (2004)

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?

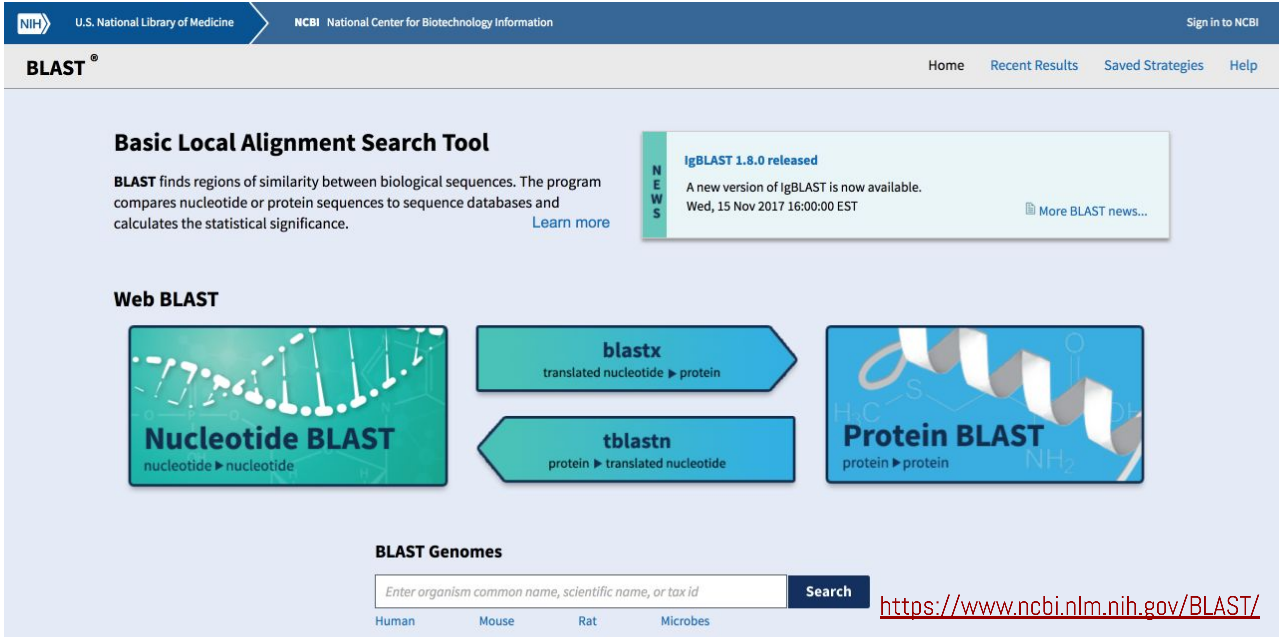
- Individual alignments need not be perfect: Once initial matches are found, they can fine-tune them later.
- Must be very fast.

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

- If any match with % identity ≤ 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.

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



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

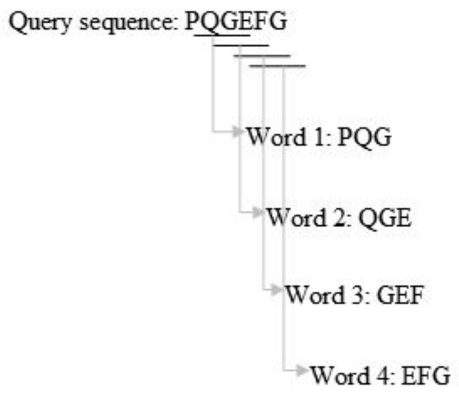
BLAST Genomes

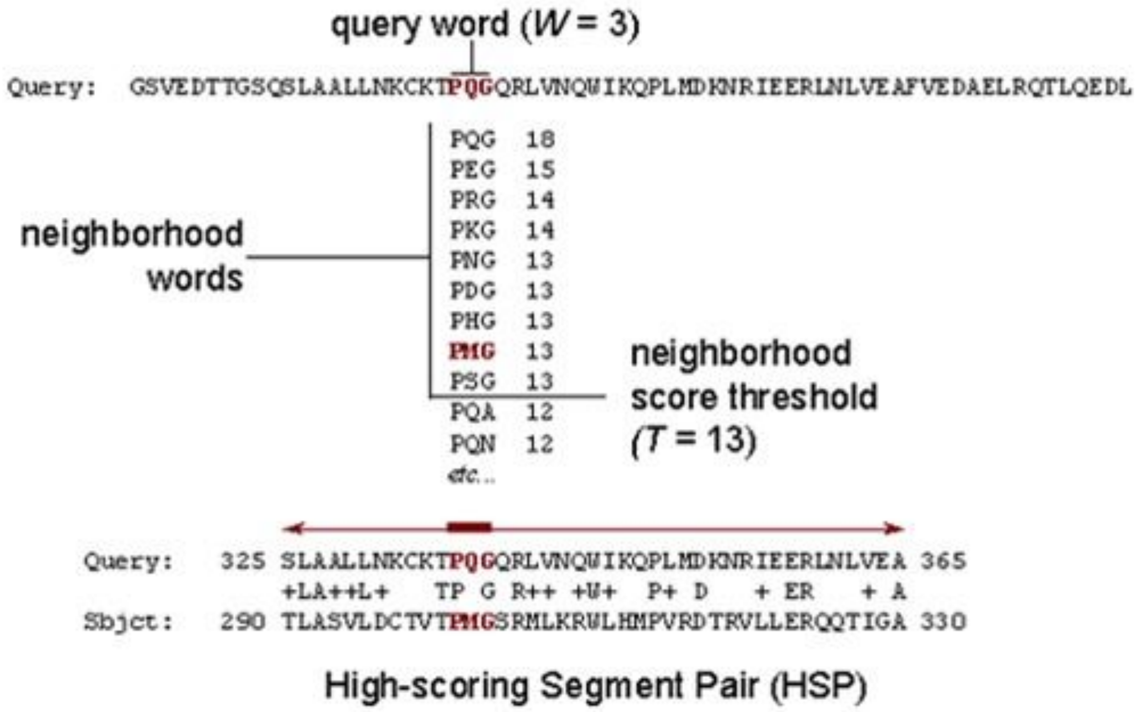
Enter organism common name, scientific name, or tax id · Search

Human · Mouse · Rat · Microbes

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

BLAST



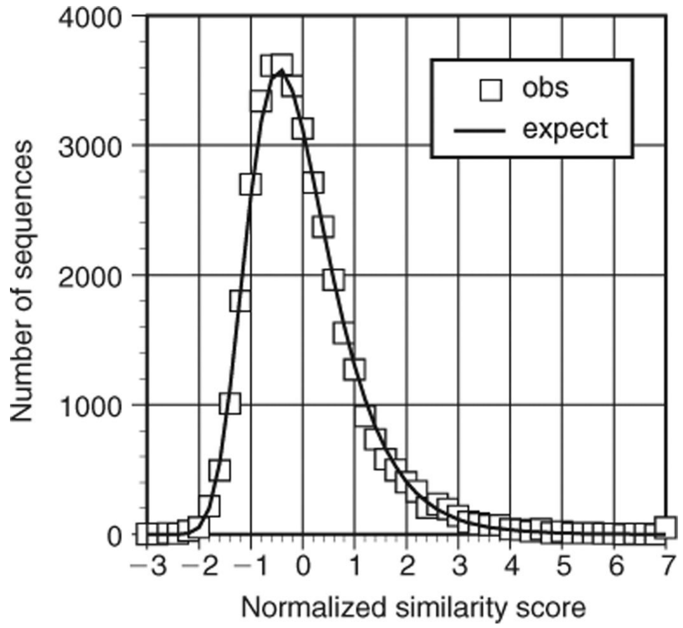

query word (W = 3)
Query: GSVEDTTGSQSLAALLNKCKTPQGQRLVNQWIKQPLMDKNRIEERLNLVEAFVEDAELRQTLQEDL
neighborhood words
PQG 18
PEG 15
PRG 14
PKG 14
PNG 13
PDG 13
PHG 13
PMG 13
PSG 13
PQA 12
PQN 12
etc...
neighborhood score threshold (T = 13)
Query: 325 SLAALLNKCKTPQGQRLVNQWIKQPLMDKNRIEERLNLVEA 365
+LA++L+ TP G R++ +W+ P+ D + ER + A
Sbjct: 290 TLASVLDCTVTPMGSRMLKRWLHMPVRDTRVLLERQQTIGA 330
High-scoring Segment Pair (HSP)

NCBI Handbook: https://www.ncbi.nlm.nih.gov/books/NBK153387/

Some uses of BLAST

- Finding the right/relevant species:
  - If you have a DNA sequence from unknown species, BLAST can help identify the correct/related species.
- Finding protein domains:
  - If you a protein sequence (or a translated nucleotide sequence), BLAST can be used to look for known protein domains in the query sequence.
- Mapping the phylogeny of a gene/protein:
  - BLAST can be used to find potential homologs of your gene/protein of interest across many species, which you can then use to generate a phylogenetic tree.
- Mapping DNA to a known chromosome:
  - If you are sequencing a gene from a known species but have no idea of the chromosome location, BLAST can help you. BLAST will show you the position of the query sequence in relation to the hit sequences.
- Annotations:
  - BLAST can also be used to map gene/protein annotations from one organism to another.

Statistics of similarity search



Distribution of real (squares) & expected similarity scores (Gumbel extreme value distribution).

P-value:

- The probability of observing a score equal to or greater than the observed score S.

E-value:

- The expected number of HSPs with score at least S.
- $E = Kmne^{-\lambda S}$

Database E-value:

- E-value after thousands/millions of searches ≈ E*D.

Bit score:

- Normalized raw score.