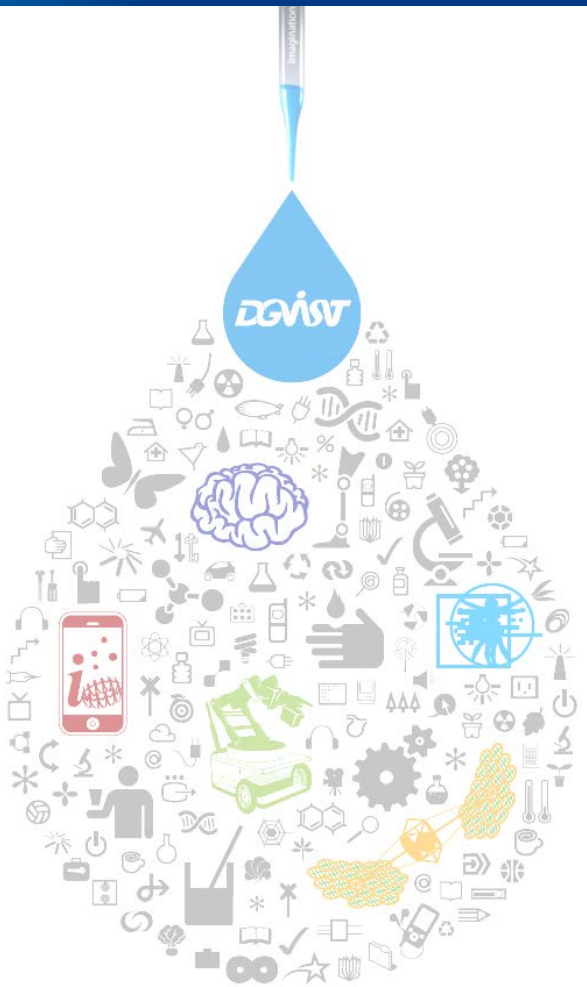
InfoLab

CR521
Neuroinformatics
Sequence Alignment

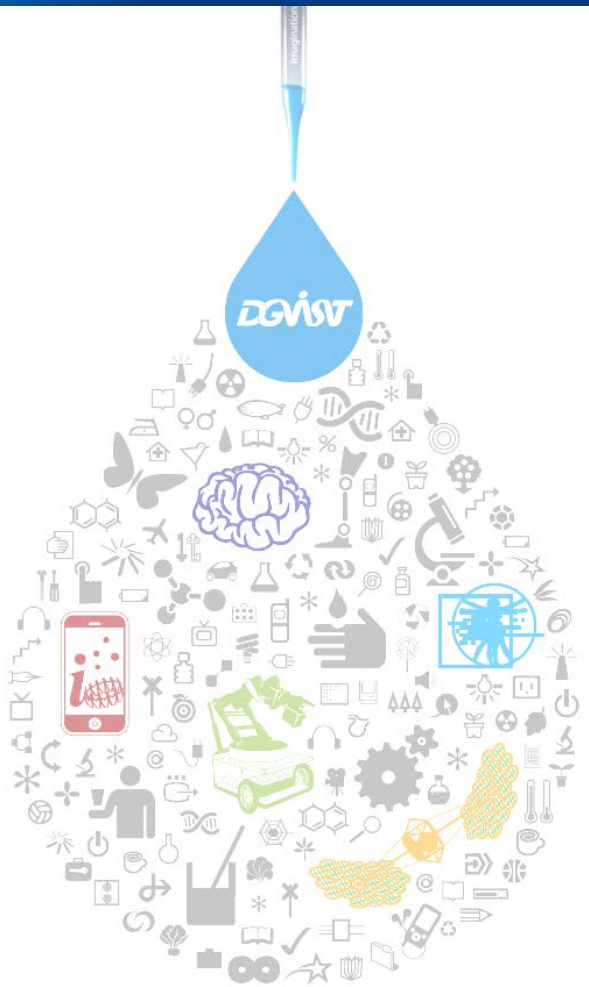


2013.5.6

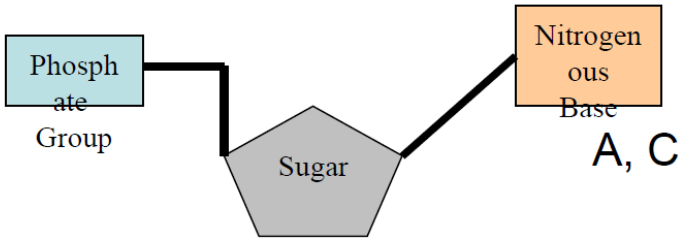
Min-Soo Kim

DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology

Background

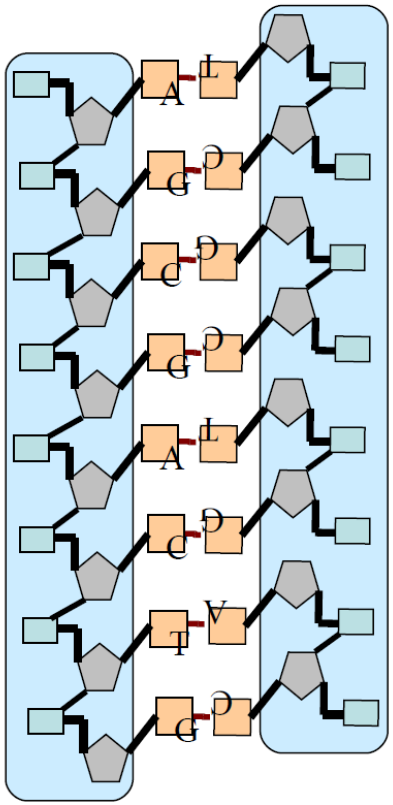


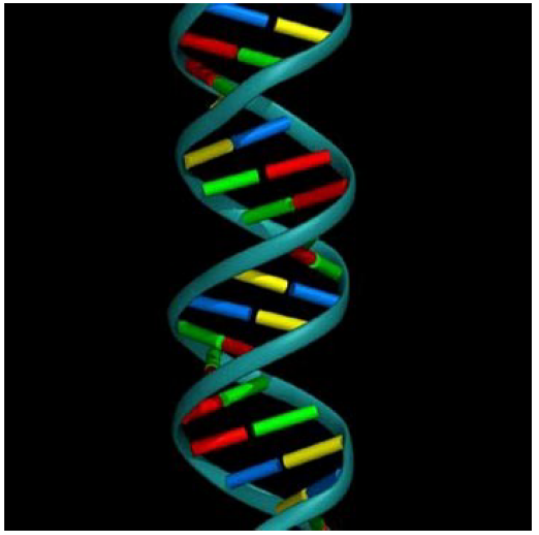
Birth of Molecular Biology

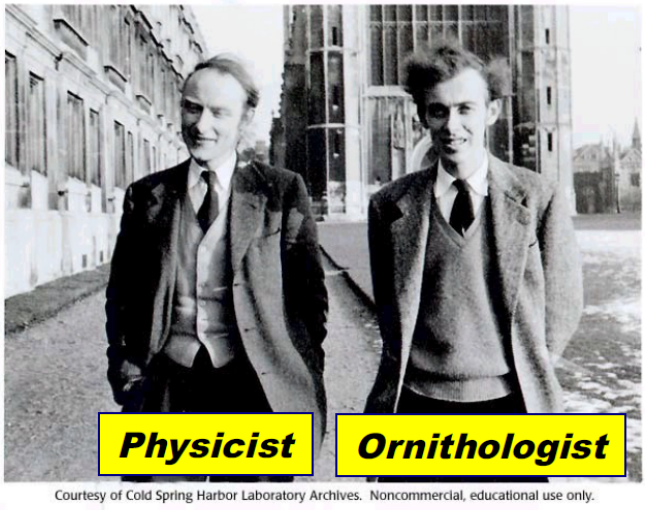


A, C, G, T

DNA



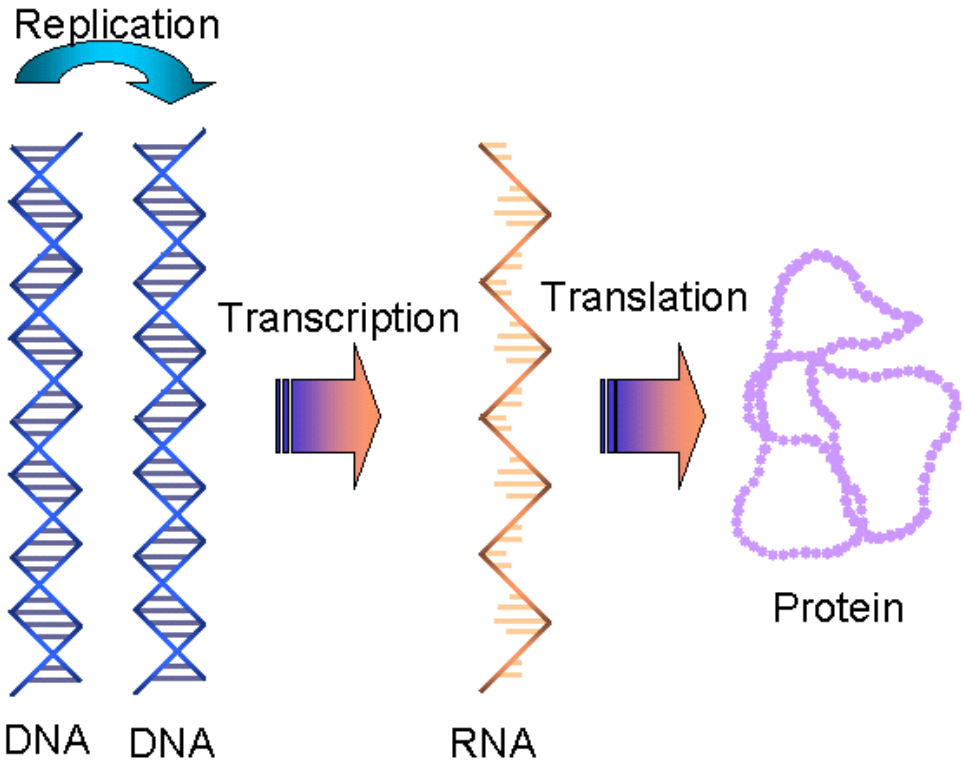


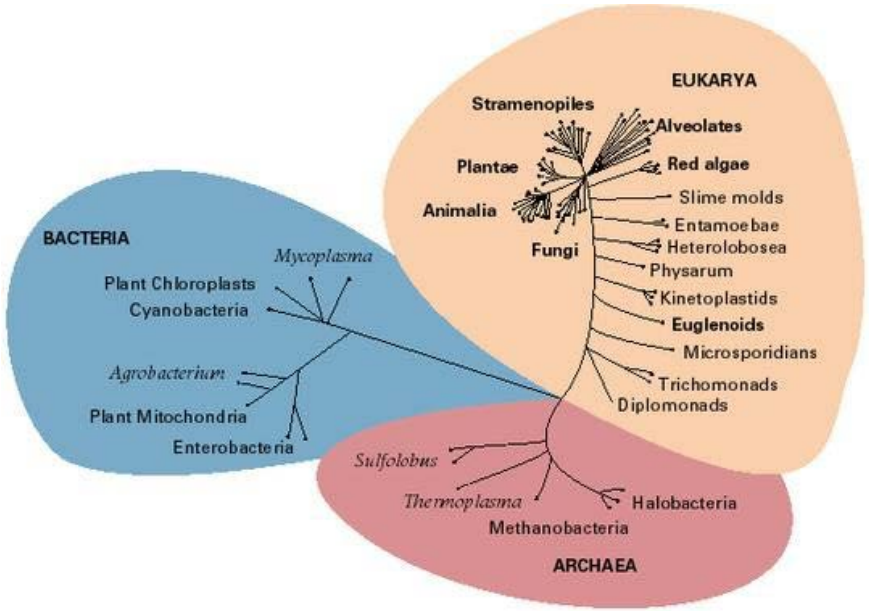


Courtesy of Cold Spring Harbor Laboratory Archives. Noncommercial, educational use only.

from http://cs262.stanford.edu

Genetics in the 20th Century





from http://cs262.stanford.edu

Human Genome Project

1990: Start

2000: Bill Clinton: *"most important scientific discovery in the 20th century"*

2001: Draft

2003: Finished

3 billion basepairs

$3 billion

now what?

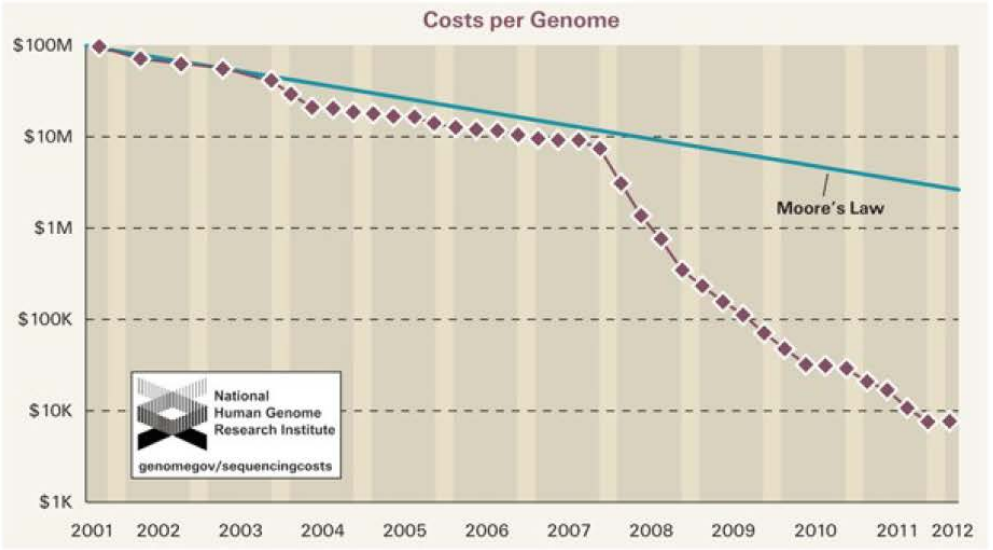
from http://cs262.stanford.edu

Sequencing Growth

Cost of one human genome

- 2004: $30,000,000
- 2008: $100,000
- 2010: $10,000
- **2013: $2,000 (today)**
- ???: $300

How much would you pay for a smartphone?



from http://cs262.stanford.edu

Uses of Genomes

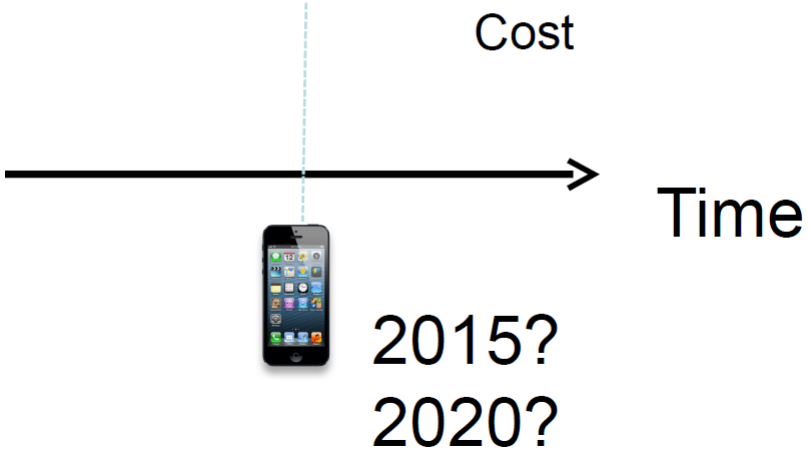
- Medicine
  - Mendelian diseases
  - Cancer
  - Drug dosage (eg. Warfarin)
  - Disease risk
  - Diagnosis of infections
  - …
- Ancestry
- Genealogy
- Nutrition?
- Psychology?
- Baby Engineering?...

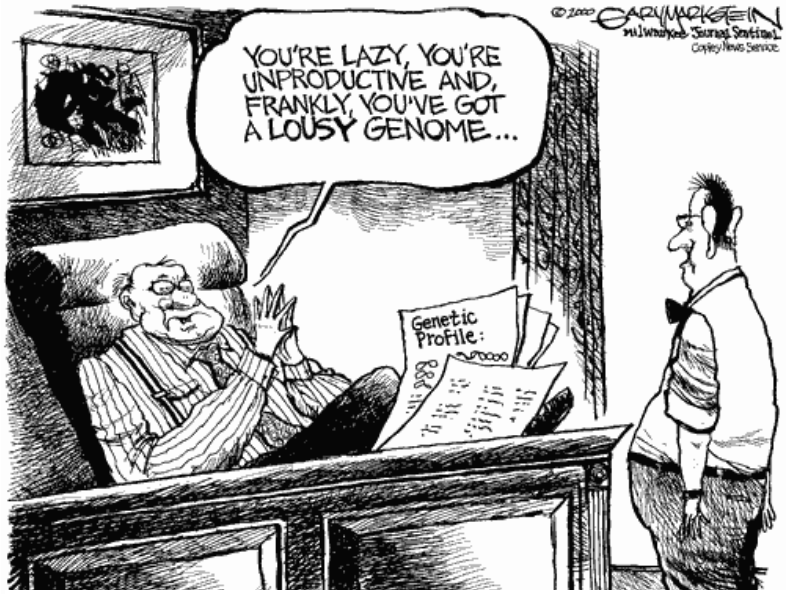
- **Ethical Issues**

from http://cs262.stanford.edu

How soon will we all be sequenced?

- Cost
- Killer apps
- Roadblocks?





from http://cs262.stanford.edu

Complete DNA Sequences





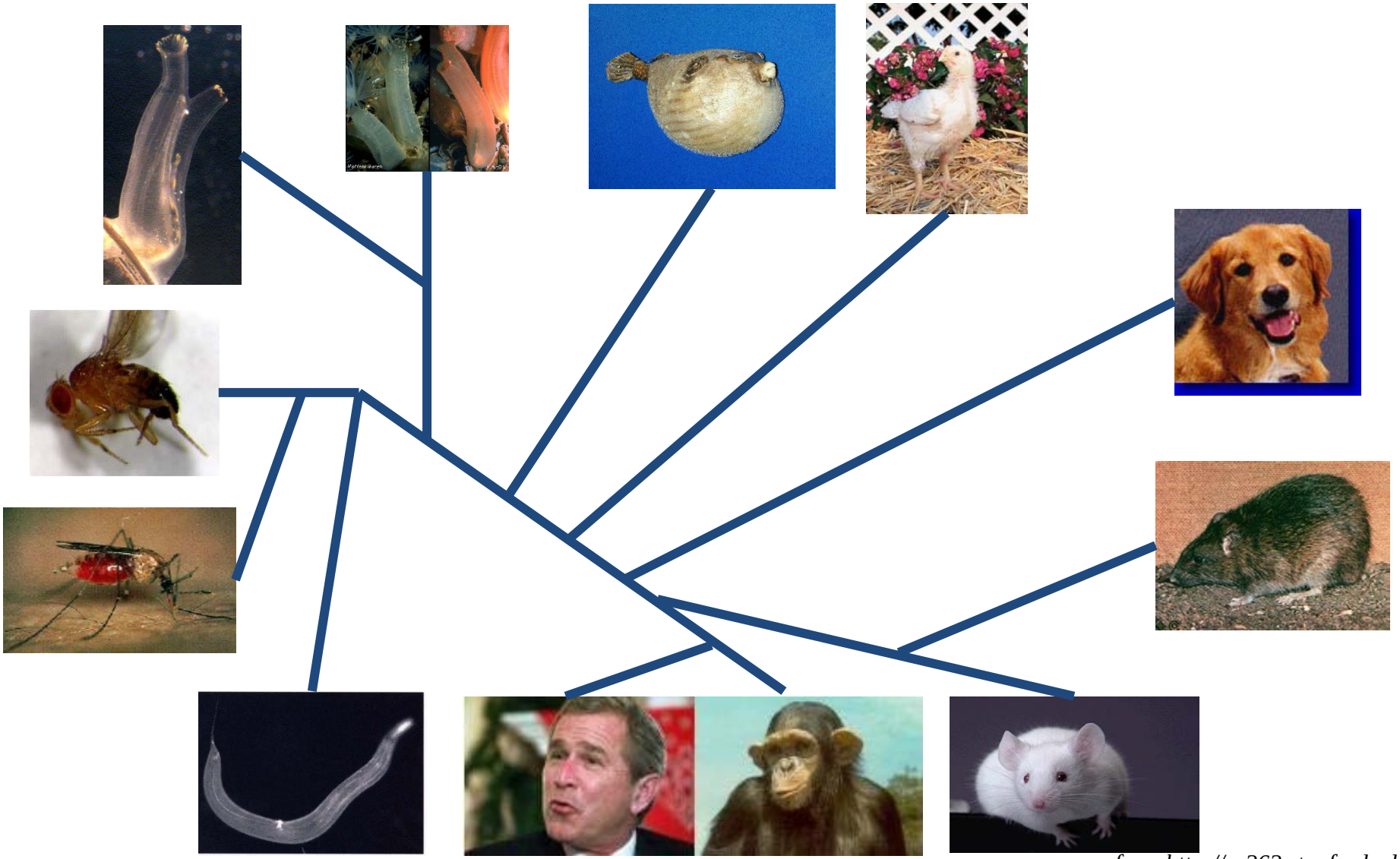

More than 1,000 complete genomes have been sequenced





from http://cs262.stanford.edu

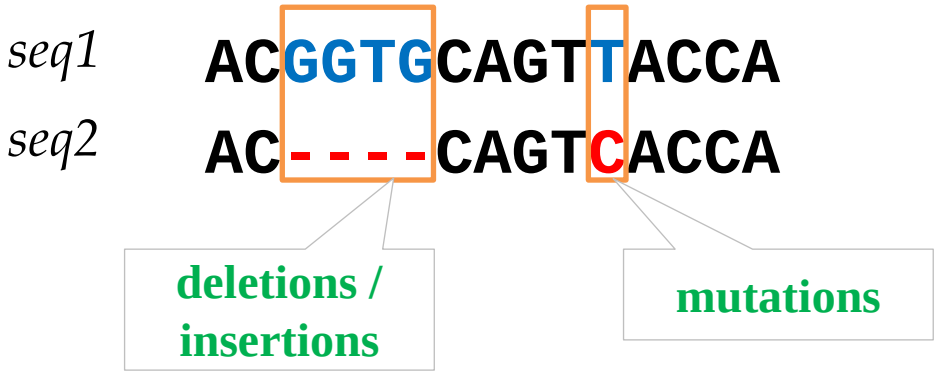
Complete DNA Sequences



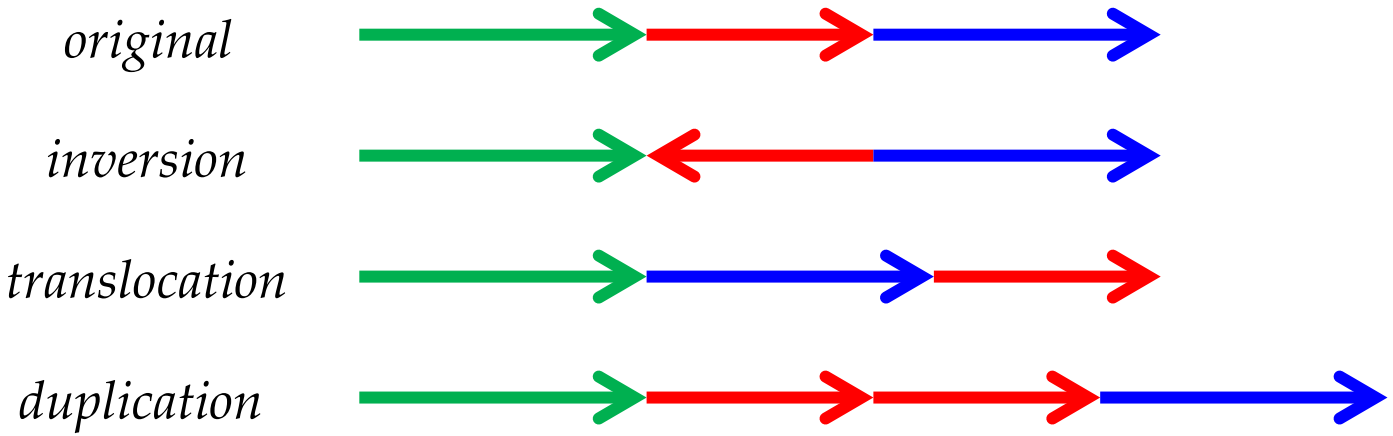
from http://cs262.stanford.edu

Evolution at the DNA level

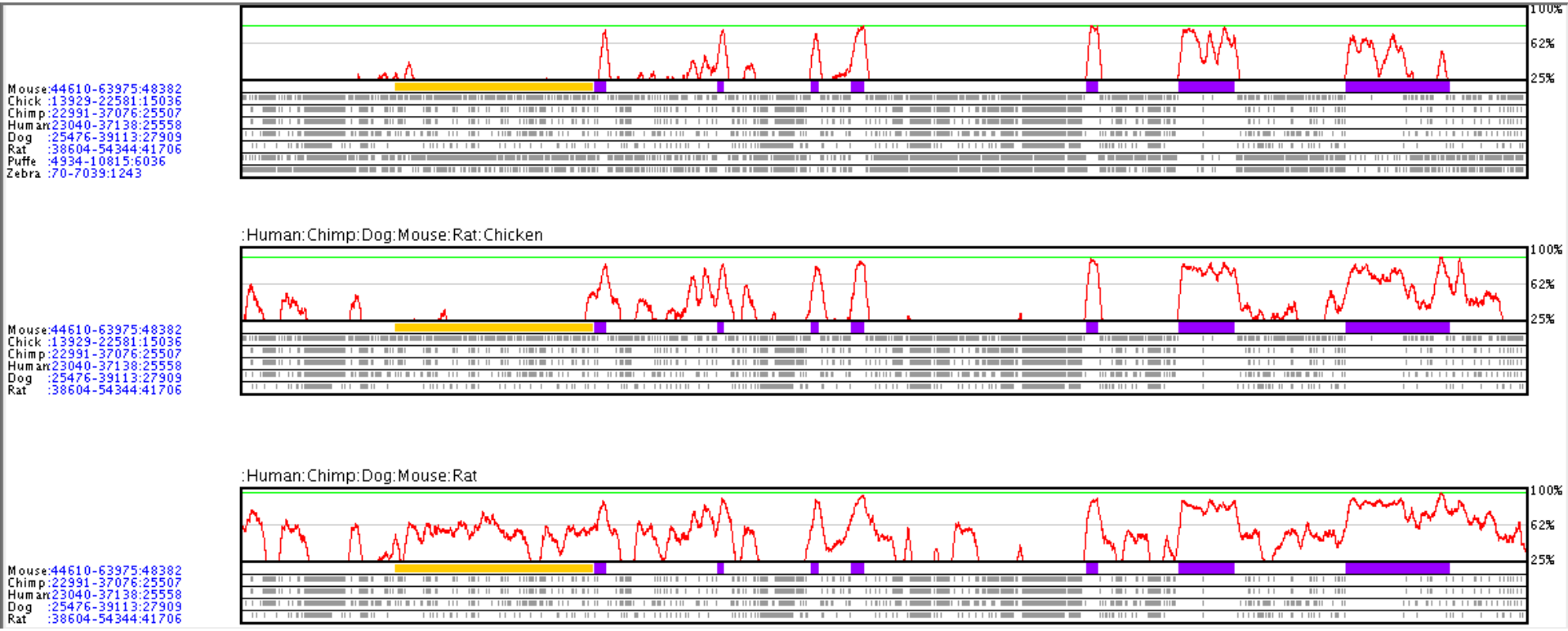
- **Sequence edits**



- **Rearrangements**

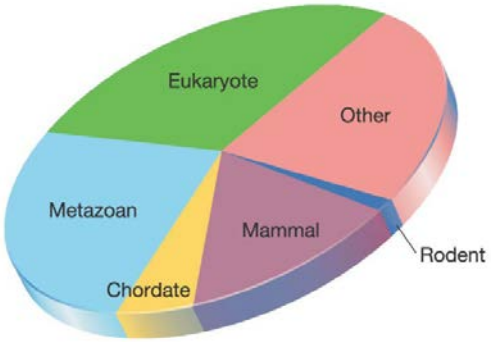


Sequence conservation implies function



Alignment is the key to

- Finding important regions
- Determining function
- Uncovering evolutionary events

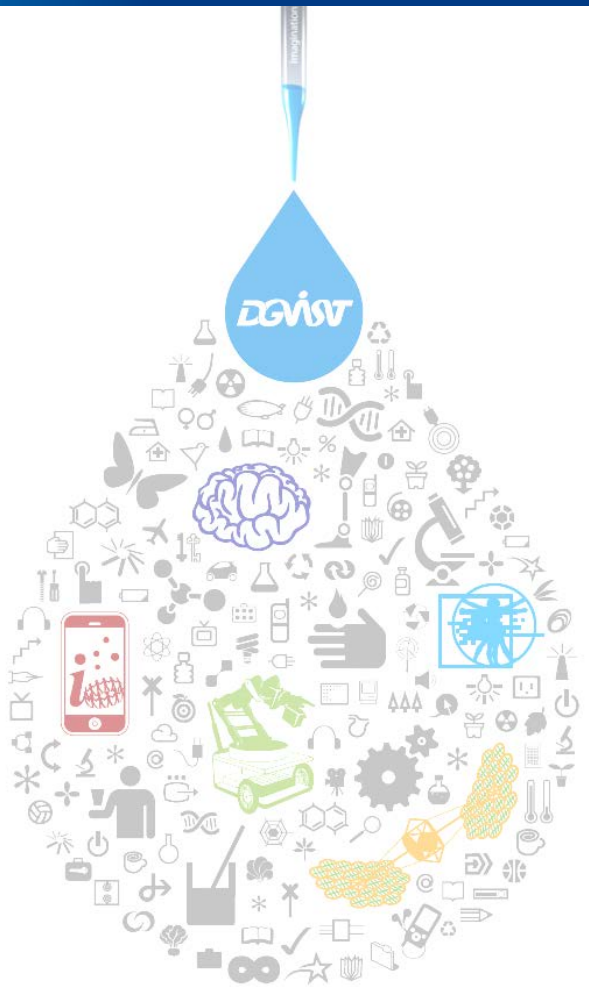


from http://cs262.stanford.edu

InfoLab

Pairwise Sequence Alignment

Exact methods



DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology

Sequence Alignment

- **Definition**

  Given two strings $x = x_1x_2...x_M$, $y = y_1y_2...y_N$,
  an alignment is an assignment of gaps to positions
  $0,\ldots, N$ in x, and $0,\ldots, N$ in y, so as to line up each letter in one sequence with either a letter, or a gap in the other sequence

- **Example**

  x AGGCTATCACCTGACCTCCAGGCCGATGCCC

  y TAGCTATCACGACCGCGGTCGATTTGCCCGAC

  -AGG**CTATCAC**C**T**G**ACC**T**C**C**A**G**G**C**CGA--TGCCC---**

  TA**G-CTATCAC--GACC**G**C--GG**T**CGA**T**T**TGCCC**GAC

from http://cs262.stanford.edu

What is a good alignment?

AGGCTAGTT,
AGCGAAGTTT

1
AGGCTAGTT-
AGCGAAGTTT
6 matches, 3 mismatches, 1 gap

2
AGGCTA-GTT-
AG-CGAAGTTT
7 matches, 1 mismatch, 3 gaps

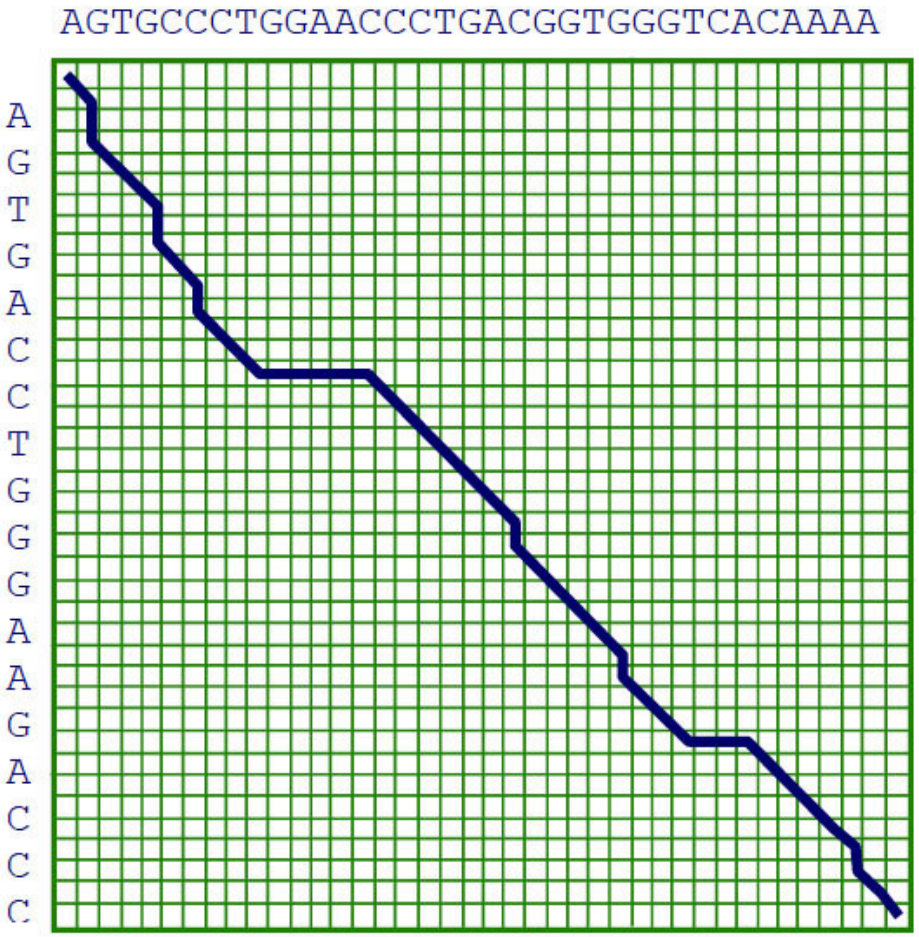
3
AGGC-TA-GTT-
AG-CG-AAGTTT
7 matches, 0 mismatches, 5 gaps

from http://cs262.stanford.edu

Scoring Function

- **Scoring function**
  - match : $+m$
  - mismatch : $-s$
  - gap : $-d$
  - Scoring function $F = (\#matches)*m + (\#mismatches)*s + (\#gaps)*d$
  - Example : m = +1, s = -1, d = -2
    - Case 1's F = 6*1 – 3*1 – 1*2 = 1 (*better*)
    - Case 2's F = 7*1 – 1*1 – 3*2 = 0
- **Alternative: minimal edit distance (MED)**
  - Given two strings x, y, find minimum # of edits (insertions, deletions, mutations) to transform one string to the other
  - Example :
    - Case 1's MED = 3 + 1 = 4
    - Case 2's MED = 1 + 3 = 4

How do we compute the best alignment?



Too many possible alignments!!

$$>> 2^N$$

from http://cs262.stanford.edu

Alignment is additive

- The score of aligning

$x_1 \ldots\ldots\ldots x_M$

$y_1 \ldots\ldots\ldots y_N$

- is the same as adding up two sub-alignments

$x_1 \ldots x_i \quad x_{i+1} \ldots x_M$

$y_1 \ldots y_j \quad y_{j+1} \ldots y_N$

- The score function becomes:

$$F(x[1{:}M], y[1{:}N]) = F(x[1{:}\mathbf{i}], y[1{:}\mathbf{j}]) + F(x[\mathbf{i{+}1}{:}M], y[\mathbf{j{+}1}{:}N])$$

Dynamic programming

- Why it works for DNA alignment?
  - There are only a polynomial number of subproblems

    Align $x_1 \ldots x_i$ to $y_1 \ldots y_j$ where $1 \le i \le M$ and $1 \le j \le N$

  - Original problem is one of the subproblems

    Align $x_1 \ldots x_M$ to $y_1 \ldots y_N$

  - Each subproblem is easily solved from smaller subproblems
    - The general idea is to construct longer alignments with shorter alignments

- So, we can apply *Dynamic Programming!!!*

DP: How it works?

- We build a dynamic programming (DP) “Matrix” or “Table” called F
  - `F(i,j)` = optimal score of aligning between $x_1 \ldots x_i$ and $y_1 \ldots y_j$

- This process, called *Memorization*, is to store the optimal scores of subproblems for solving larger problems

DP: Three cases when building matrix

① x_i aligns to y_j

$x_1 \ldots x_{i-1}$	x_i
$y_1 \ldots y_{j-1}$	y_j

$$F(i,j) = F(i-1,j-1) + \begin{cases} m & if\ x_i = y_j \\ -s & if\ not \end{cases}$$

② x_i aligns to a gap

$x_1 \ldots x_{i-1}$	x_i
$y_1 \ldots y_j$	$-$

$$F(i,j) = F(i-1,j) - d$$

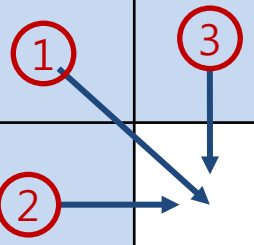
③ y_j aligns to a gap

$x_1 \ldots x_i$	$-$
$y_1 \ldots y_{j-1}$	y_j

$$F(i,j) = F(i,j-1) - d$$

$F(i, j)$

		…	x_{i-1}	x_i	x_{i+1}	…
	0	…	$-i+1$	$-i$	$-i-1$	…
…	…					
y_{j-1}	$-j+1$					
y_j	$-j$					
y_{j+1}	$-j-1$					
…	…					



DP: Optimal Score

- By **inductive assumption**, *F(i, j-1), F(i-1, j)*, and *F(i-1, j-1)* are optimal

- We choose the one with the max score

$$F(i,j) = max \begin{cases} F(i-1,j-1) + s(x_i, y_j) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases} \qquad where\ s(x_i, y_j) = \begin{cases} m & if\ x_i = y_j \\ -s & if\ not \end{cases}$$

DP: Procedure

① Initialization

- one alignment of a letter to a gap has a score of -1
- we fill in the first row and the first col with (-1 × the corresponding index)

② Iteration : filling in the rest by the formula

③ Termination

- we reach the bottom right corner of the matrix
- we backtrace from the bottom right corner to the top left corner with the pointers and output the alignment
  - When in the diagonal direction, output x_i and y_j
  - When in the up direction, output y_j
  - When in the left direction, output x_i

DP: Example

$x = AGTA$
$y = ATA$

$m = 1$
$s = -1$
$d = -1$

$$F(1,1) = max \begin{cases} F(0,0) + s(A,A) \\ F(0,1) - d \\ F(1,0) - d \end{cases} = 1$$

F(i,j)	i = 0	1	2	3	4
		A	G	T	A
j = 0	0	-1	-2	-3	-4
1 A	-1	1	0	-1	-2
2 T	-2	0	0	1	0
3 A	-3	-1	-1	0	2

output

AGTA
A-TA

Global Alignment (Needleman-Wunsch algorithm)

- General idea is the same with DP
- Every nondecreasing path from (0,0) to (M,N) corresponds to an alignment of the two sequences

- Main iteration (filling in partial alignments)

$$F(i,j) = max \begin{cases} F(i-1,j-1) + s(x_i, y_j) & [\text{case 1}] \\ F(i-1,j) - d & [\text{case 2}] \\ F(i,j-1) - d & [\text{case 3}] \end{cases}$$

$$Ptr(i,j) = \begin{cases} DIAG & \text{if } [\text{case 1}] \\ LEFT & \text{if } [\text{case 2}] \\ UP & \text{if } [\text{case 3}] \end{cases}$$

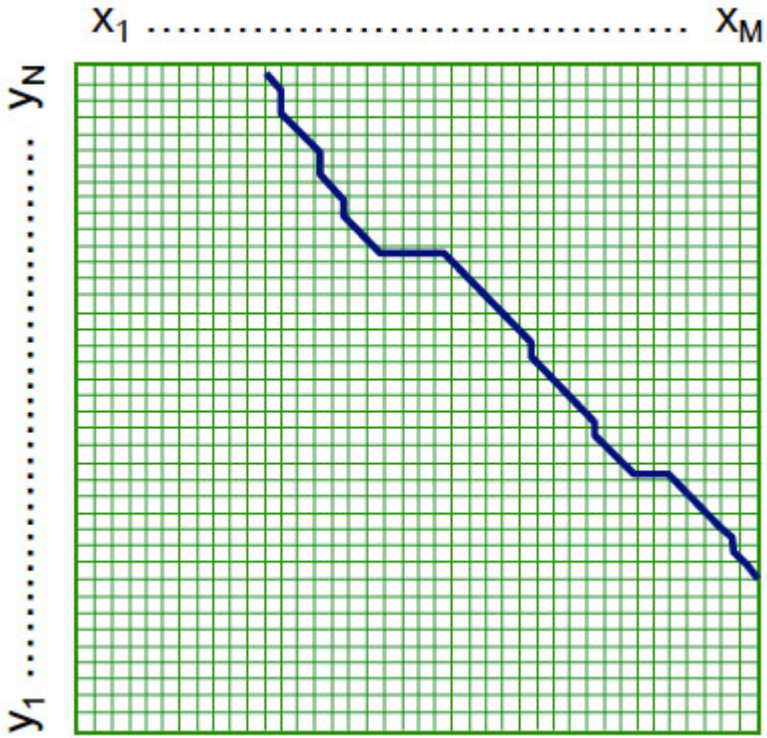
- Termination
  - $F(M,N)$ is the optimal score
  - we can trace back the optimal alignment from $Ptr(M,N)$

Overlap Detection Variant

- In some cases, it's OK to have an unlimited # of gaps in the beginning and the end
  - e.g., DNA assembly in a sequencing project
  - e.g., searching for a small gene within a large chromosome

- We don't want to penalize gaps in the ends
  - need to slightly modify Needleman-Wunsch algorithm to not penalize the gaps at both ends

- Notice that the path can start from any column in the first row and end in any row of the last column



1. Initialization.
   For all i, j:
   $$F(i, 0) = 0$$
   $$F(0, j) = 0$$

2. Termination.
   $$F_{OPT} = max \begin{cases} max_i F(i, N) \\ max_j F(M, j) \end{cases}$$

Local alignment (Smith-Waterman algorithm)

- Given two strings

  $x_1 . . . . . x_M$

  $y_1 . . . . . y_N$

- we want to find substrings x' and y' whose similarity (optimal global alignment value) is maximum

- e.g, the local alignment of x and y is x' = y' = cccggg

$$\begin{array}{ccc} & & x' \\ x & = & \text{aaaacc}\boxed{cccggg}\text{gtta} \\ y & = & \text{tt}\boxed{cccggg}\text{aaccaacc} \\ & & y' \end{array}$$

- Compared to the overlap problem, local alignment doesn't have to start from one of the ends of the sequences

- Local alignments can be used to
  - identify genes that are shuffled between genomes
  - identify conserved regions of portions of proteins (domains)

- Smith-Waterman algorithm
  - it can be obtained by making subtle modifications to the Needleman-Wunsch algorithm
  - idea is to ignore badly aligned regions
  - we replace a negative score in the DP matrix with a 0 to disregard the bad alignments

Procedure of Smith-Waterman Algorithm

① Initialization

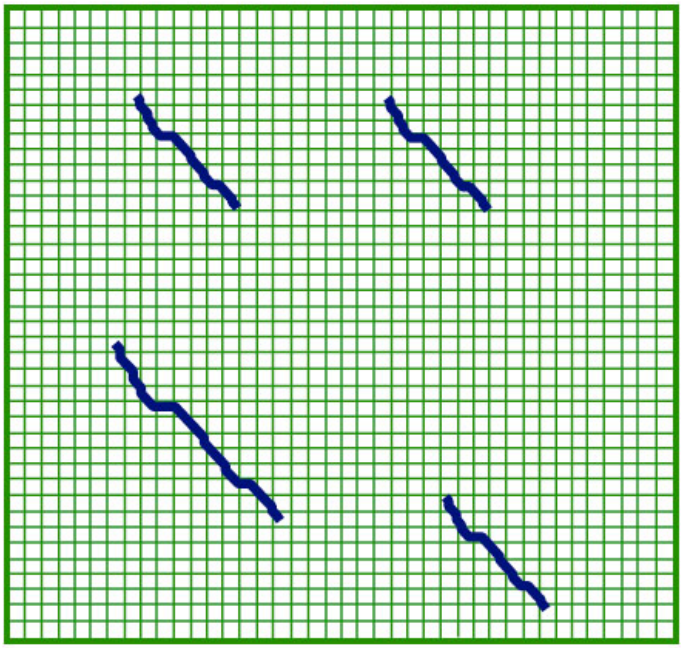
$$\text{For all } i, j\text{:}$$
$$F(i, 0) = 0$$
$$F(0, j) = 0$$

② Iteration

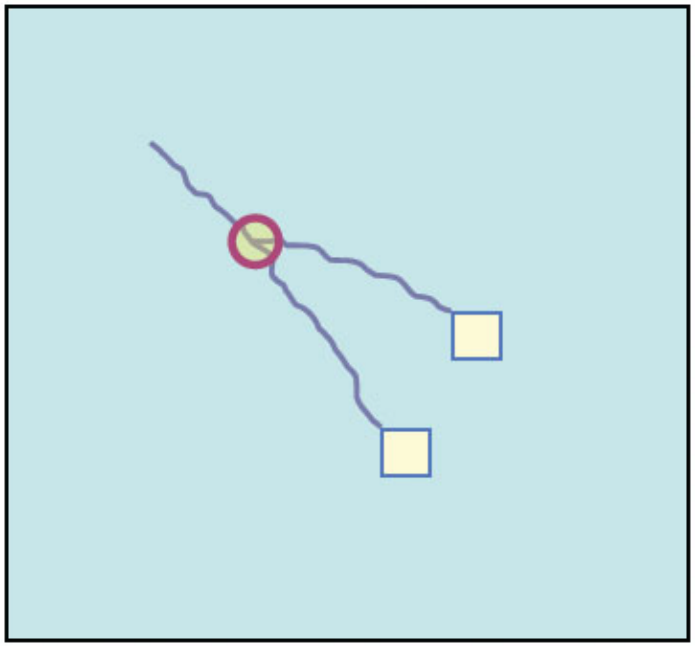
$$F(i, j) = max \begin{cases} 0 \\ F(i-1, j) - d \\ F(i, j-1) - d \\ F(i-1, j-1) + s(x_i, y_j) \end{cases}$$

③ Termination

- If we want the best local alignment, then we find $F_{OPT} = max_{i,j} F(i, j)$ and trace back
- If we want all local alignments scoring > t, then the problem becomes complicated by *overlapping local alignments*



Four possible local alignments



Two overlapping local alignments :

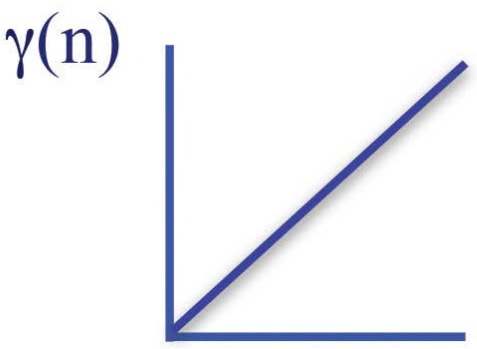
- Two squares represent the ends of local alignments
- In backtracing, the two paths will merge at some point shown as a circle

Scoring gaps more accurately

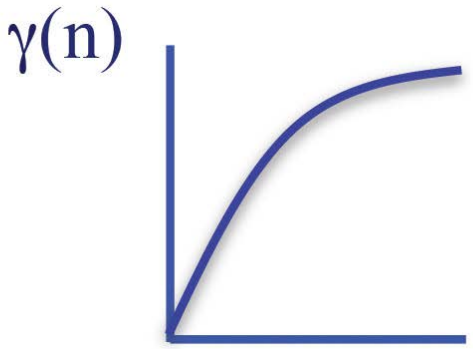
- **Observation**
  - gaps usually appear in bunches.
  - it's better to reduce gap penalty as the gap length increases instead of penalizing every gap alignment equally

- **Convex gap model**

$$\gamma(n):$$
$$\text{for all } n,\ \gamma(n+1)-\gamma(n) \leq \gamma(n)-\gamma(n-1)$$



regular gap model

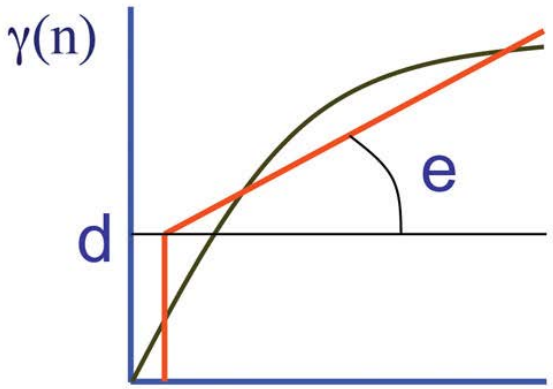


convex gap model

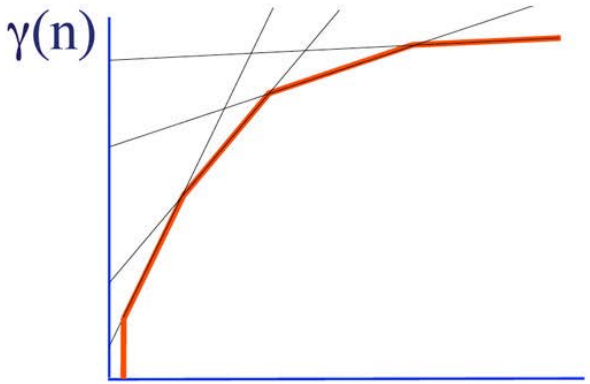
Affine gap model

- d : gap open penalty
- e : gap extension penalty

$$\gamma(n) = d + (n-1) * e$$



using the same gap extension penalty



using four different gap extension penalties

Using affine gap model

- We need to remember at position (i, j)
  - the best score *if gap is open*
  - the best score *if gap is not open*

- We need to know the state of previous step to decide whether the current step is (1) to open a new gap or (2) to extend an existing gap
  - three matrices to represent three different states

    - $F(i,j)$: Score of alignment $x_1 \ldots x_i$ to $y_1 \ldots y_j$ if x_i aligns to y_j
    - $G(i,j)$: Score if x_i aligns to a gap after y_j
    - $H(i,j)$: Score if y_j aligns to a gap after x_i

  - one matrix to store the best score of the three states

    - $V(i,j)$: Best score of alignment $x_1 \ldots x_i$ to $y_1 \ldots y_j$

Needleman-Wunsch using affine gaps

1. Initialization.

$$V(i,0) = d + (i-1) * e$$
$$V(0,j) = d + (j-1) * e$$

2. Iteration.

$$F(i,j) = V(i-1,j-1) + s(x_i, y_j)$$

$$G(i,j) = max \begin{cases} V(i-1,j) - d & \text{// opening a new gap} \\ G(i-1,j) - e & \text{// extend an existing gap} \end{cases}$$

$$H(i,j) = max \begin{cases} V(i,j-1) - d & \text{// opening a new gap} \\ H(i,j-1) - e & \text{// extend an existing gap} \end{cases}$$

$$V(i,j) = max\{F(i,j), G(i,j), H(i,j)\}$$

3. Termination.

$V(i,j)$ has the best alignment.

Amino Acid Similarity and Typical Objective Function

```
  X         220         230         240         250              X
  F--SGGNTHIYMNHVEQCKEILRREPKELCELVISGLPYKFRYLSTKE-QLK-Y
  |   :  |::|||:||:|   |   | |||: : :| |       |    :::::  |:: |
GDFIHTLGDAHIYLNHIEPLKIQLQREPRPFPKLRILRKVEKIDDFKAEDFQIEGYN
  X          260         270         280         290              X
```

$$Score = \sum_{Region\_Start}^{Region\_End} Similarity\_Weights - \sum_{Region\_start}^{Region\_End} Gap\_Penalties$$

where:

$$Gap\_Penalty = Gap\_Start\_Penalty + (Gap\_Size - 1) * Gap\_Size\_Penalty$$

Needleman-Wunsch Alignment Algorithm Matches and Mismatches

	A	D	C	N	Y	R	Q	C	L	C	R	P	M
A	1												
Y					1								
C			1					1		1			
Y					1								
N				1									
R						1					1		
C			1					1		1			
K													
C			1					1		1			
R						1					1		
D		1											
P												1	

Needleman-Wunsch Alignment Algorithm Recursion

	A	D	C	N	Y	R	Q	C	L	C	R	P	M
A	1												
Y					1								
C			1					1		1			
Y					1								
N				1									
R						1	4	3	3	2	2	0	0
C	3	3	4	3	3	3	3	4	3	3	1	0	0
K	3	3	3	3	3	3	3	3	3	2	1	0	0
C	2	2	3	2	2	2	2	3	2	3	1	0	0
R	2	1	1	1	1	2	1	1	1	1	2	0	0
D	1	2	1	1	1	1	1	1	1	1	1	0	0
P	0	0	0	0	0	0	0	0	0	0	0	1	0

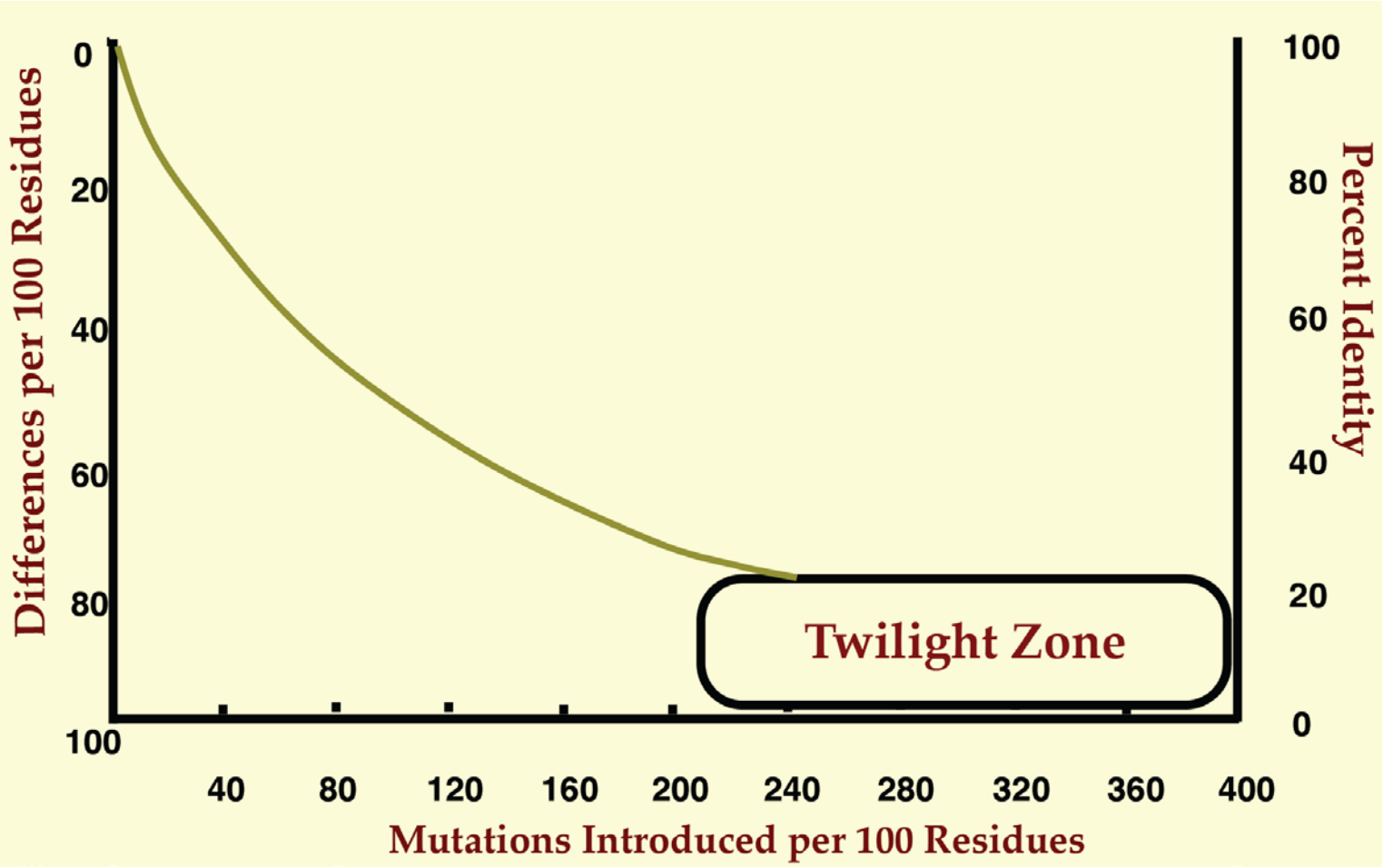
Needleman-Wunsch Alignment Algorithm Maximal Scores

	A	D	C	N	Y	R	Q	C	L	C	R	P	M
A	8	7	6	6	5	4	4	3	3	2	1	0	0
Y	7	7	6	6	6	4	4	3	3	2	1	0	0
C	6	6	7	6	5	4	4	4	3	3	1	0	0
Y	6	6	6	5	6	4	4	3	3	2	1	0	0
N	5	5	5	6	5	4	4	3	3	2	1	0	0
R	4	4	4	4	4	5	4	3	3	2	2	0	0
C	3	3	4	3	3	3	3	4	3	3	1	0	0
K	3	3	3	3	3	3	3	3	3	2	1	0	0
C	2	2	3	2	2	2	2	3	2	3	1	0	0
R	2	1	1	1	1	2	1	1	1	1	2	0	0
D	1	2	1	1	1	1	1	1	1	1	1	0	0
P	0	0	0	0	0	0	0	0	0	0	0	1	0

Needleman-Wunsch Alignment Algorithm Trace Back

		A	D	C	N	Y	R	Q	C	L	C	R	P	M
	A	8	7	6	6	5	4	4	3	3	2	1	0	0
	Y	7	7	6	6	6	4	4	3	3	2	1	0	0
	C	6	6	7	6	5	4	4	4	3	3	1	0	0
	Y	6	6	6	5	6	4	4	3	3	2	1	0	0
	N	5	5	5	6	5	4	4	3	3	2	1	0	0
	R	4	4	4	4	4	5	4	3	3	2	2	0	0
	C	3	3	4	3	3	3	3	4	3	3	1	0	0
	K	3	3	3	3	3	3	3	3	3	2	1	0	0
	C	2	2	3	2	2	2	2	3	2	3	1	0	0
	R	2	1	1	1	1	2	1	1	1	1	2	0	0
	D	1	2	1	1	1	1	1	1	1	1	1	0	0
	P	0	0	0	0	0	0	0	0	0	0	0	1	0

Sequence Similarity vs Evolutionary Distance



Score matrix (substitution matrix)

- **Most commonly used amino acid substitution matrices**
  - PAM (Percent or Point Accepted Mutation)
  - BLOSUM (BLOcks SUbstitution Matrix)

- **PAM**
  - PAM1 is constructed by comparing ***global alignments*** of very closely related sequences (less than 1% divergence) and tallying observed differences
    - used for aligning sequences that have approximately 1% divergence
  - PAM1 is extrapolated for more distant comparisons
    - it is not a linear extrapolation
    - e.g., PAM250 is not for proteins that are 250% divergent

BLOSUM

- BLOSUM1 is built by direct observation of ***local alignments*** between sequences of differing similarity
- Various different versions of the matrix are numbered by how similar the proteins should be
- Default matrix used by BLAST, BLOSUM62, is built from a comparison of sequences that are a minimum of 62% identical
- Choice of which matrix to use in any single case is therefore largely determined by the comparison being made

Dayhoff's Acceptable Point Mutations(PAMs)

		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
		Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
Ala	A																				
Arg	R	30																			
Asn	N	109	17																		
Asp	D	154	0	532																	
Cys	C	33	10	0	0																
Gln	Q	93	120	50	76	0															
Glu	E	266	0	94	831	0	422														
Gly	G	579	10	156	162	10	30	112													
His	H	21	103	226	43	10	243	23	10												
Ile	I	66	30	36	13	17	8	35	0	3											
Leu	L	95	17	37	0	0	75	15	17	40	253										
Lys	K	57	477	322	85	0	147	104	60	23	43	39									
Met	M	29	17	0	0	0	20	7	7	0	57	207	90								
Phe	F	20	7	7	0	0	0	0	17	20	90	167	0	17							
Pro	P	345	67	27	10	10	93	40	49	50	7	43	43	4	7						
Ser	S	772	137	432	98	117	47	86	450	26	20	32	168	20	40	269					
Thr	T	590	20	169	57	10	37	31	50	14	129	52	200	28	10	73	696				
Trp	W	0	27	3	0	0	0	0	0	3	0	13	0	0	10	0	17	0			
Tyr	Y	20	3	36	0	30	0	10	0	40	13	23	10	0	260	0	22	23	6		
Val	V	365	20	13	17	33	27	37	97	30	661	303	17	77	10	50	43	186	0	17	

Dayhoff's PAM 250 Matrix (Log-Odds Form)

		A Ala	R Arg	N Asn	D Asp	C Cys	Q Gln	E Glu	G Gly	H His	I Ile	L Leu	K Lys	M Met	F Phe	P Pro	S Ser	T Thr	W Trp	Y Tyr	V Val
A	Ala	.18																			
R	Arg	-.15	.61																		
N	Asn	.02	0	.20																	
D	Asp	.03	-.13	.21	.39																
C	Cys	-.20	-.36	-.36	-.51	1.19															
Q	Gln	-.04	.13	.08	.16	-.54	.40														
E	Glu	.03	-.11	.14	.34	-.53	.25	.38													
G	Gly	.13	-.26	.03	.06	-.34	-.53	.25	.38												
H	His	-.14	.16	.16	.07	-.34	.29	.07	-.21	.65											
I	Ile	-.05	-.20	-.18	-.24	-.23	-.20	-.20	-.26	-.24	.45										
L	Leu	-.19	-.30	-.29	-.40	-.60	-.18	-.34	-.41	-.21	.24	.59									
K	Lys	-.12	.34	.10	.01	-.54	.07	-.01	-.17	0	-.19	-.29	.47								
M	Met	-.11	-.04	-.17	-.26	-.52	-.10	-.21	-.28	-.21	.22	.37	.04	.64							
F	Phe	-.35	-.45	-.35	-.56	-.43	-.47	-.54	-.48	-.18	.10	.18	-.53	.02	.91						
P	Pro	.11	-.02	-.05	-.10	-.28	.02	-.06	-.05	-.02	-.20	-.25	-.11	-.21	-.46	.59					
S	Ser	.11	-.03	.07	.03	0	-.05	0	.11	-.08	-.14	-.28	-.02	-.16	-.32	.09	.16				
T	Thr	.12	-.09	.04	-.01	-.22	-.08	-.04	0	-.13	.01	-.17	0	-.06	-.31	.03	.13	.26			
W	Trp	-.58	.22	-.42	-.68	-.78	-.48	-.70	-.70	-.28	-.51	-.18	-.35	-.42	.04	-.56	-.25	-.52	1.73		
Y	Tyr	-.35	-.42	-.21	-.43	.03	-.40	-.43	-.52	-.01	-.09	-.09	-.44	-.24	.70	-.49	-.28	-.27	-.02	1.01	
V	Val	.02	-.25	-.17	-.21	-.19	-.19	-.18	-.14	-.22	.37	.19	-.24	.18	-.12	-.12	-.10	.03	-.62	-.25	.43

Dayhoff's PAM 250 Matrix (1978)

Cys	C	12																		
Ser	S	0	2																	
Thr	T	-2	1	3																
Pro	P	-3	1	0	6															
Ala	A	-2	1	1	1	2														
Gly	G	-3	1	0	-1	1	5													
Asn	N	-4	1	0	-1	0	0	2												
Asp	D	-5	0	0	-1	0	1	2	4											
Glu	E	-5	0	0	-1	0	0	1	3	4										
Gln	Q	-5	-1	-1	0	0	-1	1	2	2	4									
His	H	-3	-1	-1	0	-1	-2	2	1	1	3	6								
Arg	R	-4	0	-1	0	-2	-3	0	-1	-1	1	2	6							
Lys	K	-5	0	0	-1	-1	-2	1	0	0	1	0	3	5						
Met	M	-5	-2	-1	-2	-1	-3	-2	-3	-2	-1	-2	0	0	6					
Ile	I	-2	-1	0	-2	-1	-3	-2	-2	-2	-2	-2	-2	-2	2	5				
Leu	L	-6	-3	-2	-3	-2	-4	-3	-4	-3	-2	-2	-3	-3	4	2	6			
Val	V	-2	-1	0	-1	0	-1	-2	-2	-2	-2	-2	-2	-2	2	4	2	4		
Phe	F	-4	-3	-3	-5	-4	-5	-4	-6	-5	-5	-2	-4	-5	0	1	2	-1	9	
Tyr	Y	0	-3	-3	-5	-3	-5	-2	-4	-4	-4	0	-4	-4	-2	-1	-1	-2	7	10
Trp	W	-8	-2	-5	-6	-6	-7	-4	-7	-7	-5	-3	2	-3	-4	-5	-2	-6	0	0	17
		C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W

The log-odds values have been scaled and rounded to the nearest integer for purposes of computational efficiency

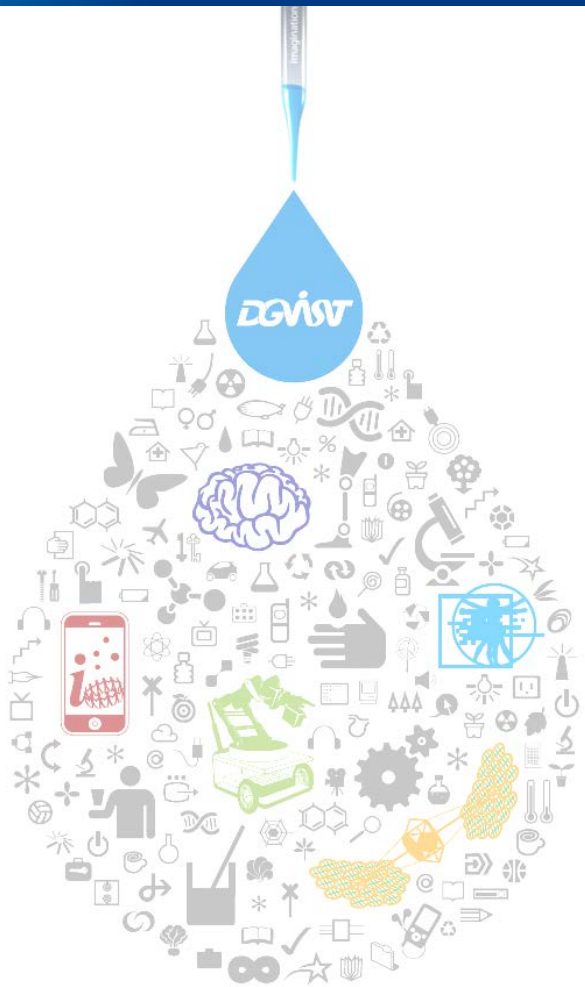
BLOSUM62

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	
C	9																				C
S	-1	4																			S
T	-1	1	5																		T
P	-3	-1	-1	7																	P
A	0	1	0	-1	4																A
G	-3	0	-2	-2	0	6															G
N	-3	1	0	-2	-2	0	6														N
D	-3	0	-1	-1	-2	-1	1	6													D
E	-4	0	-1	-1	-1	-2	0	2	5												E
Q	-3	0	-1	-1	-1	-2	0	0	2	5											Q
H	-3	-1	-2	-2	-2	-2	1	-1	0	0	8										H
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	5									R
K	-3	0	-1	-1	-1	-2	0	-1	1	1	-1	2	5								K
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	5							M
I	-1	-2	-1	-3	-1	-4	-3	-3	-3	-3	-3	-3	-3	1	4						I
L	-1	-2	-1	-3	-1	-4	-3	-4	-3	-2	-3	-2	-2	2	2	4					L
V	-1	-2	0	-2	0	-3	-3	-3	-2	-2	-3	-3	-2	1	3	1	4				V
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	6			F
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	2	-2	-2	-1	-1	-1	-1	3	7		Y
W	-2	-3	-2	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	11	W
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	

InfoLab

Pairwise Sequence Alignment

Heuristic methods



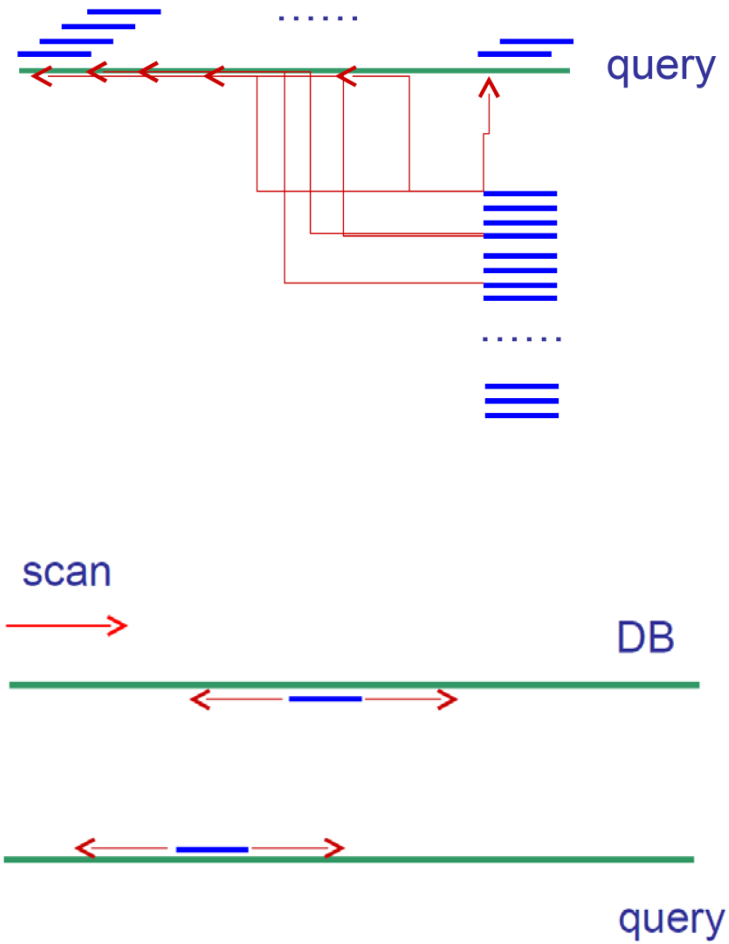
DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology

Original BLAST Algorithm

- Basic Local Alignment Search Tool
- Indexes words in database
- Calculates “neighborhood” of each word in query using BLOSUM matrix and probability threshold
- Looks up all words and neighbors from query in database index to find High-scoring Segment Pairs (HSPs)
- Extends High-scoring Segment Pairs (HSPs) left and right to maximal length
- Finds Maximal Segment Pairs (MSPs) between query and database
- Does not permit gaps in alignments

BLAST: Indexing-based local alignment

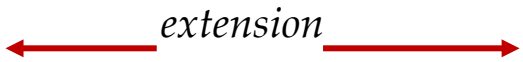
- **Dictionary**
  - keep all words of length *k* (~10)
  - initiate alignment between words of alignment score > T (typically T=k)
- **Alignment**
  - extend the candidate sequence without gaps until the score falls below the statistical threshold
- **Output**
  - output all local alignments with score > statistical threshold



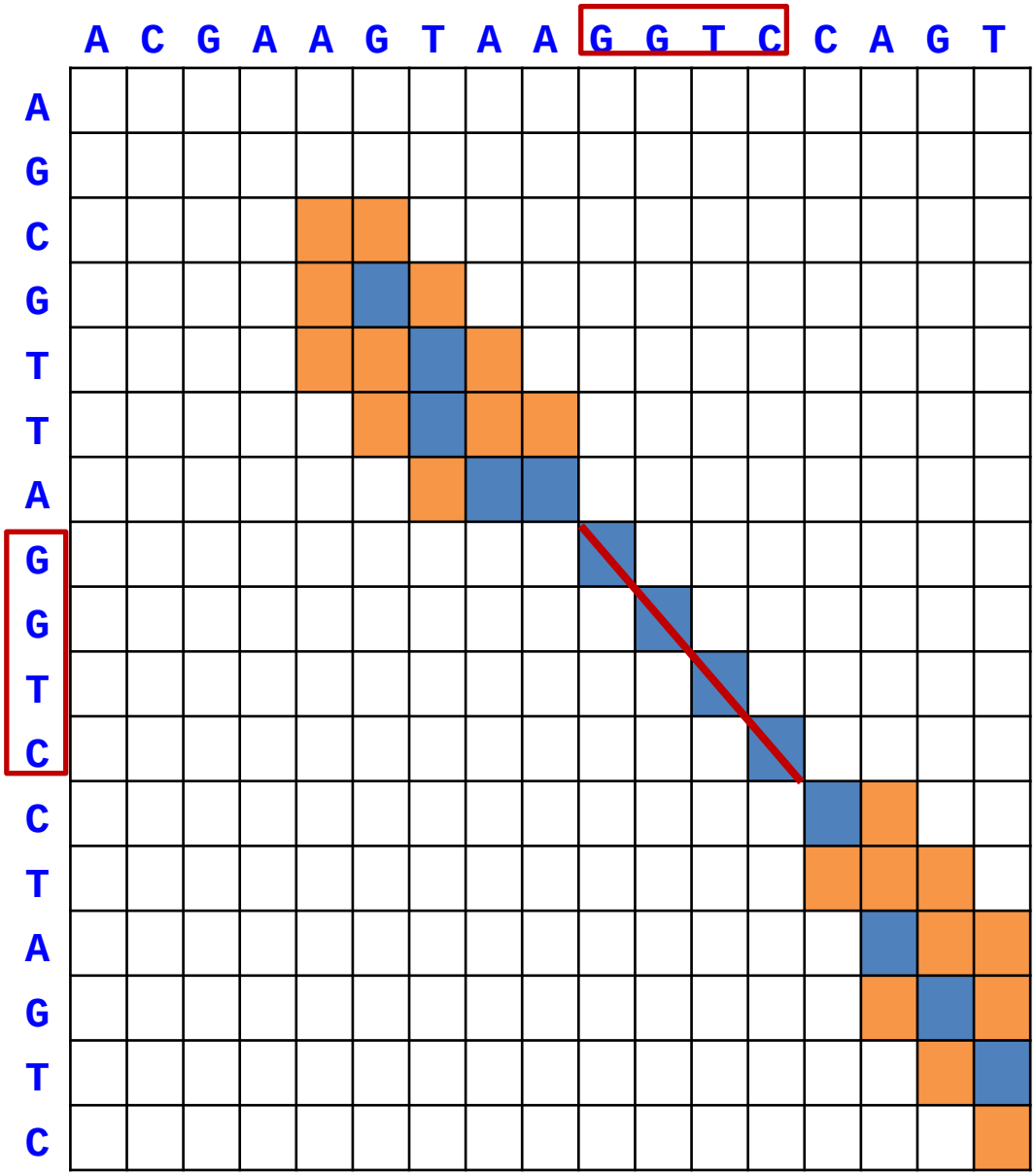
Improvement of Extensions

- **Gapped extensions until threshold**
  - extensions with gaps until the score < C below best score so far

- **Output**



GTAAGGTCC-AGT
GTTAGGTCCTAGT



Sensitivity-Speed Tradeoff

- longer and longer *k* removes more and more potential homologies
  - pros: improving speed
  - cons: decreasing sensitivity and ability to detect meaningful homologies between the two sequences

Sens.

Speed

# of spurious matches

Table 3. Sensitivity and Specificity of Single Perfect Nucleotide K-mer Matches as a Search Criterion

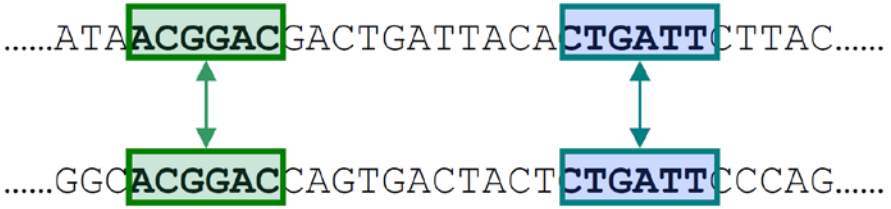
		7	8	9	10	11	12	13	14
A.	81%	0.974	0.915	0.833	0.726	0.607	0.486	0.373	0.314
	83%	0.988	0.953	0.897	0.815	0.711	0.595	0.478	0.415
	85%	0.996	0.978	0.945	0.888	0.808	0.707	0.594	0.532
	87%	0.999	0.992	0.975	0.942	0.888	0.811	0.714	0.659
	89%	1.000	0.998	0.991	0.976	0.946	0.897	0.824	0.782
	91%	1.000	1.000	0.998	0.993	0.981	0.956	0.912	0.886
	93%	1.000	1.000	1.000	0.999	0.995	0.987	0.968	0.957
	95%	1.000	1.000	1.000	1.000	0.999	0.998	0.994	0.991
	97%	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.999
B.	K	7	8	9	10	11	12	13	14
	F	1.3e+07	2.9e+06	635783	143051	32512	7451	1719	399

(*A*) Columns are for K sizes of 7–14. Rows represent various percentage identities between the homologous sequences. The table entries show the fraction of homologies detected as calculated from equation 3 assuming a homologous region of 100 bases. The larger the value of K, the fewer homologies are detected.
(*B*) K represents the size of the perfect match. F shows how many perfect matches of this size expected to occur by chance according to equation 4 in a genome of 3 billion bases using a query of 500 bases.

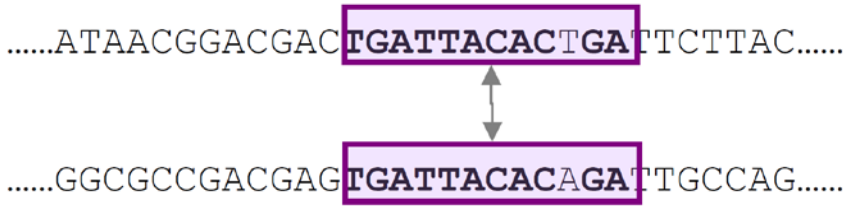
Kent WJ, Genome Research 2002

Improving Sensitivity-Speed

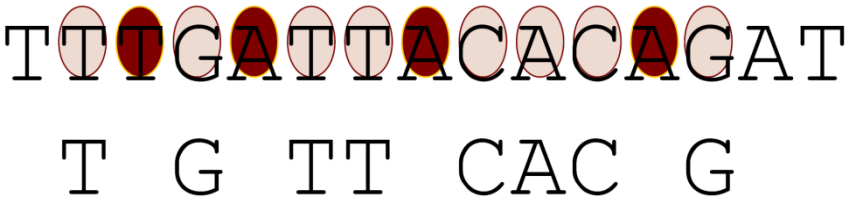
- Split a single k-mer into a pair of shorter k-mers



- Using inexact words (not just exact matches)



- Patterns – nonconsecutive positions



Measured improvement

Table 7. Sensitivity and Specificity of Multiple (2 and 3) Perfect Nucleotide K-mer Matches as a Search Criterion

	2,8	2,9	2,10	2,11	2,12	3,8	3,9	3,10	3,11	3,12
A. 81%	0.681	0.508	0.348	0.220	0.129	0.389	0.221	0.112	0.051	0.021
83%	0.790	0.638	0.475	0.326	0.208	0.529	0.339	0.193	0.099	0.045
85%	0.879	0.762	0.615	0.460	0.318	0.676	0.487	0.313	0.180	0.093
87%	0.942	0.866	0.752	0.611	0.461	0.809	0.649	0.470	0.305	0.177
89%	0.978	0.940	0.868	0.761	0.625	0.910	0.801	0.648	0.476	0.314
91%	0.994	0.980	0.947	0.884	0.787	0.969	0.914	0.815	0.673	0.505
93%	0.999	0.996	0.986	0.962	0.912	0.993	0.976	0.933	0.851	0.722
95%	1.000	1.000	0.998	0.993	0.979	0.999	0.997	0.987	0.961	0.902
97%	1.000	1.000	1.000	1.000	0.999	1.000	1.000	0.999	0.997	0.987

B. N,K	2,8	2,9	2,10	2,11	2,12	3,8	3,9	3,10	3,11	3,12
F	524	27	1.4	0.1	0.0	0.1	0.0	0.0	0.0	0.0

(*A*) Columns are for N sizes of 2 and 3 and K sizes of 8–12. Rows represent various percentage identities between the homologous sequences. The table entries show the fraction of homologies detected as calculated by equation 10. (*B*) N and K represent the number and size of the near-perfect matches, respectively. F shows how many perfect clustered matches expected to occur by chance according to equation 14 in a translated genome of 3 billion bases using a query of 167 amino acids.

Table 5. Sensitivity and Specificity of Single Near-Perfect (One Mismatch Allowed) Nucleotide K-mer Matches as a Search Criterion

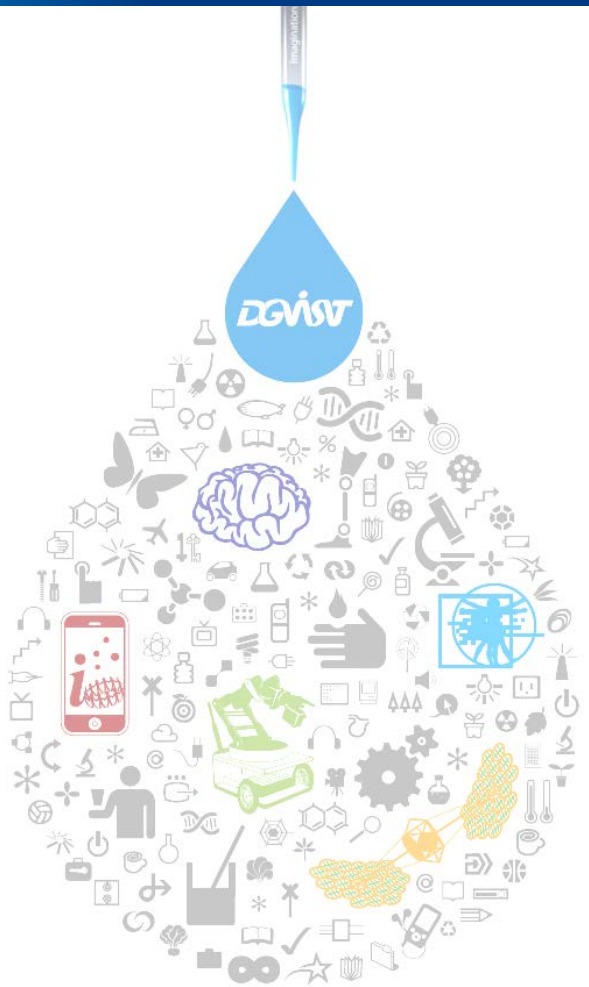
	12	13	14	15	16	17	18	19	20	21	22
A. 81%	0.945	0.880	0.831	0.721	0.657	0.526	0.465	0.408	0.356	0.255	0.218
83%	0.975	0.936	0.904	0.820	0.770	0.649	0.591	0.535	0.480	0.361	0.318
85%	0.991	0.971	0.954	0.900	0.865	0.767	0.719	0.669	0.619	0.490	0.445
87%	0.997	0.990	0.983	0.954	0.935	0.867	0.833	0.796	0.757	0.634	0.591
89%	1.000	0.997	0.995	0.984	0.976	0.939	0.920	0.897	0.872	0.775	0.741
91%	1.000	1.000	0.999	0.996	0.994	0.979	0.971	0.962	0.950	0.890	0.869
93%	1.000	1.000	1.000	0.999	0.999	0.996	0.994	0.991	0.988	0.963	0.954
95%	1.000	1.000	1.000	1.000	1.000	1.000	0.999	0.999	0.999	0.994	0.992
97%	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

B. K	12	13	14	15	16	17	18	19	20	21	22
F	275671	68775	17163	4284	1070	267	67	17	4.2	1.0	0.3

(*A*) Columns are for K sizes of 12–22. Rows represent various percentage identities between the homologous sequences. The table entries show the fraction of homologies detected as calculated by equation 6 assuming a homologous region of 100 bases. (*B*) K represents the size of the near-perfect match. F shows how many perfect matches of this size expected to occur by chance according to equation 7 in a genome of 3 billion bases using a query of 500 bases.

InfoLab

Thank you!



DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology