Topic 4

Pairwise Sequence Alignments

Alignment Concepts

Sequence Alignment

- Suppose we have sequences
  - agct
  - accaat
- The following is an alignment
  - a g c - - t
  - a c c a a t

Gapped Alignments

- Biological sequences
  - Different lengths
  - Regions of insertions and deletions
- Notion of gaps (denoted by '-')

```
A L I G N M E N T
|   | |   | | | |
A - I G A M E N T
```

Possible Residue Alignments

- Match
- Mismatch (substitution or mutation)
- Insertion/Deletion (INDELS – gaps)

Sequence Alignment

- Question: Are two sequences related?
- Compare the two sequences, see if they are similar

Sequence Alignment

- Goal: to deduce whether sequences are related
- Cornerstone of bioinformatics
- Two major categories:
  - Pairwise alignments
  - Multiple alignments

Pairwise Sequence Alignment

- Why might an pairwise alignment be significant?
  - An alignment is *a mapping from one sequence to another, identifying elements that are likely to have arisen from a common ancestor*
  - A good alignment is an indication of homology
- Alignments are NOT exact matches. We will need a method to find good alignments in a database...

Fundamental Operation

- Pairwise sequence alignment is the most fundamental operation of bioinformatics
  - It is used to decide if two proteins (or genes) are related structurally or functionally
  - It is used to identify domains or motifs that are shared between proteins
  - It is used in the analysis of genomes

Relation of sequences

- **Homologs**: similar sequences in 2 different organisms derived from a common ancestor sequence.
- **Orthologs**: Similar sequences in 2 different organisms that have arisen due to a speciation event. Functionality Retained.

Similarity vs. Homology

- *Similarity* is a measure of the quality of alignment between two sequences. High similarity is evidence for homology.
- *Homology* is an evolutionary relationship that either exists or does not. It cannot be partial.

Relation of sequences

- **Paralogs**: Similar sequences within a single organism that have arisen due to a gene duplication event.
- **Xenologs**: similar sequences that have arisen out of horizontal transfer events (symbiosis, viruses, etc)

Relation of sequences

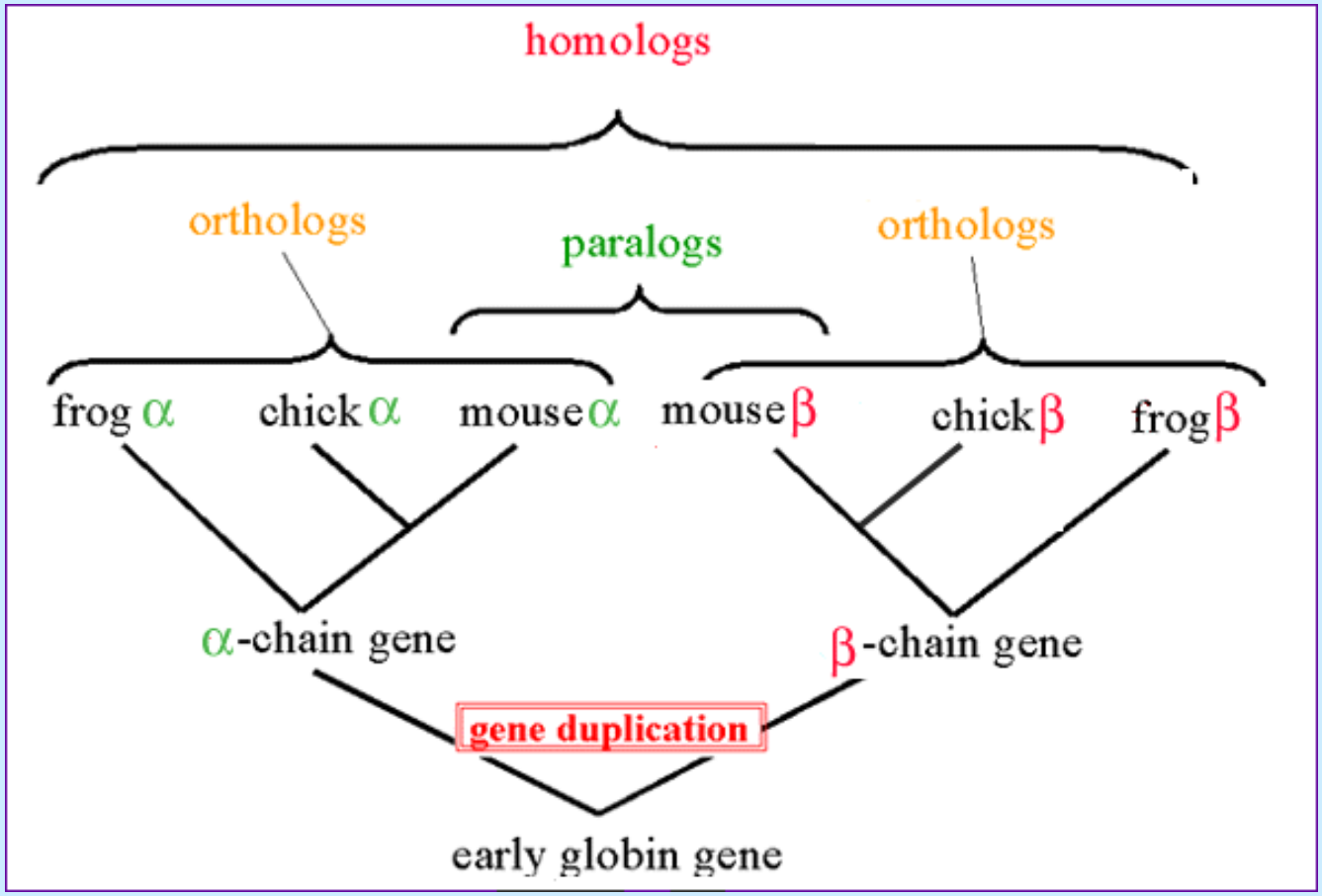
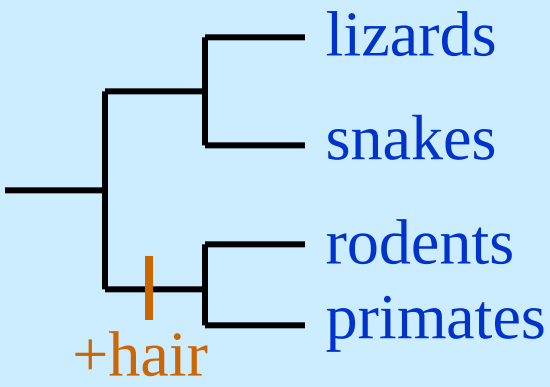


Image Source: http://www.ebi.ac.uk/2can/genomes/Eukaryotes/Info/Orthology.html

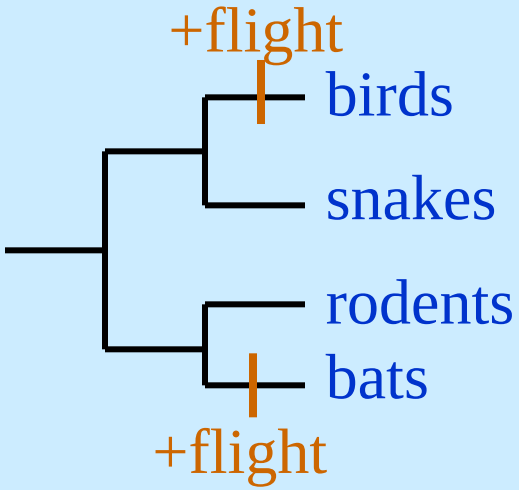
Phylogenetic concepts

Homology: identical character due to shared ancestry (evolutionary *signal*)

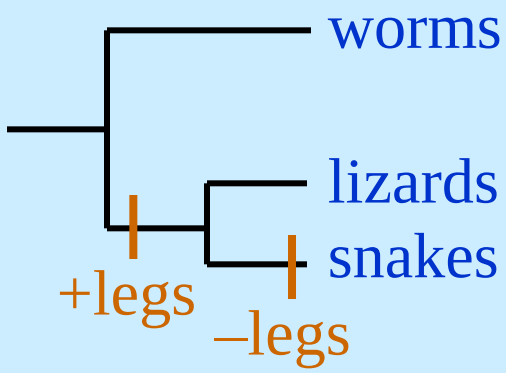
Homoplasy: identical character due to evolutionary convergence or reversal (evolutionary *noise*)



Homology



Homoplasy (Convergence)



Homoplasy (Reversal)

Tony Weisstein, http://bioquest.org:16080/bedrock/terre_haute_03_04/phylogenetics_1.0.ppt

Biological Sequences

- Similar biological sequences tend to be related
- Information:
  - Functional
  - Structural
  - Evolutionary

Biological sequences

- Common mistake:
  - Sequence similarity is not homology!
- Example: ***pear*** and ***tear***
  - Similar words, different meanings

Alignments

- Which alignment is best?

```
A - C - G G - A C T
|   |   |       | |
A T C G G A T - C T
```

```
A T C G G A T C T
|   | | |     | |
A - C G G - A C T
```

Global vs. Local

- Global alignments
- Local alignments

Global Alignments

- Along their entire length
- Used for highly similar sequences of approximately the same length

Local Alignments

- To find the most similar regions in the two sequences

Global vs. Local Alignment

- Global Alignment

```
--T--CC-C-AGT--TATGT-CAGGGGACACG-A-GCATGCAGA-GAC
  |  || |  ||  | | | |||    || | | |  | ||||   |
AATTGCCGCC-GTCGT-T-TTCAG----CA-GTTATG-T-CAGAT--C
```

- Local Alignment

```
                  tccCAGTTATGTCAGgggacacgagcatgcagagac
                     ||||||||||||

aattgccgccgtcgttttcagCAGTTATGTCAGatc
```

How do we compute similarity?

- Similarity can be defined by counting positions that are identical between two sequences
- Gaps (insertions/deletions) can be important

```
abcdef    abcdef    abcdef
|||  ||   |         | ||||
abceef    acdef     a-cdef
```

Hamming Distance

- Minimum number of letters by which two words differ
- Calculated by summing number of mismatches
- Hamming Distance between PEAR and TEAR is 1

Edit distance

- Two sequences:
  - `ACATTGTGGAT`
  - `ACTTGTAGATG`
- Although the two sequences look different, one can change the first one to the second by three editing operations
  - insert “G” at the end of seq1
  - substitute the 8th letter “G” with “A”
  - delete the 3rd letter “A”

Edit distance

- The edit distance between two sequences is the minimum number of editing operations that need to convert one to the other.
- Editing operations:
  - insertion, deletion, substitution
- d(s, t) = d(t, s)
  - Proof: ……

Alignments

- Which alignment is best?

```
A - C - G G - A C T
|   |   |       | |
A T C G G A T - C T
```

```
A T C G G A T C T
|   | | |     | |
A - C G G - A C T
```

Alignment Scoring Scheme

- Possible scoring scheme:

  match: +2

  mismatch: -1

  indel –2

- Alignment 1: 5 * 2 – 1(1) – 4(2) = 10 – 1 – 8 = 1
- Alignment 2: 6 * 2 – 1(1) – 2 (2) = 12 – 1 – 4 = 7

Affined gap penalty

- For a gap with k consecutive spaces, we use

$$g(k) = a + k*b$$

as the penalty. a is called gap open penalty, b is the gap extension

Gapped Alignment

- Which of the following two alignments is better?
- a b c - - d
  a c c e f d

And

a b - e - d
a c c e f d

An Alignment Score

- The penalty for one long gap is smaller than the penalty for many smaller gaps that add up to the same size.

```
a b c - - d
a c c e f d
4-3 9     6 => 16 - (5 + 2) = 9
```

-

```
a b - e - d
a c c e f d
4-3   5   6 => 12-(5+1)-(5+1)= 0
```

Not all mismatches are the same

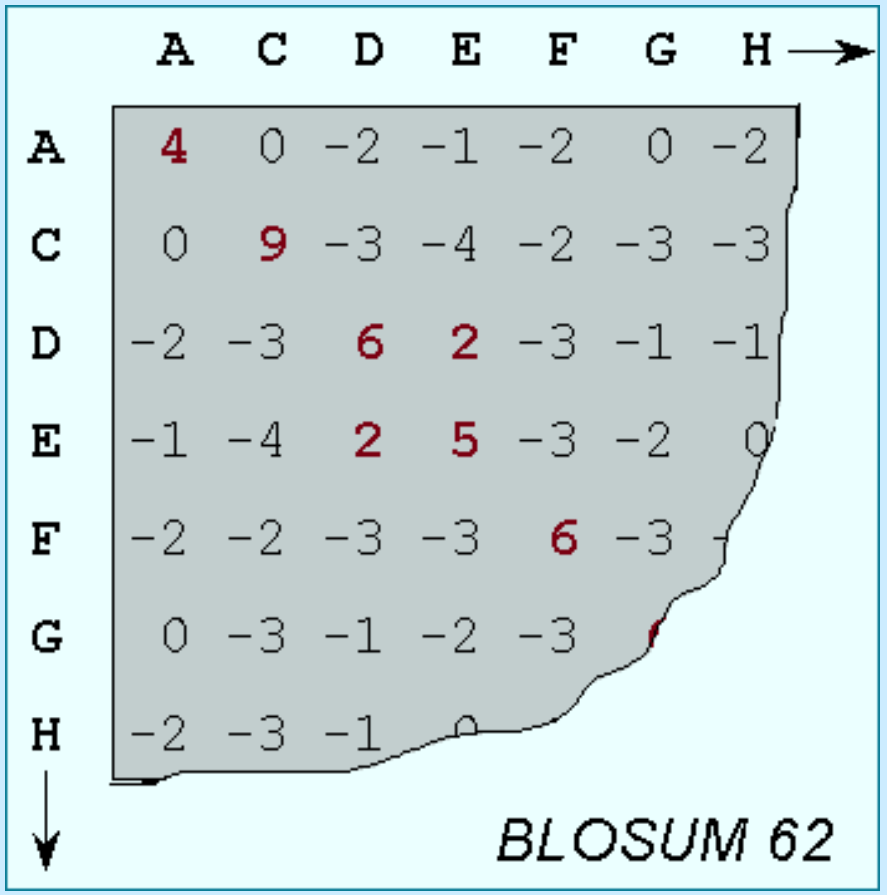
- Some amino acids are more substitutable for each other than others. Serine and threonine are more alike than tryptophan and alanine.
- We can introduce “scores matrix" for handling different substitutions.

Weights of different operation

- Substitution may be easier than indel (insertion/deletion).
  - Give higher cost to indel
- Substitution between different pairs may be different.
- To satisfy d(s,t) = d(t,s)
  - cost(insertion) = cost(deletion)
  - cost(substitute(x,y)) = cost(substitute(y,x))

Scores Matrix

- Match scores are often calculated on the basis of the frequency of particular mutations in very similar sequences.



Alignment v.s. edit distance

- Each edit distance problem can be reduced to a sequence alignment problem.

Finding the optimal alignment

- Given a pair of sequences and a score function, identify the best scoring (optimal) alignment between the sequences.
- Remember, exponential number of possible alignments.

Formal Definition

- Let s, t be two sequences over alphabet Σ. Let f(x,y) be a score scheme. An alignment is two sequences S, T with same length, generated by inserting spaces into s, t, respectively. The score of the alignment is sum(f(S[i],T[i])).
- The sequence alignment is to find the alignment with maximum score.

Alignment Methods

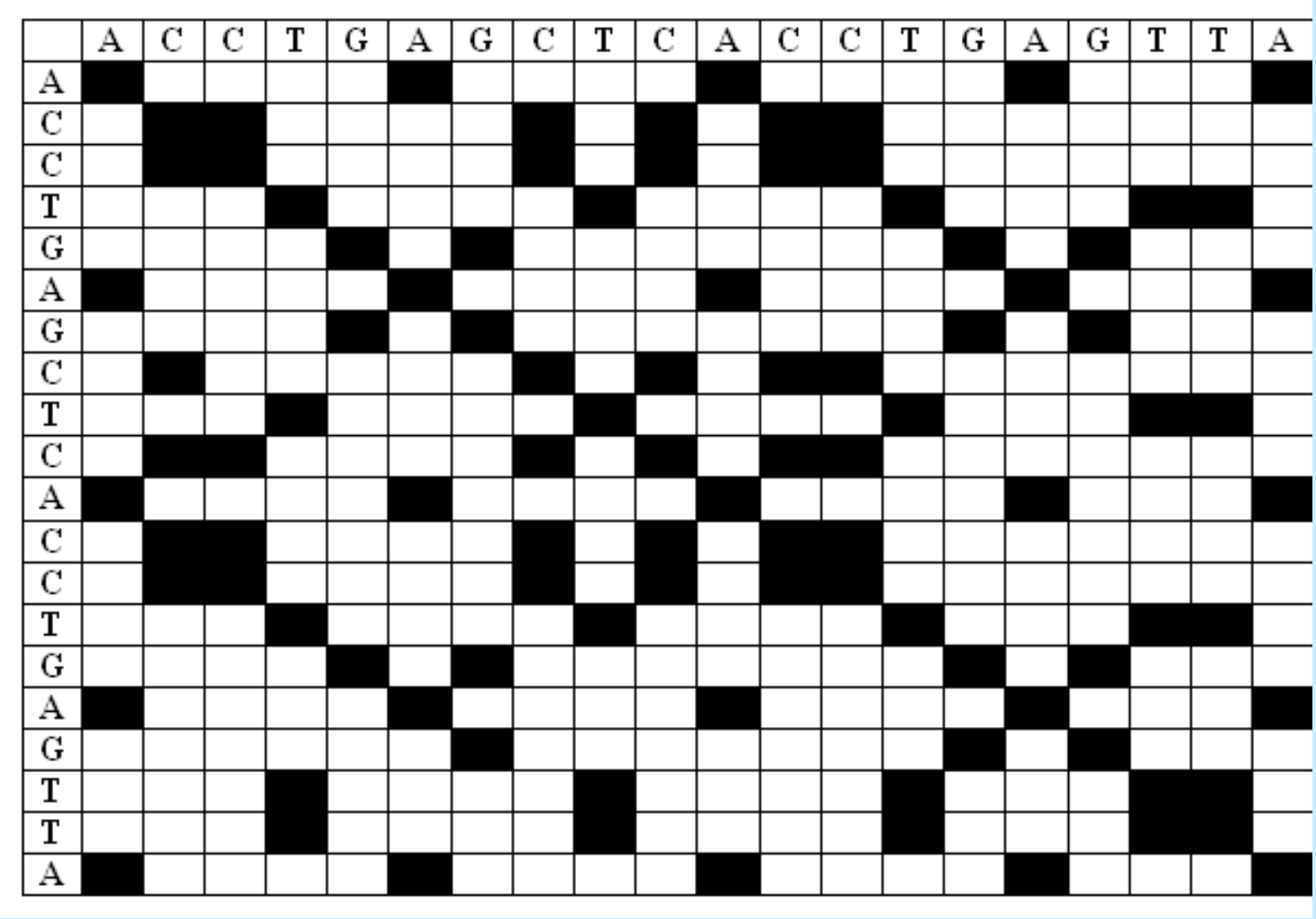
- Visual
- Brute Force
- Dynamic Programming

Visual Alignments (Dot Plots)

Visual Alignments (Dot Plots)

- Matrix
  - Rows: Characters in one sequence
  - Columns: Characters in second sequence
- Filling
  - Loop through each row; if character in row, col match, fill in the cell
  - Continue until all cells have been examined

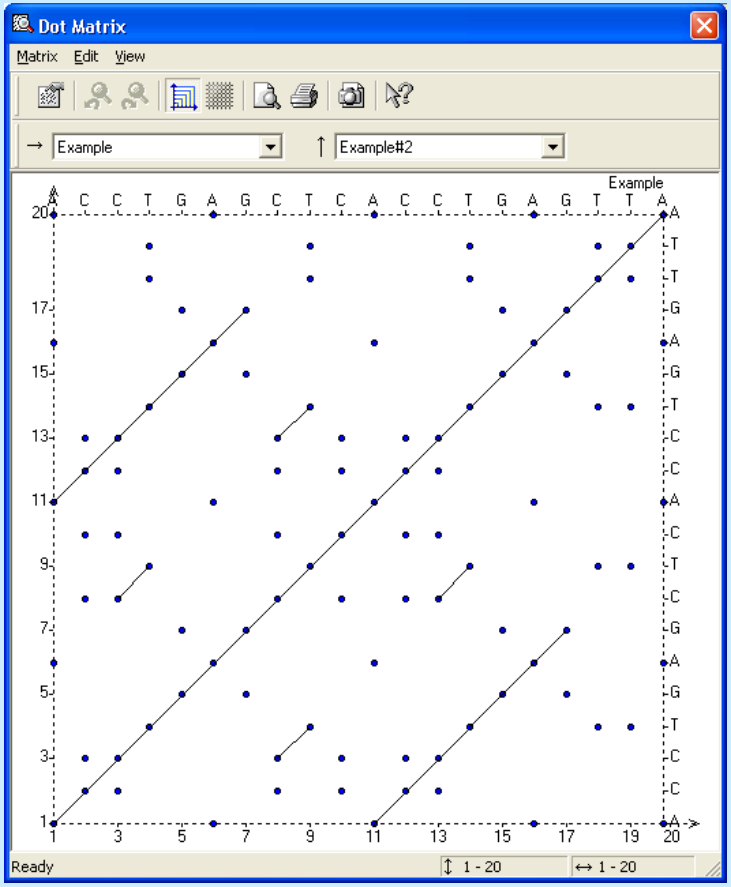
Example Dot Plot

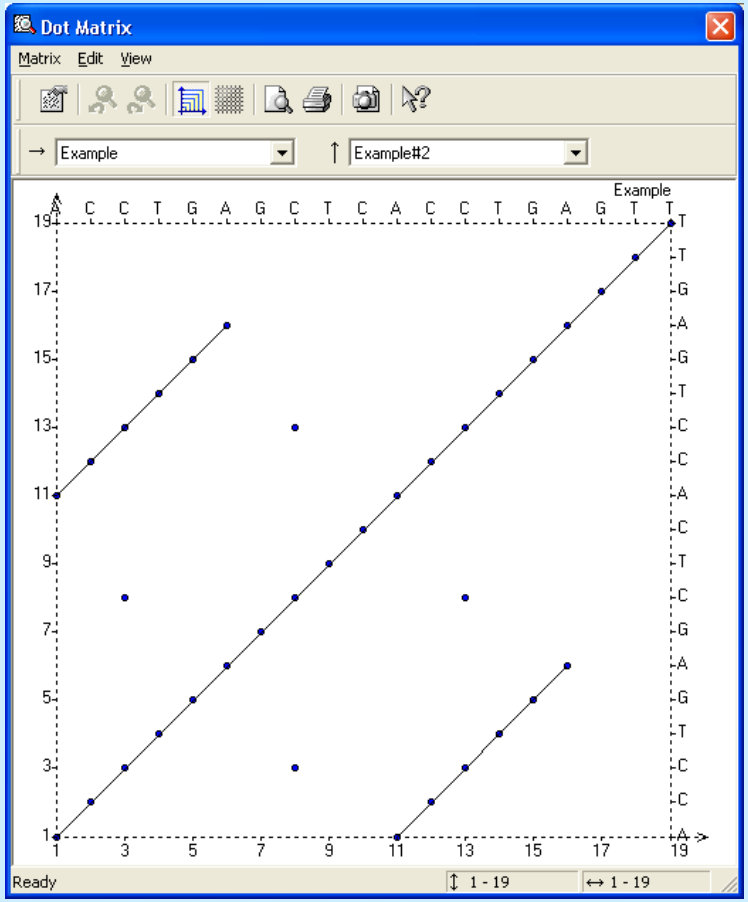


Noise in Dot Plots

- Nucleic Acids (DNA, RNA)
  - 1 out of 4 bases matches at random

Reduction of Dot Plot Noise



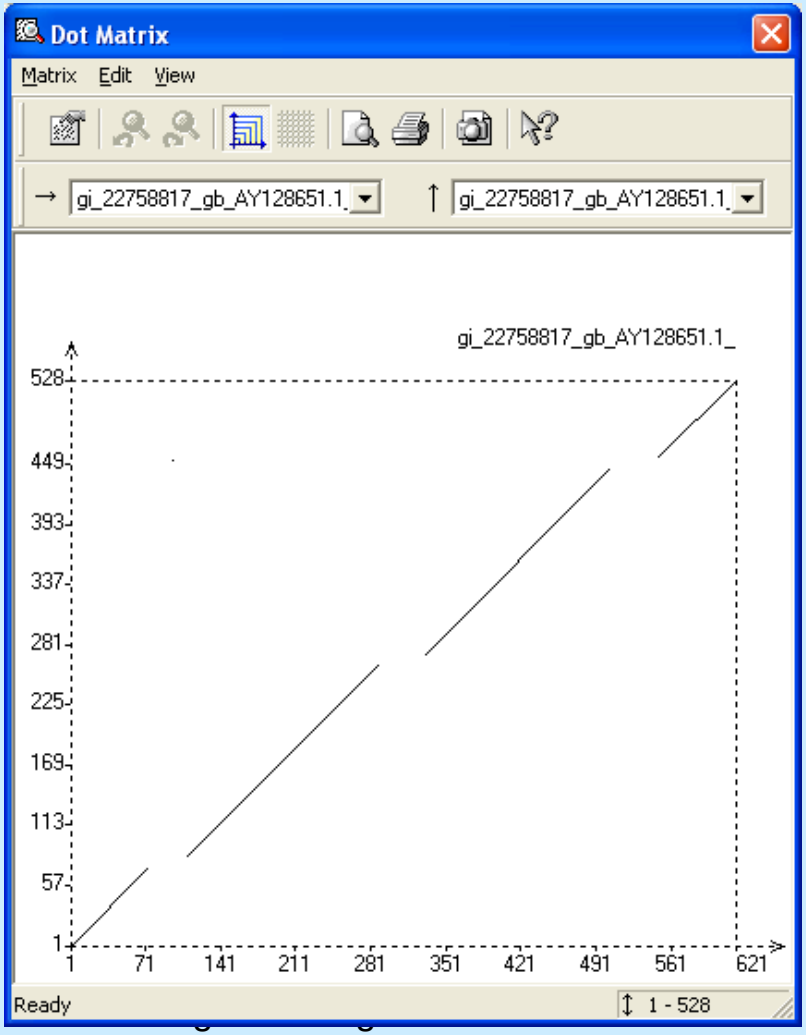


Self alignment of ACCTGAGCTCACCTGAGTTA

Information Inside Dot Plots

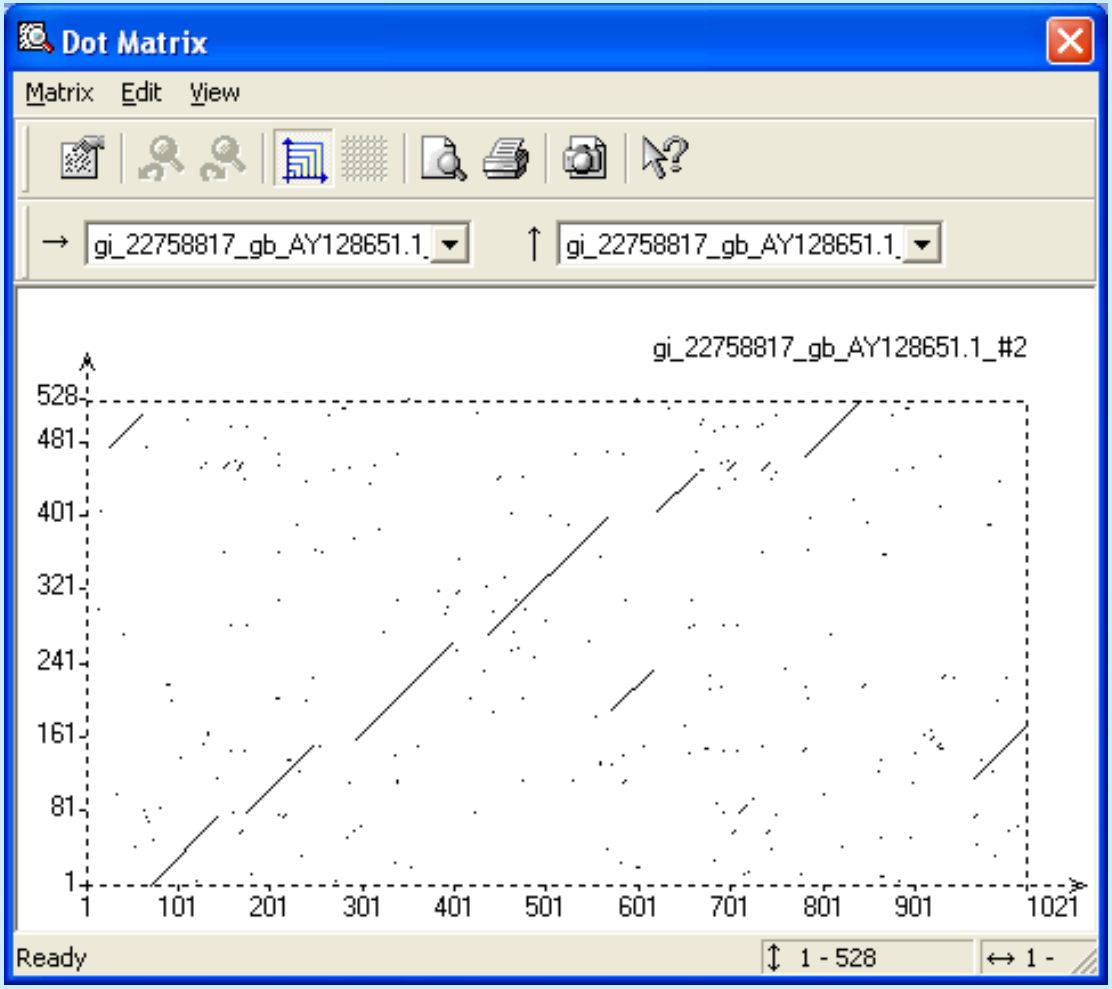
- Regions of similarity: diagonals
- Insertions/deletions
- Repeats and Inverted Repeats
  - Inverted repeats = reverse complement
  - Used to determine folding of RNA molecules

Insertions/Deletions



Available Dot Plot Programs

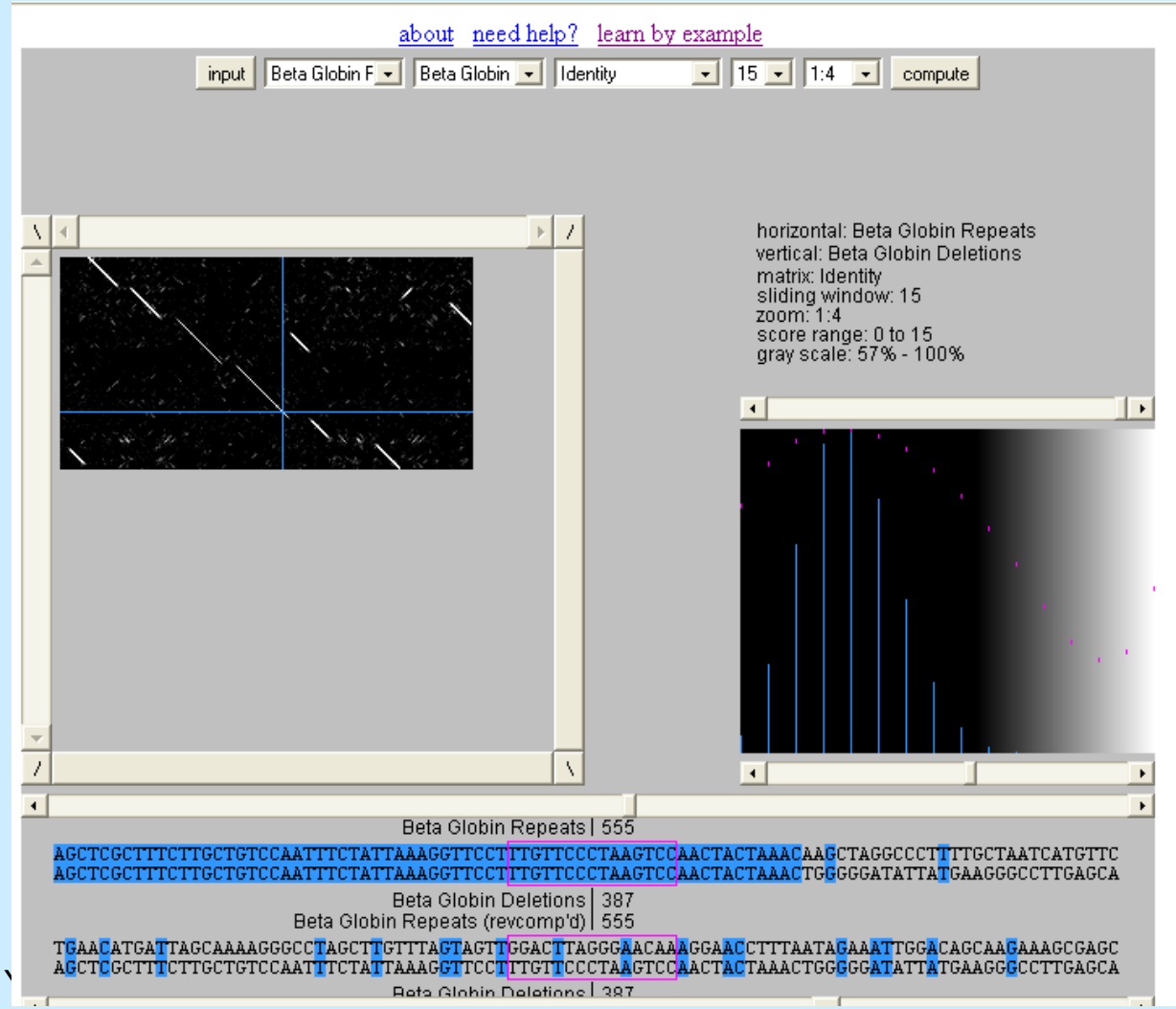
- Vector NTI software package (under AlignX)



Available Dot Plot Programs

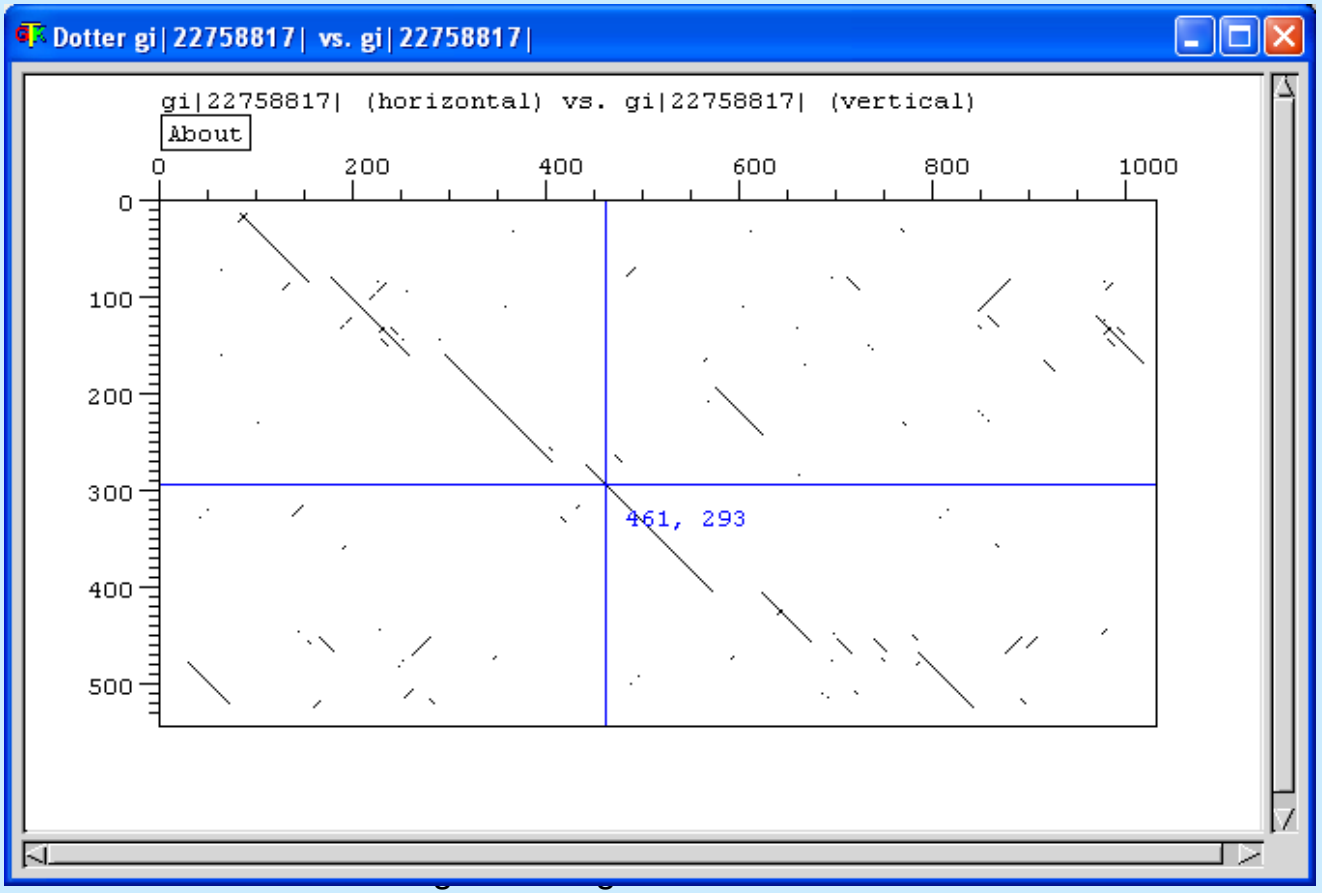
Dotlet (Java Applet)

http://www.isrec.isb-sib.ch/java/dotlet/Dotlet.html



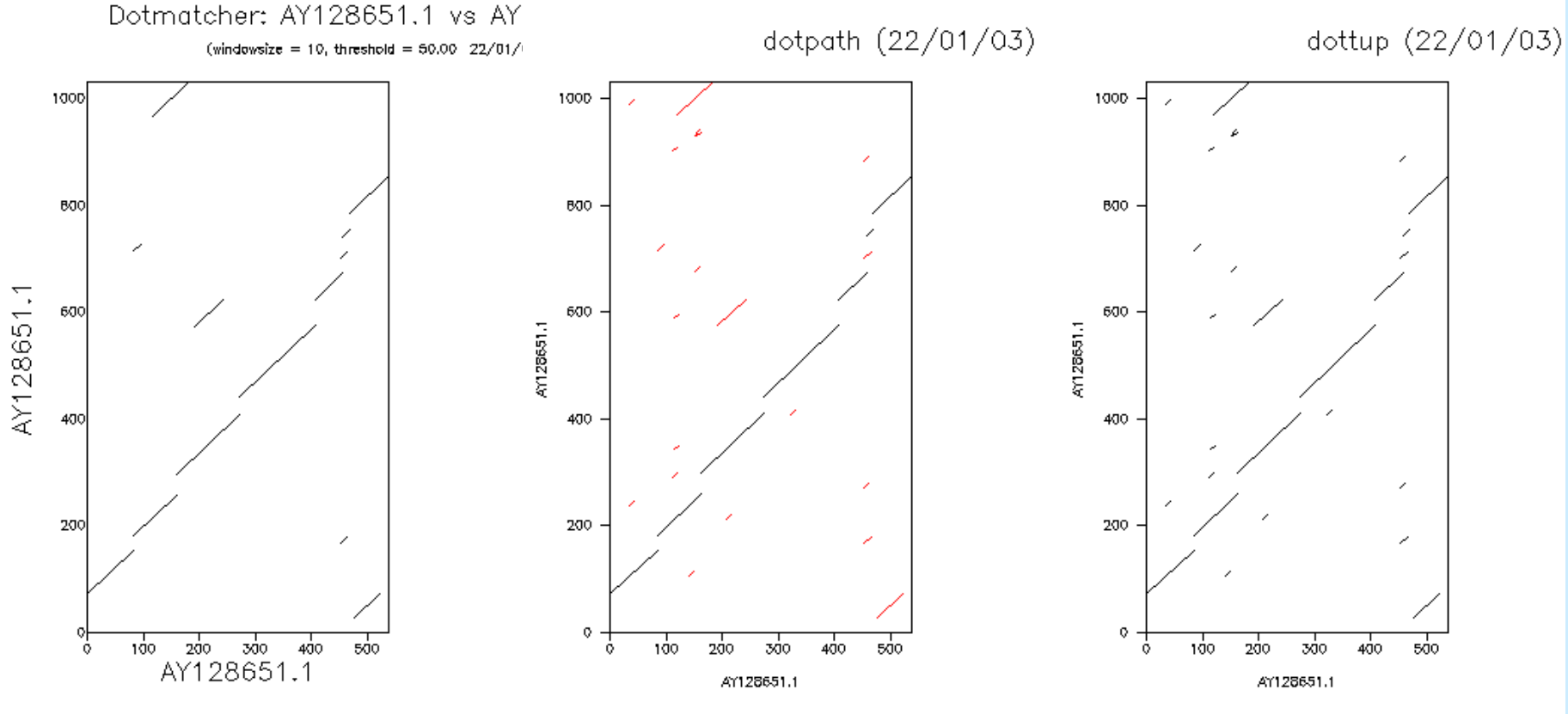
Available Dot Plot Programs

Dotter (http://www.cgr.ki.se/cgr/groups/sonnhammer/Dotter.html)



Available Dot Plot Programs

EMBOSS DotMatcher, DotPath,DotUp



Available Dot Plot Programs

GCG software package:

- Compare http://www.hku.hk/bruhk/gcgdoc/compare.html
- DotPlot+ http://www.hku.hk/bruhk/gcgdoc/dotplot.html
- DNA strider
- PipMaker

Brute Force

Many possible alignments

- There are many possible arrangements to consider:

```
agcgct     agcgct    agcgct
|||  |     |  |||    ||  ||
agc--t     a--gct    ag--ct
```

- This becomes a very large number when we allow mismatches
- Quickly becomes impractical

Matching Score Function

- Not only are there many possible gapped alignments, but introducing too many gaps makes nonsense alignments possible:

```
s--e------qu---en--ce
sometimesquipsentice
```

- Need to distinguish between alignments that occur due to homology, and those that could be expected to be seen just by chance.

Dynamic Programming

Dynamic Programming

- Dynamic programming identifies optimal alignments in time proportional to the sum of the lengths of the sequences

Dynamic Programming

- The name comes from an operations research task, and has nothing to do with writing programs.
- Called Needleman-Wunch Algorithm or Smith-Waterman Algorithm

Dynamic Programming

- Sequence alignment has optimal substructure property
  - Subproblem: alignment of prefixes of two sequences
  - Each subproblem is computed once and stored in a matrix

Dynamic Programming

- Optimal score: built upon optimal alignment computed to that point
- Aligns two sequences beginning at ends, attempting to align all possible pairs of characters

Dynamic Programming

- Guaranteed to provide optimal alignment given:
  - Two sequences
  - Scoring scheme

Steps in DP

- Initialization
- Matrix Fill (scoring)
- Traceback (alignment)

DP Example

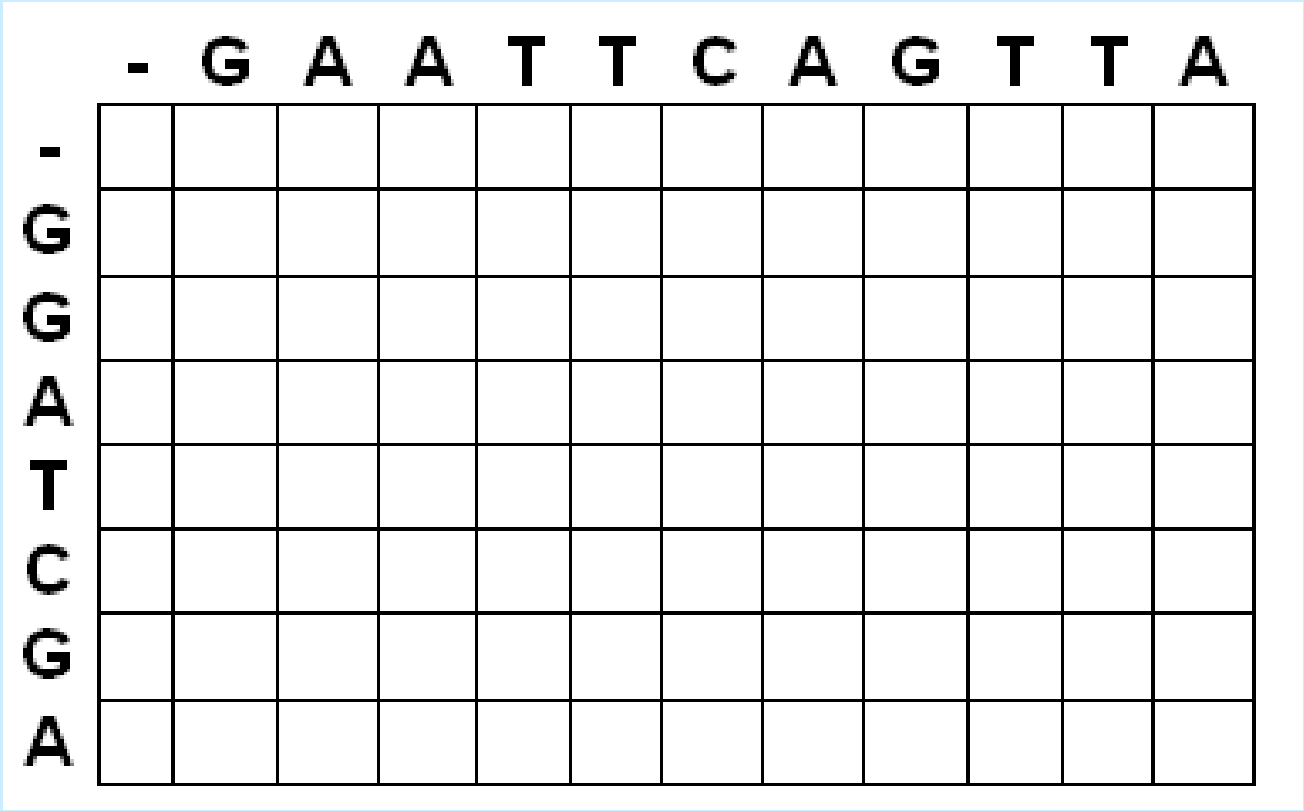
Sequence #1: GAATTCAGTTA; M = 11

Sequence #2: GGATCGA; N = 7

- $s(a_i b_j) = +5$ if $a_i = b_j$ (match score)
- $s(a_i b_j) = -3$ if $a_i \neq b_j$ (mismatch score)
- $w = -4$ (gap penalty)

View of the DP Matrix

- M+1 rows, N+1 columns



Global Alignment (Needleman-Wunsch)

- ***INITIALIZATION***: Set the first row and the first column
- $S_{i,0} = w * i$
- $S_{0,j} = w * j$

Initialized Matrix (Needleman-Wunsch)

		-	G	A	A	T	T	C	A	G	T	T	A
	-	0	-4	-8	-12	-16	-20	-24	-28	-32	-36	-40	-44
G		-4											
G		-8											
A		-12											
T		-16											
C		-20											
G		-24											
A		-28											

Matrix Fill (Global Alignment)

$$S_{i,j} = \text{MAXIMUM}[$$

$$S_{i-1,\,j-1} + s(a_i, b_j) \text{ (match/mismatch in the diagonal)},$$

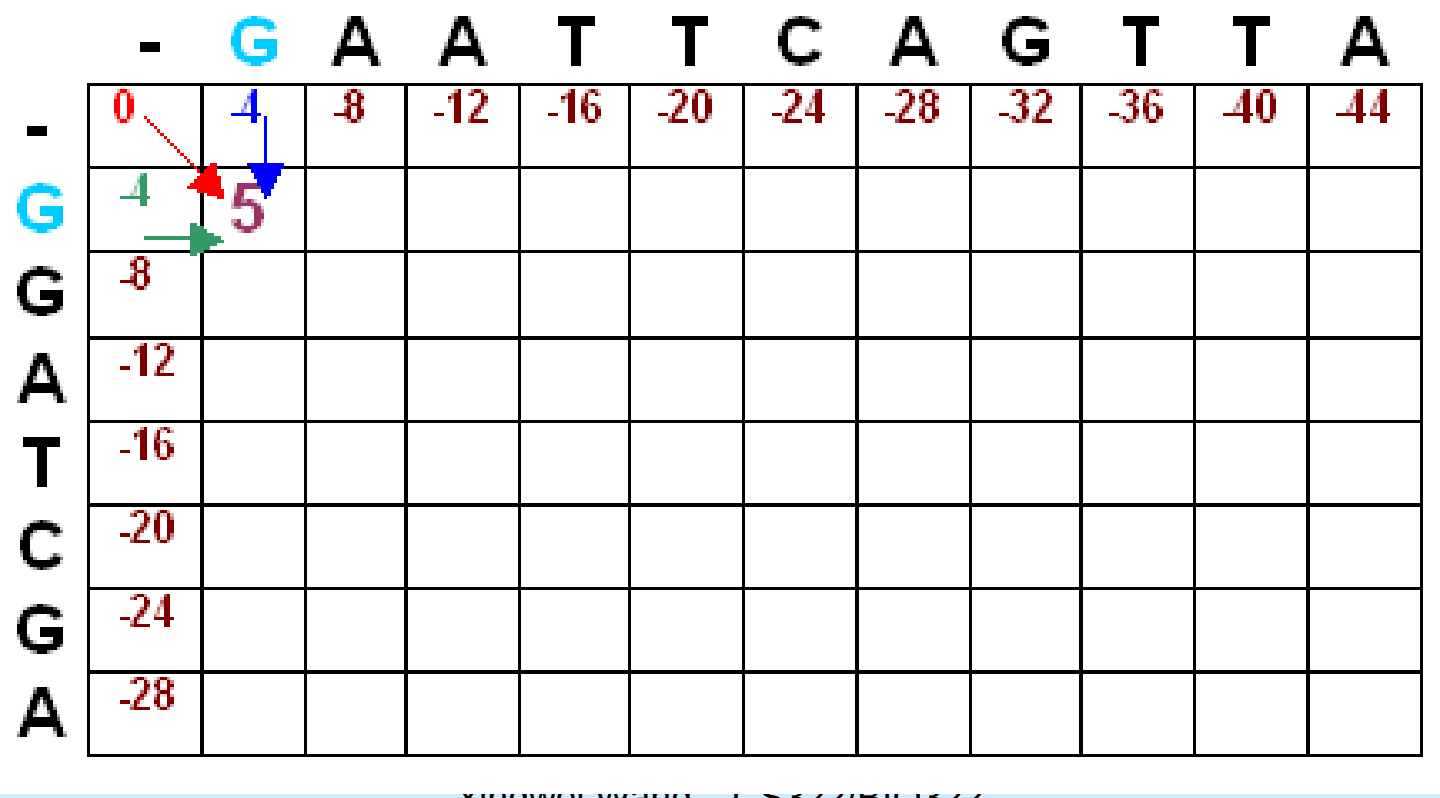
$$S_{i,j-1} + w \text{ (gap in sequence \#1)},$$

$$S_{i-1,j} + w \text{ (gap in sequence \#2)}$$

$$]$$

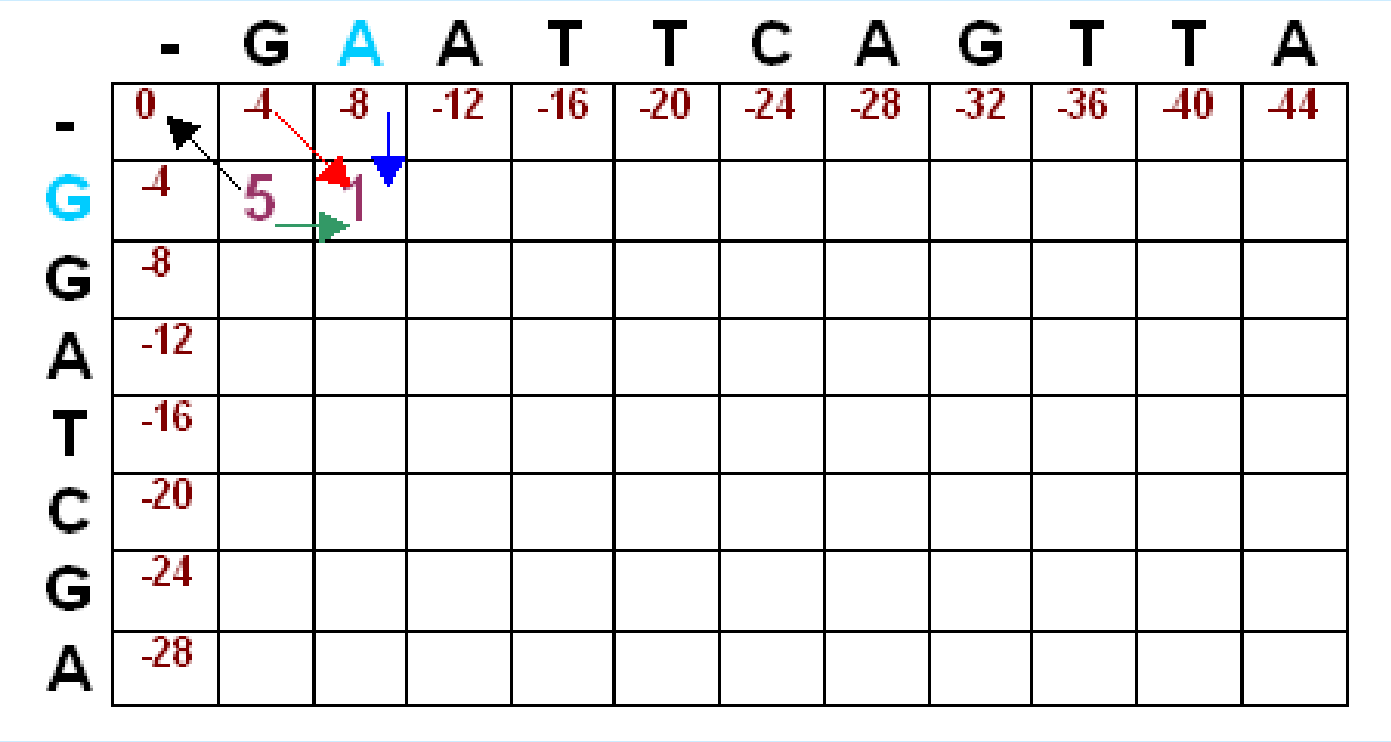
Matrix Fill (Global Alignment)

- $S_{1,1} = MAX[S_{0,0} + 5, S_{1,0} - 4, S_{0,1} - 4] = MAX[5, -8, -8]$



Matrix Fill (Global Alignment)

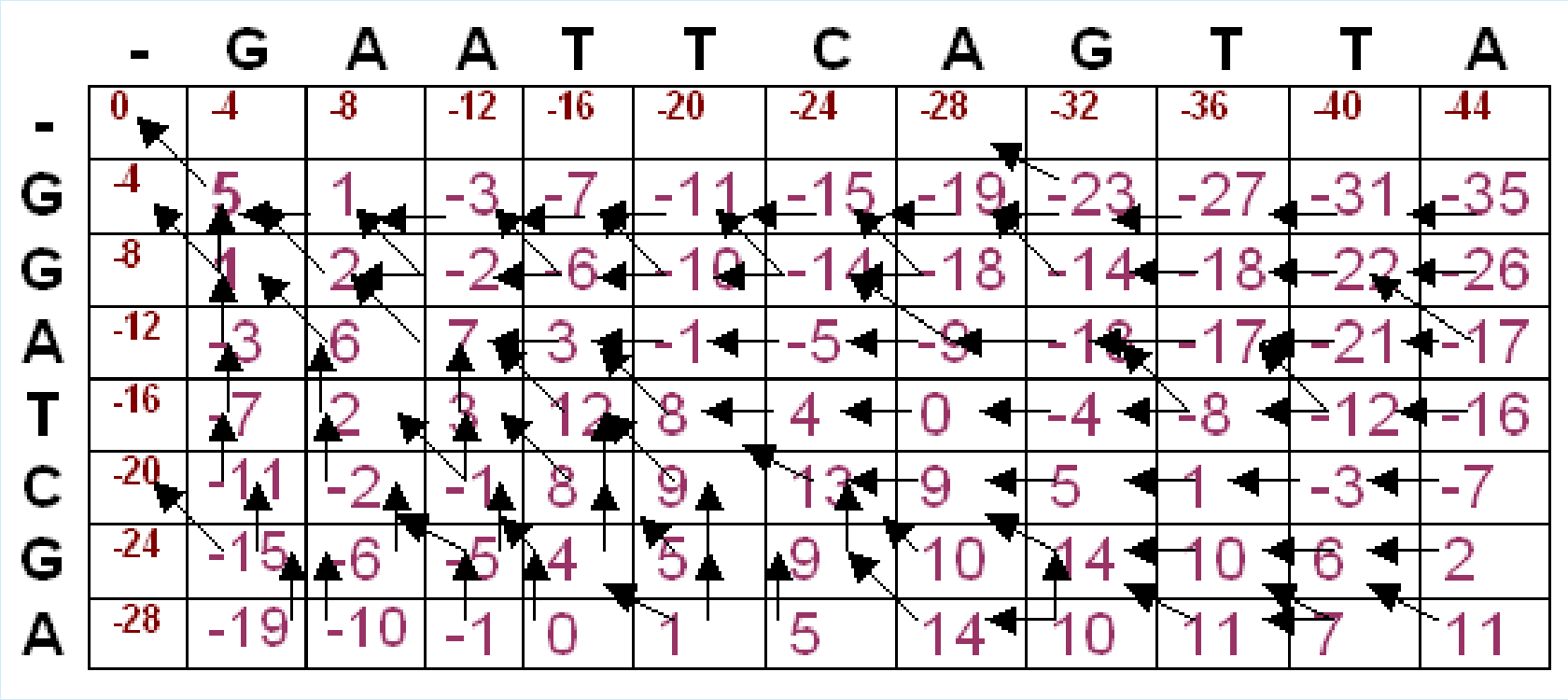
- $S_{1,2} = MAX[S_{0,1} - 3, S_{1,1} - 4, S_{0,2} - 4] = MAX[-4 - 3, 5 - 4, -8 - 4] = MAX[-7, 1, -12] = 1$



Matrix Fill (Global Alignment)

		-	G	A	A	T	T	C	A	G	T	T	A
	0	-4	-8	-12	-16	-20	-24	-28	-32	-36	-40	-44
-	-4	5	1	-3	-7	-11	-15	-19	-23	-27	-31	-35
G	-8											
G	-12											
A	-16											
T	-20											
C	-24											
G	-28											
A												

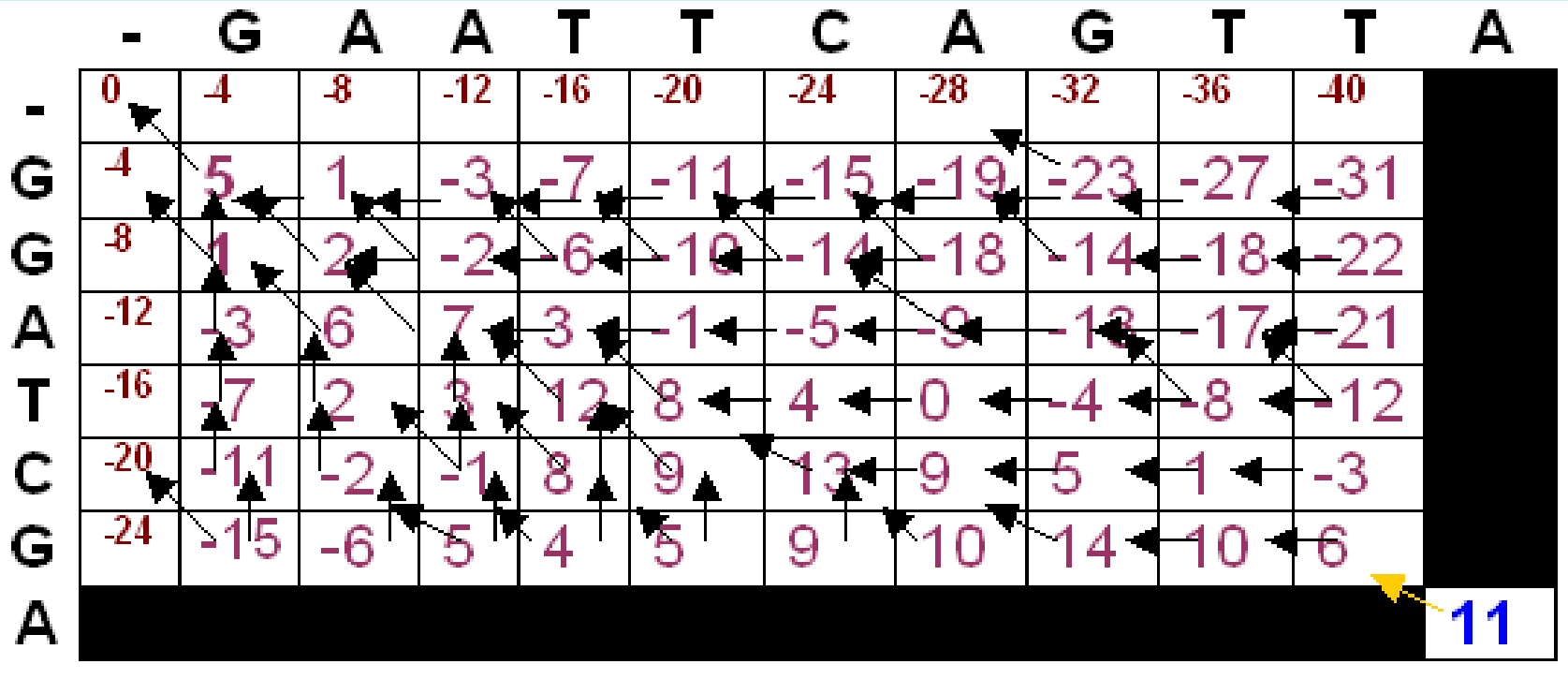
Filled Matrix (Global Alignment)



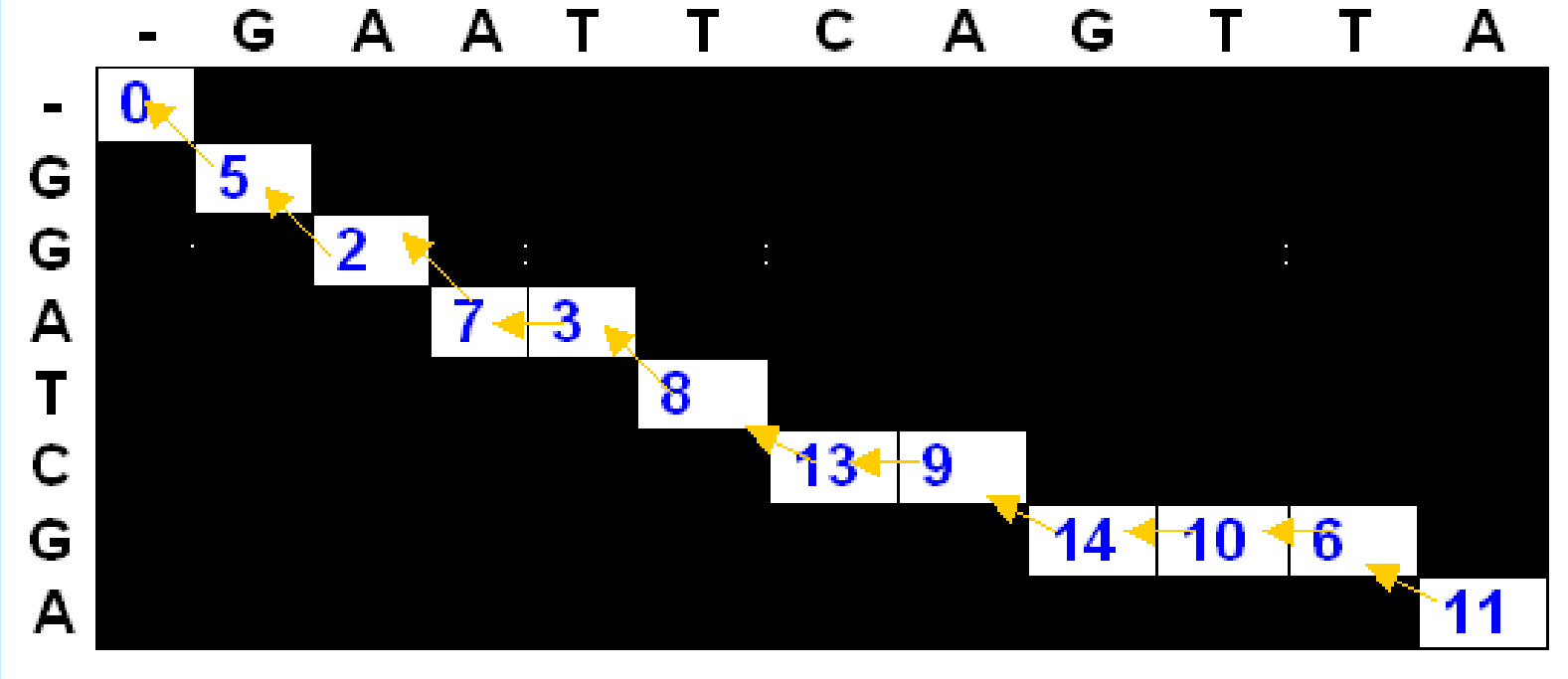
Trace Back (Global Alignment)

- Maximum global alignment score = 11 (value in the lower right hand cell).
- Traceback begins in position $S_{M,N}$; i.e. the position where both sequences are globally aligned.
- At each cell, we look to see where we move next according to the pointers.

Trace Back (Global Alignment)



Global Trace Back




```
G A A T T C A G T T A
|   |   | |   |     |
G G A - T C - G - - A
```

Checking Alignment Score

```
G A A T T C A G T T A
|   |   | |   |     |
G G A - T C - G - - A

+ - + - + + - + - - +
5 3 5 4 5 5 4 5 4 4 5
```

5 – 3 + 5 – 4 + 5 + 5 – 4 + 5 – 4 – 4 + 5 = 11 ✓

Local Alignment

- Smith-Waterman: obtain highest scoring local match between two sequences
- Requires 2 modifications:
  - Negative scores for mismatches
  - When a value in the score matrix becomes negative, reset it to zero (begin of new alignment)

Local Alignment Initialization

- Values in row 0 and column 0 set to 0.

	-	G	A	A	T	T	C	A	G	T	T	A
-	0	0	0	0	0	0	0	0	0	0	0	0
G	0											
G	0											
A	0											
T	0											
C	0											
G	0											
A	0											

Matrix Fill (Local Alignment)

$S_{i,j}$ = **MAXIMUM[**

$S_{i-1, j-1} + s(a_i, b_j)$ (match/mismatch in the diagonal),

$S_{i,j-1} + w$ (gap in sequence #1),

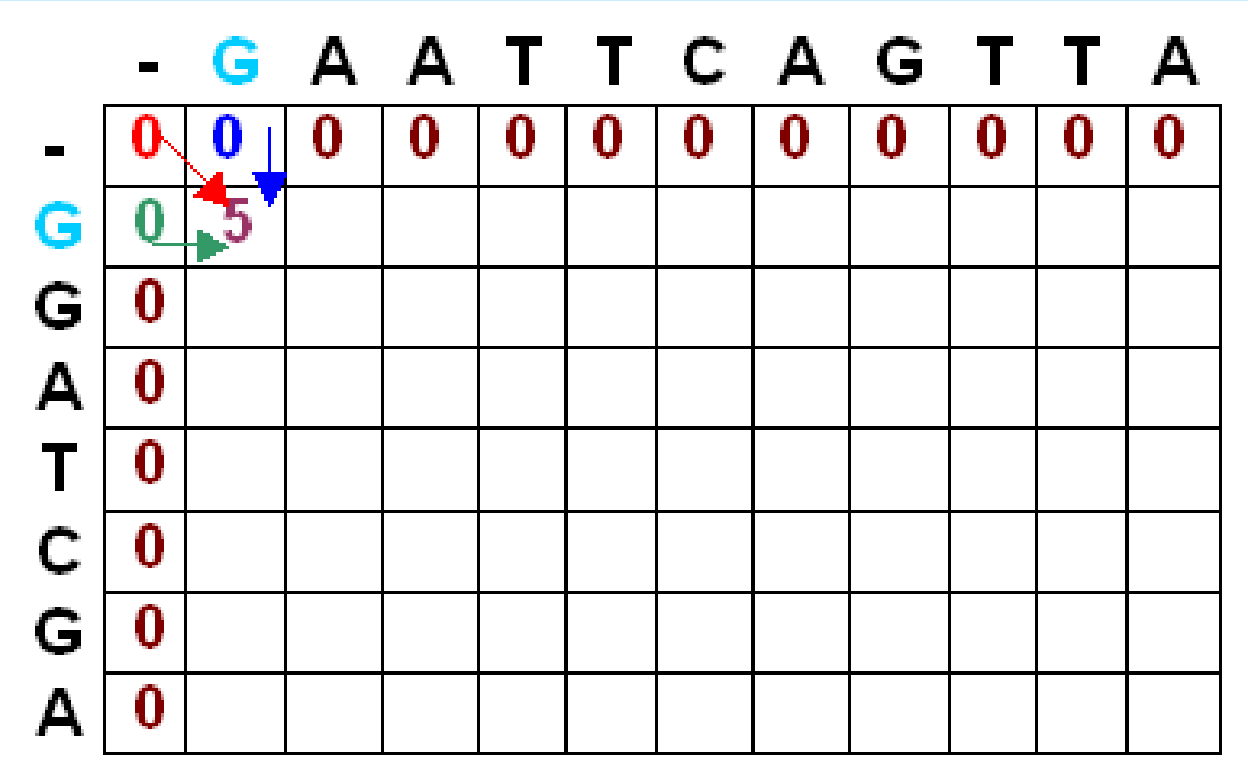
$S_{i-1,j} + w$ (gap in sequence #2),

0

]

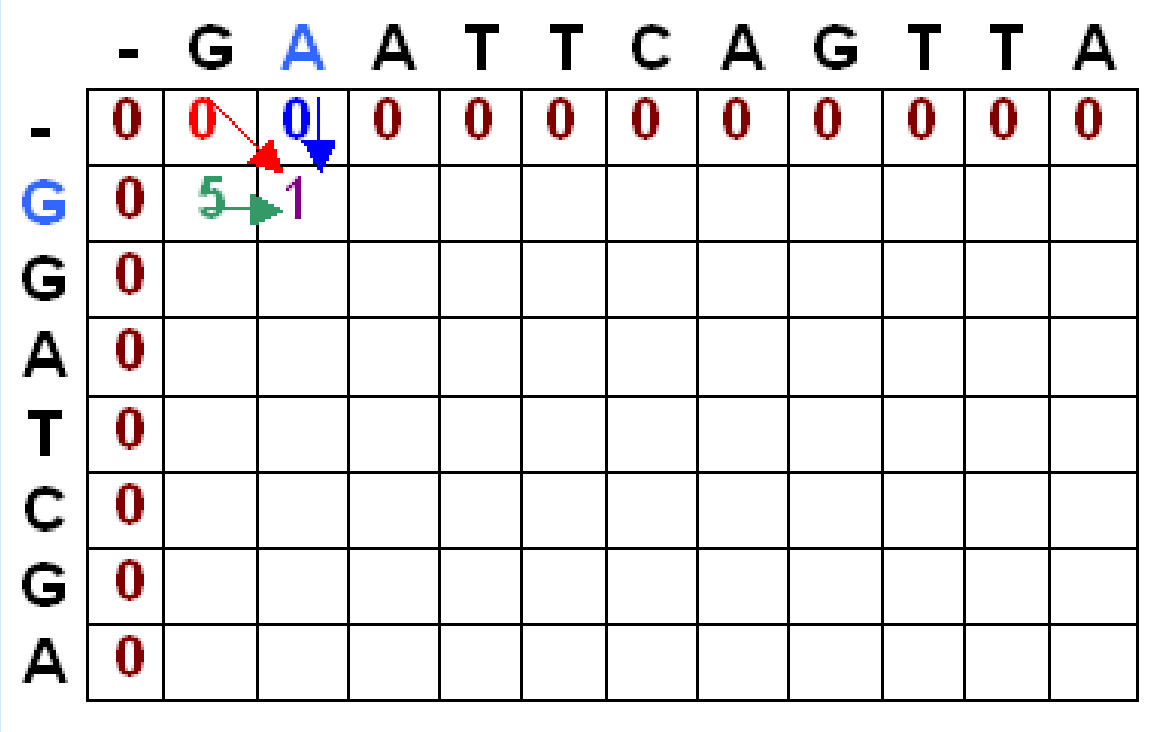
Matrix Fill (Local Alignment)

- $S_{1,1} = MAX[S_{0,0} + 5, S_{1,0} - 4, S_{0,1} - 4, 0] = MAX[5, -4, -4, 0] = 5$



Matrix Fill (Local Alignment)

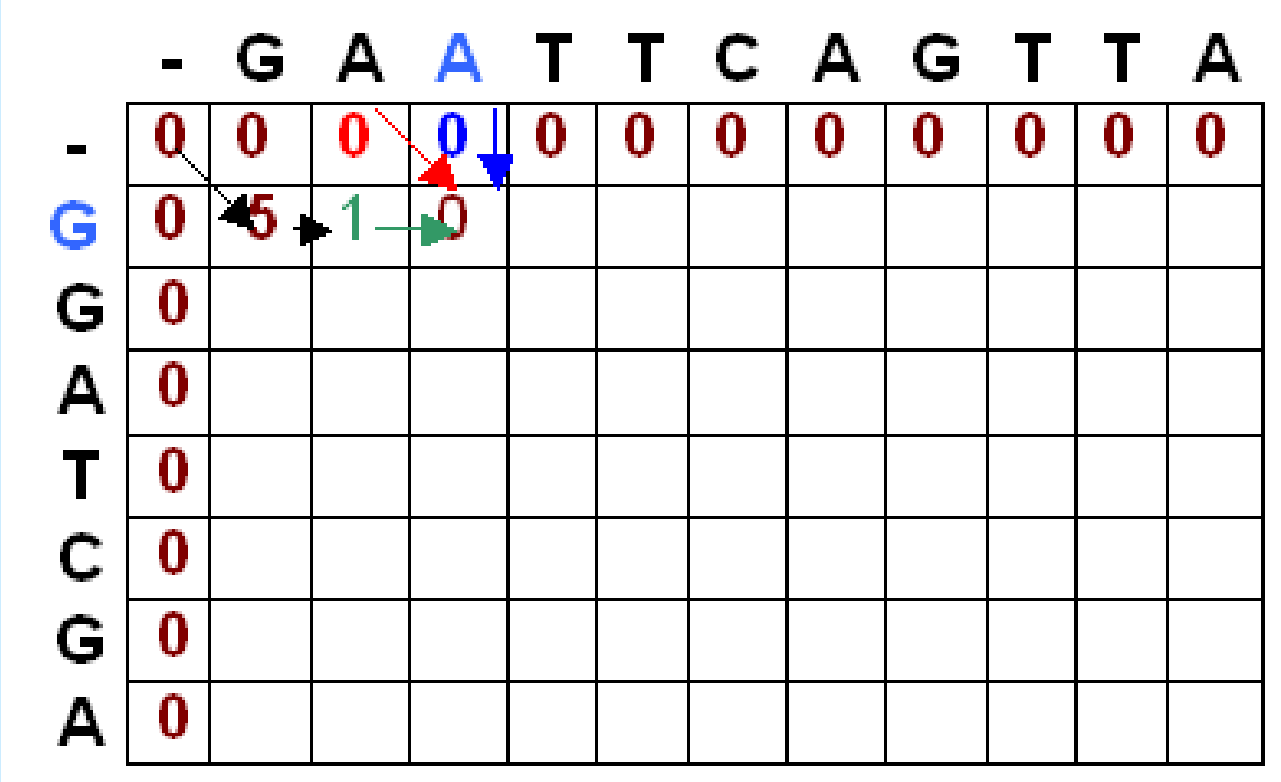
- $S_{1,2} = MAX[S_{0,1} -3, S_{1,1} - 4, S_{0,2} – 4, 0] = MAX[0 - 3, 5 – 4, 0 – 4, 0] = MAX[-3, 1, -4, 0] = 1$



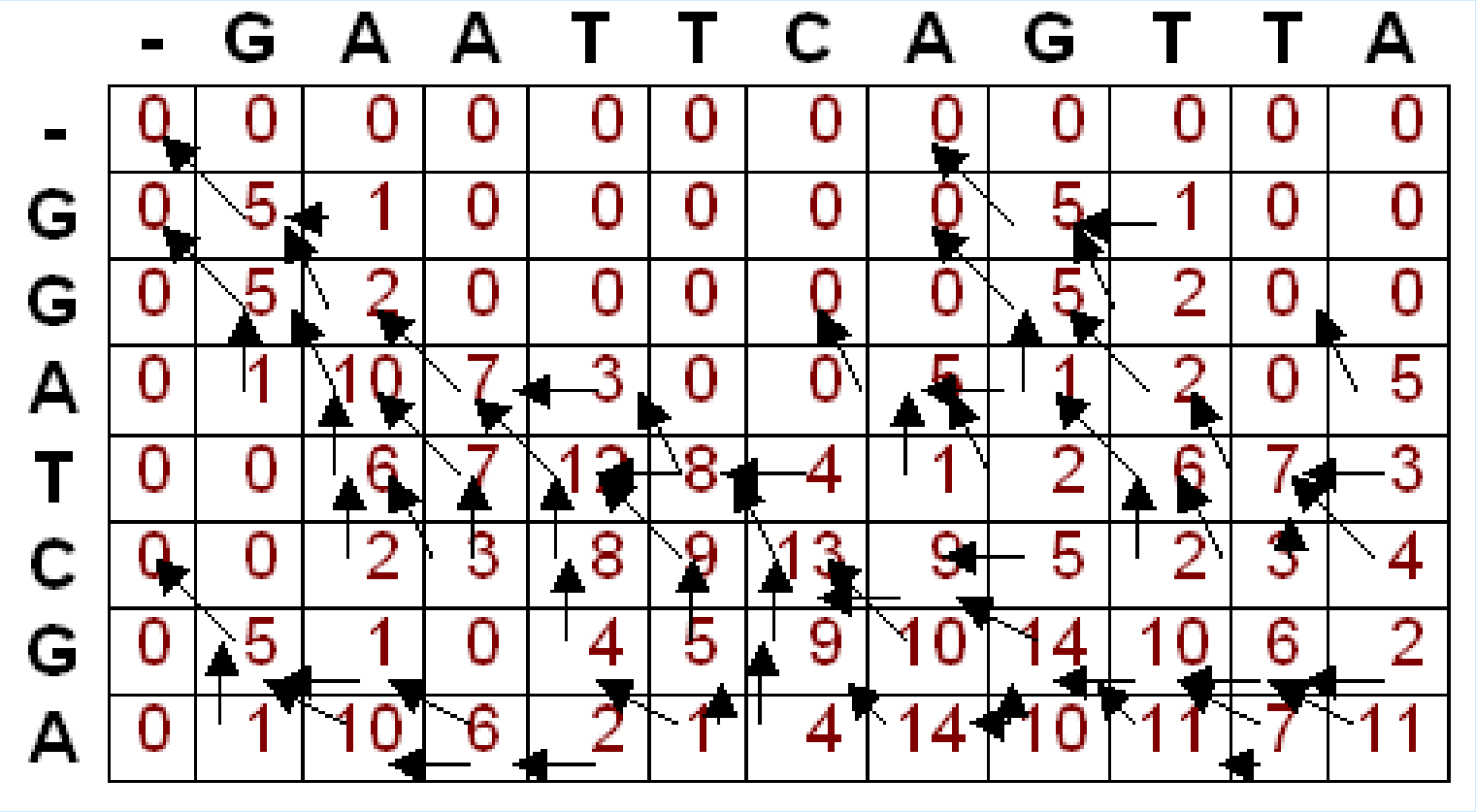
Matrix Fill (Local Alignment)

$S_{1,3} = MAX[S_{0,2} - 3, S_{1,2} - 4, S_{0,3} - 4, 0] = MAX[0 - 3, 1 - 4, 0 - 4, 0] =$

$MAX[-3, -3, -4, 0] = 0$



Filled Matrix (Local Alignment)



	-	G	A	A	T	T	C	A	G	T	T	A
-	0	0	0	0	0	0	0	0	0	0	0	0
G	0	5	1	0	0	0	0	0	5	1	0	0
G	0	5	2	0	0	0	0	0	5	2	0	0
A	0	1	10	7	3	0	0	5	1	2	0	5
T	0	0	6	7	12	8	4	1	2	6	7	3
C	0	0	2	3	8	9	13	9	5	2	3	4
G	0	5	1	0	4	5	9	10	14	10	6	2
A	0	1	10	6	2	1	4	14	10	11	7	11

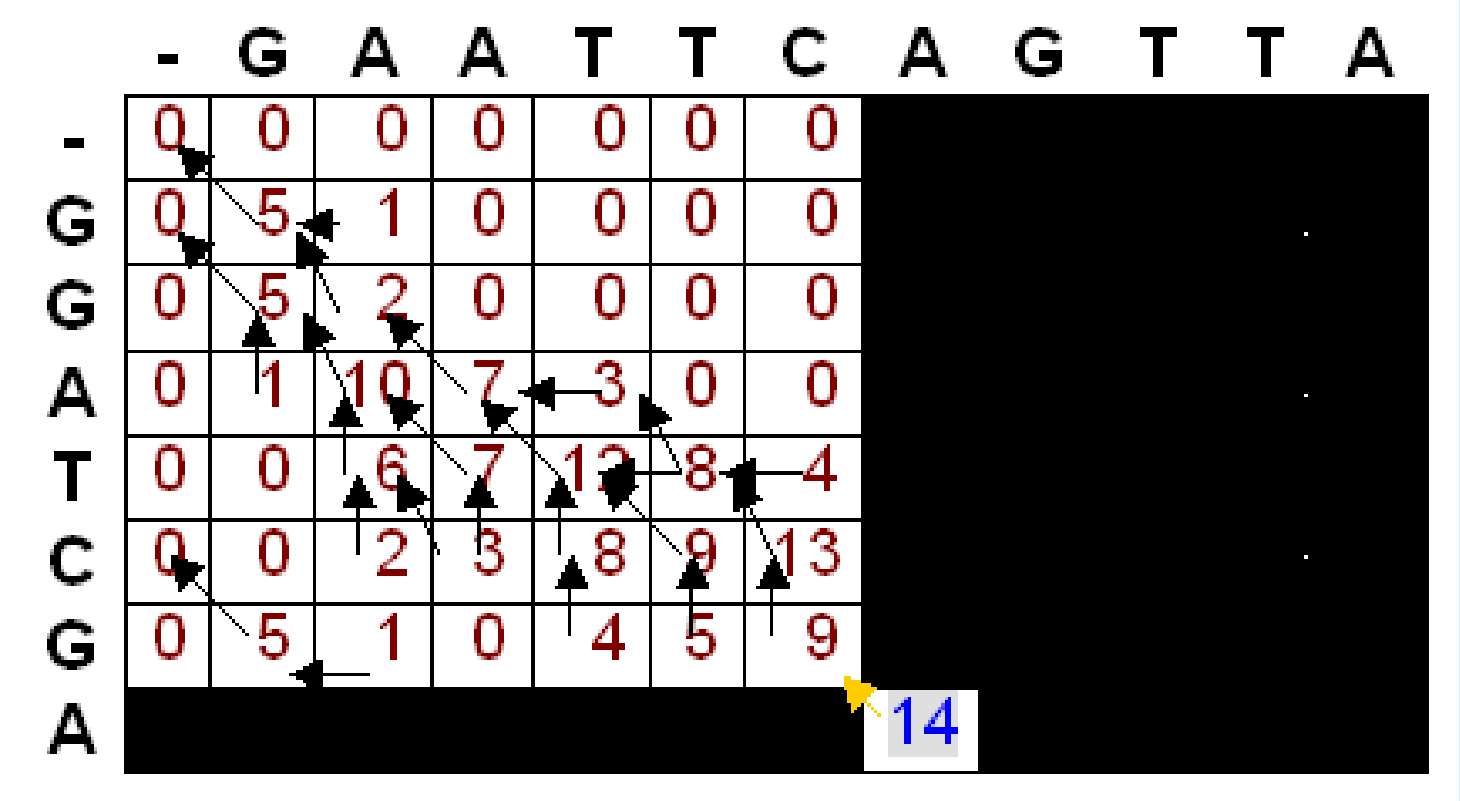
Trace Back (Local Alignment)

- Maximum local alignment score for the two sequences is 14
- Found by locating the highest values in the score matrix
- 14 is found in two separate cells, indicating multiple alignments producing the maximal alignment score

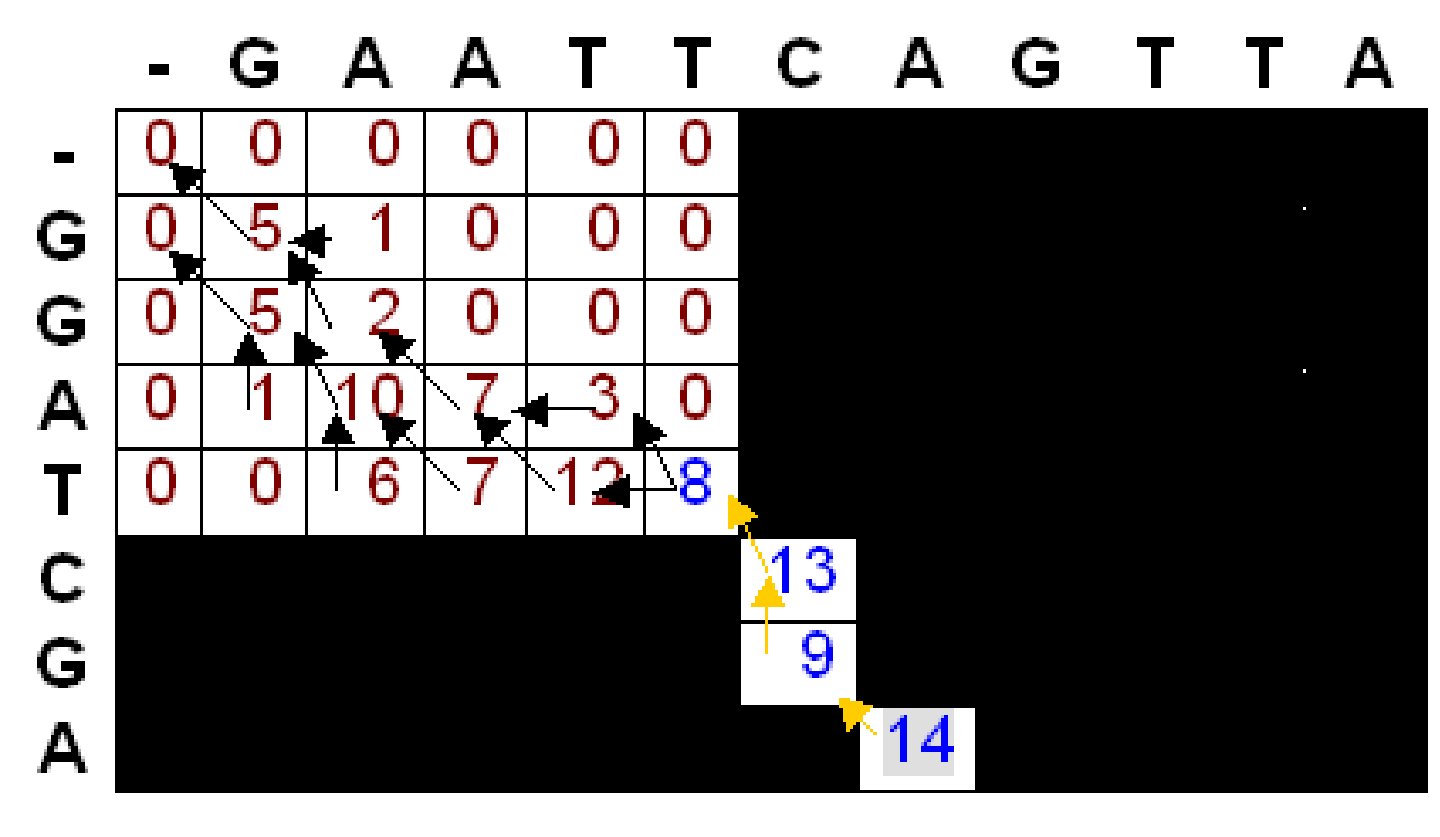
Trace Back (Local Alignment)

- Traceback begins in the position with the highest value.
- At each cell, we look to see where we move next according to the pointers
- When a cell is reached where there is not a pointer to a previous cell, we have reached the beginning of the alignment

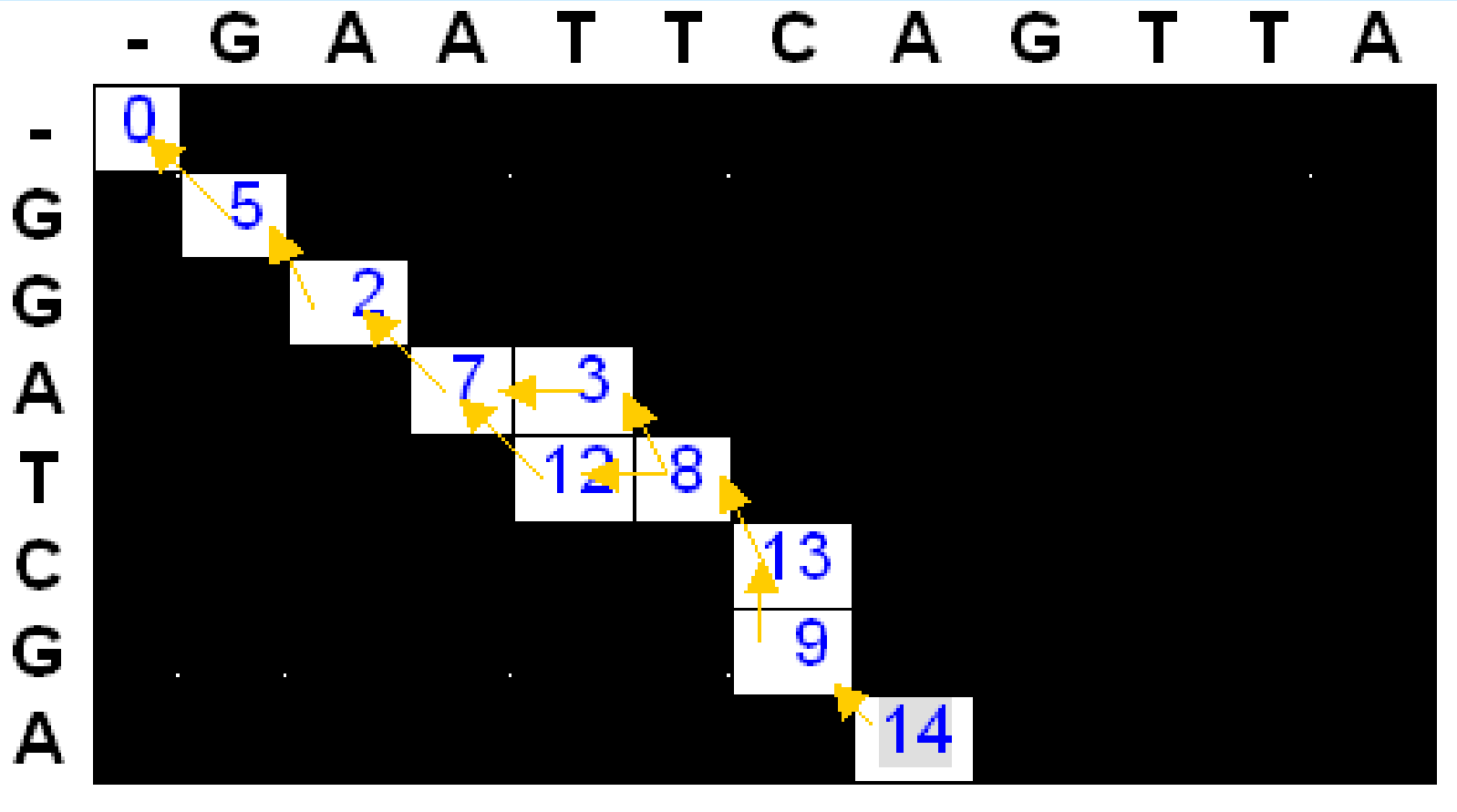
Trace Back (Local Alignment)



Trace Back (Local Alignment)



Trace Back (Local Alignment)



Maximum Local Alignment

```
G A A T T C - A      G A A T T C - A
|   | |   |   |      |   |   | |   |
G G A T - C G A      G G A - T C G A

+ - + + - + - +      + - + - + + - +
5 3 5 5 4 5 4 5      5 3 5 4 5 5 4 5
```

Dynamic Programming

- Dynamic programming is a general approach to design algorithms in computer science.
- The idea is:
  - If a function f(n) can be computed using f(n-1), f(n-2), …, f(1), then we can compute f(i) for i from 1 to n.

Alignment

- Let |s| = m and |t| = n. Let DP[i,j] be the optimal alignment score for s[1..i] and t[1..j].
- Therefore, the optimal alignment score of s and t is DP[m,n].
- Now let's find out how to compute DP[i,j] from DP[a,b] for all a<i or b<j.

Recursive definition

- Case 1: s[i] matches t[j]
  - DP[i,j] = DP[i-1, j-1] + f(s[i], t[j]);
- Case 2: s[i] matches -
  - DP[i,j] = DP[i-1, j] + f(s[i], -);
- Case 3: t[j] matches -
  - DP[i,j] = DP[i, j-1] + f(-, t[j]);
- Therefore…

$$DP[i,j] = \max \begin{cases} DP[i-1, j-1] + f(s[i], t[j]); \\ DP[i-1, j] + f(s[i], -); \\ DP[i, j-1] + f(-, t[j]); \end{cases}$$

Algorithm

DP[0,0] = 0;

DP[i,0] = DP[i-1,0] + f(s[i],-);

DP[0,j] = DP[0,j-1] + f(-, t[j]);

for i from 1 to m

  for j from 1 to n

$$DP[i,j] = \max \begin{cases} DP[i-1, j-1] + f(s[i], t[j]); \\ DP[i-1, j] + f(s[i], -); \\ DP[i, j-1] + f(-, t[j]); \end{cases}$$

Output DP[m,n];

Complexity

- Time complexity O(mn)
- Space complexity O(mn)
- There is an algorithm that outputs the best alignment with O(min(m,n)) space and O(mn) time.

Fit one sequence into the other

- Let s, t be two sequences. Find t[a..b] that aligns with s with the best alignment score.
- Let DP[i,j] be the best alignment score between s[1..i] and t[a..j].
- Then the best alignment score is

  max(DP[i,n]) for all i.

Recursive definition

- Case 1: s[i] matches t[j]
  - DP[i,j] = DP[i-1, j-1] + f(s[i], t[j]);
- Case 2: s[i] matches -
  - DP[i,j] = DP[i-1, j] + f(s[i], -);
- Case 3: t[j] matches -
  - DP[i,j] = DP[i, j-1] + f(-, t[j]);
- Therefore

$$DP[i,j] = \max \begin{cases} DP[i-1, j-1] + f(s[i], t[j]); \\ DP[i-1, j] + f(s[i], -); \\ DP[i, j-1] + f(-, t[j]); \end{cases}$$

Algorithm

DP[0,0] = 0;

DP[i,0] = DP[i-1,0] + f(s[i],-);

DP[0,j] = 0;

for i from 1 to m

  for j from 1 to n

$$DP[i,j] = \max \begin{cases} DP[i-1, j-1] + f(s[i], t[j]); \\ DP[i-1, j] + f(s[i], -); \\ DP[i, j-1] + f(-, t[j]); \end{cases}$$

Output max(DP[i,n]);

Local Alignment

- Given sequences s, t, find s[a..b] and t[c..d] such that the alignment score of s[a..b] and t[c..d] are the highest.
- Example:
  - AGATTGTGG-TA
  - AC-TTG-GGATG
  - The best local alignment should be TTGTGG v.s. TTGGG

A better method

- Let DP[i,j] be the optimal alignment score for s[a,i] and s[b, j] for all a<=i and b<=j.
- Then the best local alignment score is

  max(DP[i,j]) for all i and j.

A better method

- Case 1: s[i] matches t[j]
  - DP[i,j] = DP[i-1, j-1] + f(s[i], t[j]);
- Case 2: s[i] matches -
  - DP[i,j] = DP[i-1, j] + f(s[i], -);
- Case 3: t[j] matches -
  - DP[i,j] = DP[i, j-1] + f(-, t[j]);
- Case 4: DP[i,j]=0;
- Therefore…

$$DP[i,j] = \max \begin{cases} 0 \\ DP[i-1, j-1] + f(s[i], t[j]); \\ DP[i-1, j] + f(s[i], -); \\ DP[i, j-1] + f(-, t[j]); \end{cases}$$

Algorithm

DP[0,0] = 0;

DP[i,0] = 0;

DP[0,j] = 0;

for i from 1 to m

  for j from 1 to n

$$DP[i,j] = \max \begin{cases} 0 \\ DP[i-1, j-1] + f(s[i], t[j]); \\ DP[i-1, j] + f(s[i], -); \\ DP[i, j-1] + f(-, t[j]); \end{cases}$$

Output max(DP[i,j]);

Recursive definition

- Let DP0[i,j] be the alignment score that s[i] matches t[j].
- Let DP1[i,j] be the alignment score that s[i] matches -.
- Let DP2[i,j] be the alignment score that t[j] matches -.

Recursive definition

$$DP0[i,j] = f(s[i], t[j]) + \max \begin{cases} DP0[i-1, j-1]; \\ DP1[i-1, j-1]; \\ DP2[i, j-1]; \end{cases}$$

$$DP1[i,j] = f(s[i], -) + \max \begin{cases} DP0[i-1, j] + a; \\ DP1[i-1, j]; \\ DP2[i-1, j] + a; \end{cases}$$

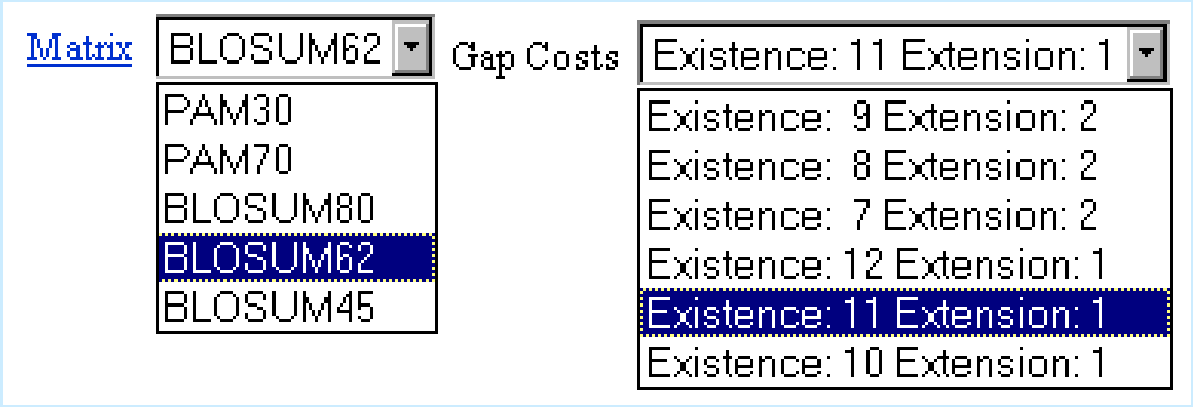
$$DP2[i,j] = f(s[i], t[j]) + \max \begin{cases} DP0[i, j-1] + a; \\ DP1[i, j-1] + a; \\ DP2[i, j-1]; \end{cases}$$

Drawbacks to DP Approaches

- Compute intensive
- Memory Intensive

Scoring Matrices

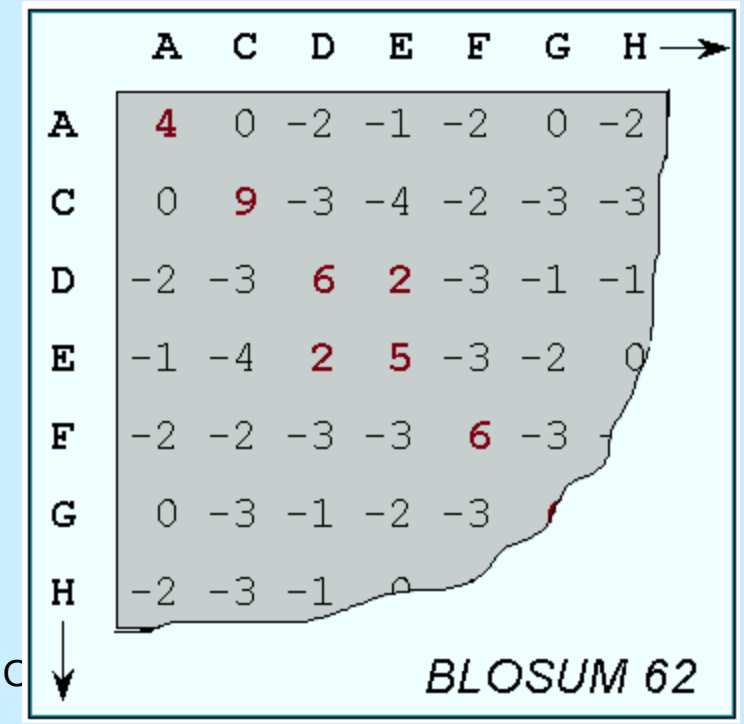
Blastp ~ Special Parameters



Gap: penalties for opening a new gap, or for extending an existing gap.

Matrix: a table of scores that are assigned to various amino acid substitutions. In general, different substitution matrices are tailored to detecting similarities among sequences that are diverged by differing degrees.

BLOSUM-62 matrix is among the best for detecting most weak protein similarities. For particularly long and weak alignments, the BLOSUM-45 matrix may prove superior. For short queries, PAM matrices may be used instead.



PAM

- Percent Accepted Mutation (PAM)
- Studied by Margaret Dayhoff in1978
- Amino acid substitutions
  - 71 groups of protein, 85% similar
  - 1572 amino acid substitutions
- “Accepted” mutations – do not negatively affect a protein’s fitness

PAM

- Similar sequences organized into phylogenetic trees
- Number of amino acid changes counted
- Relative mutabilities evaluated
- 20 x 20 amino acid substitution matrix calculated

PAM

- PAM 1: 1 accepted mutation event per 100 amino acids; PAM 250: 250 mutation events per 100 …
- PAM 1 matrix can be multiplied by itself N times to give transition matrices for sequences that have undergone N mutations
- PAM 250: 20% similar; PAM 120: 40%; PAM 80: 50%; PAM 60: 60%

PAM1 Matrix

normalized probabilities multiplied by 10000

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	9867	2	9	10	3	8	17	21	2	6	4	2	6	2	22	35	32	0	2	18
R	1	9913	1	0	1	10	0	0	10	3	1	19	4	1	4	6	1	8	0	1
N	4	1	9822	36	0	4	6	6	21	3	1	13	0	1	2	20	9	1	4	1
D	6	0	42	9859	0	6	53	6	4	1	0	3	0	0	1	5	3	0	0	1
C	1	1	0	0	9973	0	0	0	1	1	0	0	0	0	1	5	1	0	3	2
Q	3	9	4	5	0	9876	27	1	23	1	3	6	4	0	6	2	2	0	0	1
E	10	0	7	56	0	35	9865	4	2	3	1	4	1	0	3	4	2	0	1	2
G	21	1	12	11	1	3	7	9935	1	0	1	2	1	1	3	21	3	0	0	5
H	1	8	18	3	1	20	1	0	9912	0	1	1	0	2	3	1	1	1	4	1
I	2	2	3	1	2	1	2	0	0	9872	9	2	12	7	0	1	7	0	1	33
L	3	1	3	0	0	6	1	1	4	22	9947	2	45	13	3	1	3	4	2	15
K	2	37	25	6	0	12	7	2	2	4	1	9926	20	0	3	8	11	0	1	1
M	1	1	0	0	0	2	0	0	0	5	8	4	9874	1	0	1	2	0	0	4
F	1	1	1	0	0	0	0	1	2	8	6	0	4	9946	0	2	1	3	28	0
P	13	5	2	1	1	8	3	2	5	1	2	2	1	1	9926	12	4	0	0	2
S	28	11	34	7	11	4	6	16	2	2	1	7	4	3	17	9840	38	5	2	2
T	22	2	13	4	1	3	2	2	1	11	2	8	6	1	5	32	9871	0	2	9
W	0	2	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	9976	1	0
Y	1	0	3	0	3	0	1	0	4	1	1	0	0	21	0	1	1	2	9945	1
V	13	2	1	1	3	2	2	3	3	57	11	1	17	1	3	2	10	0	2	9901

BLOSUM

- 1992, more data available than 1978
- BLOcks amino acid SUbstitution Matrices
- Larger set of sequences considered
- Sequences organized into signature blocks
- Directly calculated
  - BLOSUM 62, no more than 62% identical
  - BLOSUM 80, no more than 80% identical

- BLOSUM62

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

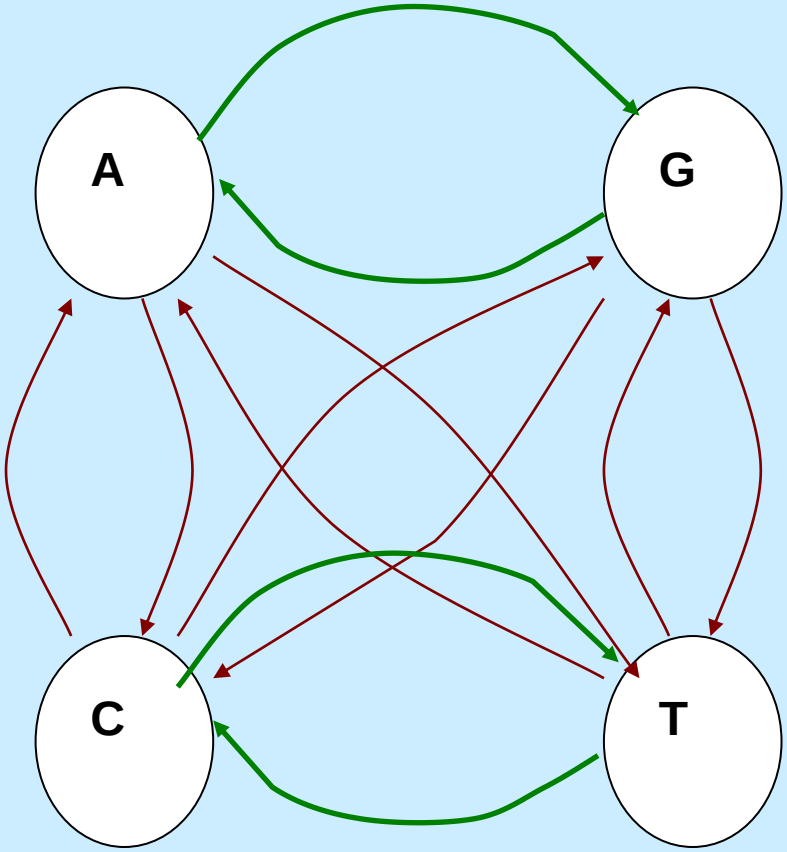
What Matrices Should Be Used

- BLOSUM30
- PAM250 ~ BLOSUM45: longer alignments of more divergent sequences (<30%)
- PAM160 ~ BLOSUM62
- PAM120 ~ BLOSUM80: Detecting members of a protein family (50-60%)
- PAM70
- PAM40 ~ BLOSUM90: short alignments that are highly similar (70-90%)

Nucleic Acid Scoring Matrices

- Two mutation models:
  - Uniform mutation rates (Jukes-Cantor)
  - Two separate mutation rates (Kimura)
    - Transitions
    - Transversions

DNA Mutations



PURINES: A, G
PYRIMIDINES C, T

Transitions: A↔G; C↔T
Transversions: A↔C, A↔T,
C↔G, G↔T

PAM1 DNA odds matrices

A. Model of uniform mutation rates among nucleotides.

	A	G	T	C
A	0.99			
G	0.00333	0.99		
T	0.00333	0.00333	0.99	
C	0.00333	0.00333	0.00333	0.99

B. Model of 3-fold higher transitions than transversions.

	A	G	T	C
A	0.99			
G	0.006	0.99		
T	0.002	0.002	0.99	
C	0.002	0.002	0.006	0.99

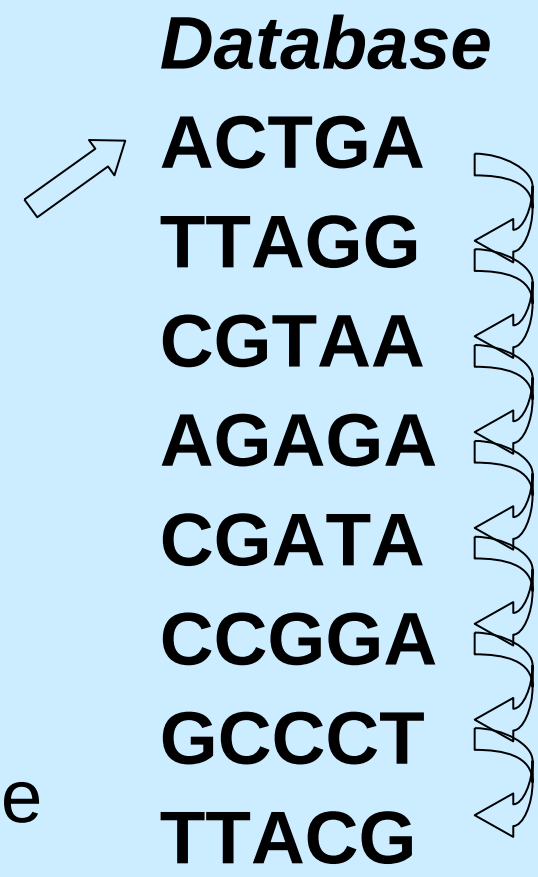
Database Alignment Search

Database Alignment Search

- Scheme
  - A query against a sequence
  - A query against a database!
- Search method
  - Linear search
  - Binary search
  - Hash search

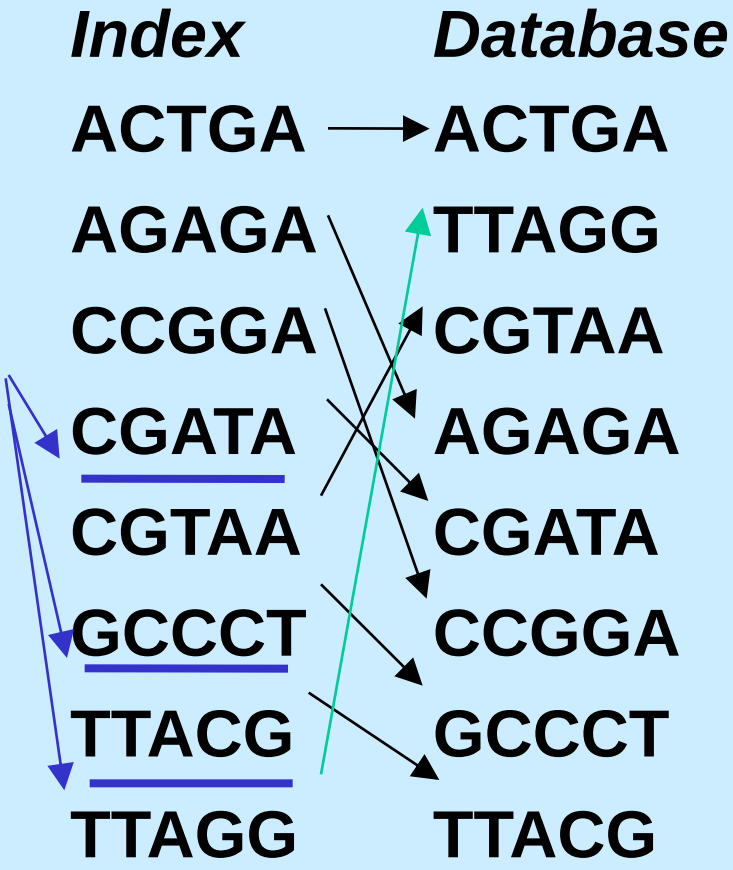
Linear search

- Test query against each target sequentially
- Worst case, query matches last target and you have as many tests as targets (size of database)
- Average case, test half the targets.
- Linear in the size of the database



Indexed (binary) search

- Create an sorted set of keys that point to entries
- Start in the middle, then figure out which half
- Eliminate half the database each step, so need $\log_2$ steps at worst
- Need to build the index (takes space and time at each database update)



Hash tables

- Map each query to an arbitrary number with a “hash function”
- Use those numbers as an index into a table
- “Collisions” can happen, but are rare
- Constant time lookup

f (TTACG)= 8

Hash table

1. CGATA

2. GCCCT

3. CGTAA, AGAGA

4.

5. ACTGA

6. CCGGA

7. TTAGG

8. TTACG

BLAST

What is BLAST?

BLAST® (Basic *Local* Alignment Search Tool) is a set of similarity search programs designed to explore all of the available sequence databases regardless of whether the query is protein or DNA.

Currently, it is the most popular and most accepted sequence analysis tool.

Why BLAST?

• Identify unknown sequences - The best way to identify an unknown sequence is to see if that sequence already exists in a public database. If the database sequence is a well-characterized sequence, then you may have access to a wealth of biological information.

• Help gene/protein function and structure prediction – genes with similar sequences tend to share similar functions or structure.

• Identify protein family – group related (paralog or ortholog) genes and their proteins into a family.

•Prepare sequences for multiple alignments

• And more …

Why BLAST?

- Dynamic programming solutions to alignment problems are relatively slow, and don't lend themselves to efficient database search.
- Need some way to search a large database to find sequences that have an inexact match to a query sequence

Why BLAST?

- Competing solutions: FASTA & BLAST.
  - Both imperfect approximations to DP. DP finds some distantly related sequences the approximations don't
  - BLAST is more commonly used, although both are fine.

How does BLAST work

- Break the query into overlapping “words,” by default of length 3.
  - ACDEF → ACD, CDE, DEF
- For each word, define ~50 other words that are similar (use a score matrix and a threshold)
  - ACD →ACE, GCD, ...

How Does BLAST work

- Repeat for each of the words.
- Use a hash table to find all places in DB with *exact match* to any of those words.

How Does BLAST work

- Identify database sequences that contain *several matching words on the same diagonal* and within a short distance.
- Extend these short, ungapped alignments in both directions along the sequence so long as score of alignment increases.
- Call these extended alignments HSP's for “high scoring pairs”

How Does BLAST Work

Two-step procedure:

1. Compare query sequence to every database entries. For each entry, if there are segments of certain length (word size) similar to part of the query sequence, they have a hit.

   Word size = 7

   Query: GTTGACCGTTAGCCGACGTTAAGCT

   DB entry: ACATAGCCCGTTAGCCGCTGATACGACCGTAC

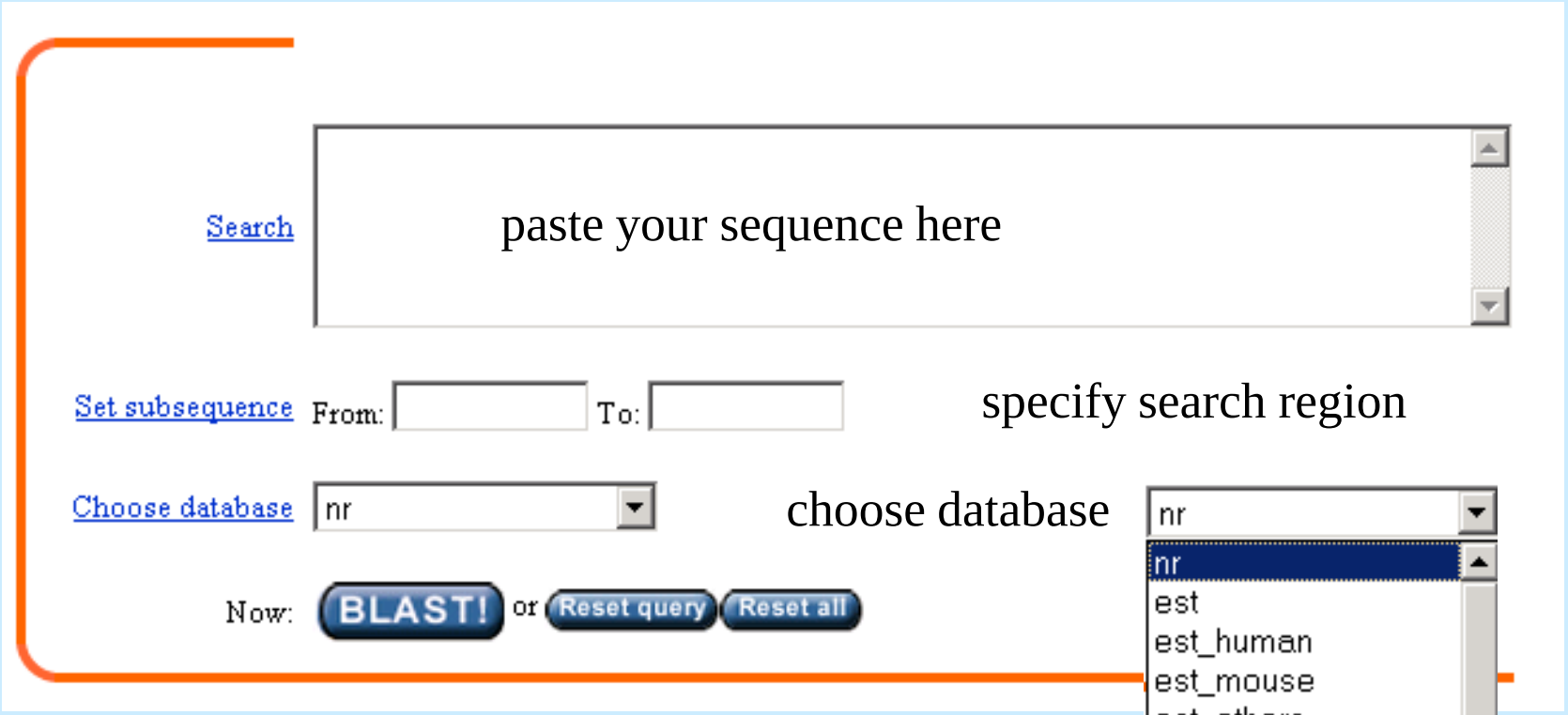
2. For each hit, extending two both ends until the expect value falls below the threshold. They become “high scoring pair” (HSP)

3. A Smith-Waterman like algorithm is used to do local alignment around each HSP.

BLAST

- Default on NCBI web site
- Provides gapped alignments
- Use BLAST to find HSPs, then runs DP to find optimal alignments. Fast (enough) database search, along with optimal alignments.
  - Still might miss some alignments DP would find as database search tool

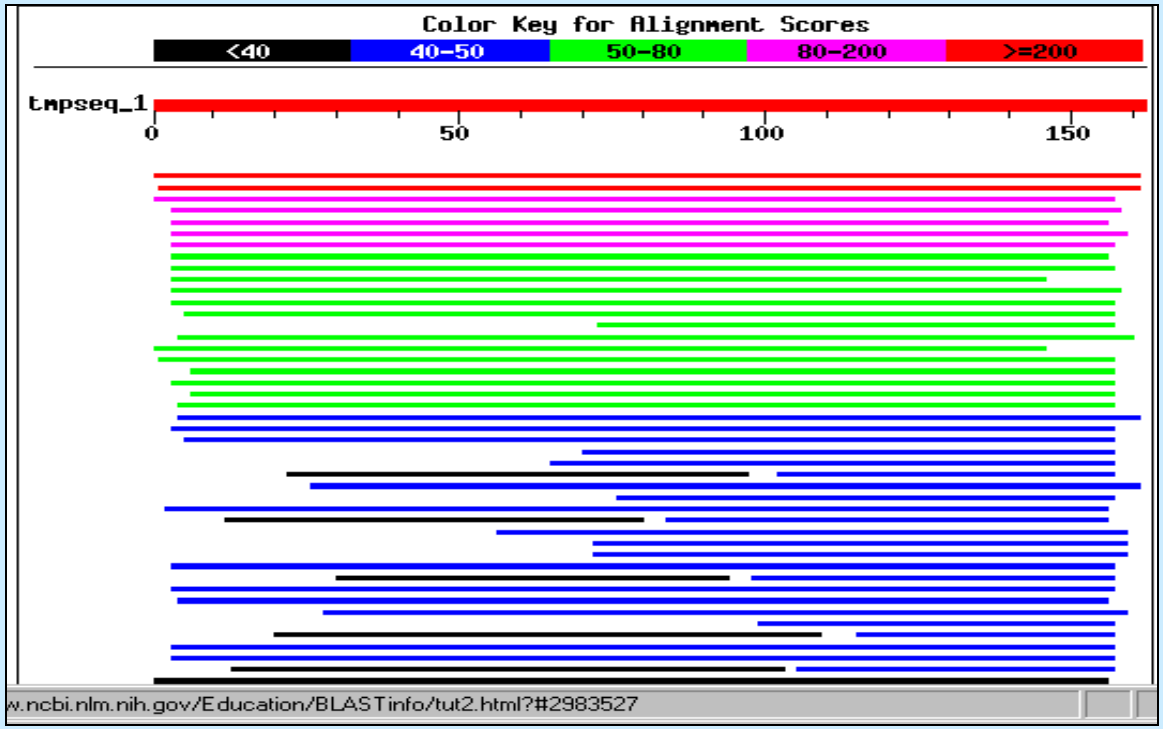
Blastn ~ Construct Queries



nr ~ non-redundant database

Others are subsets of nr database.

Interpret BLAST results - Distribution



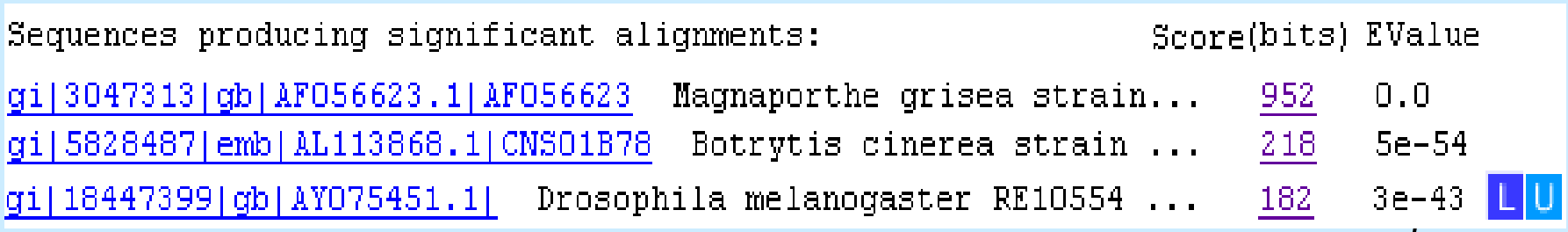
Query sequence

BLAST hits. Click to access the pairwise alignment.

This image shows the distribution of BLAST hits on the query sequence. Each line represents a hit. The span of a line represents the region where similarity is detected. Different colors represent different ranges of scores.

Interpret BLAST results - Description

The description (also called definition) lines are listed below under the heading "Sequences producing significant alignments". The term "significant" simply refers to all those hits whose E value was less than the threshold. It does **not** imply biological significance.



ID (GI #, refseq #, DB-specific ID #) Click to access the record in GenBank

Gene/sequence Definition

Bit score – higher, better. Click to access the pairwise alignment

Expect value – lower, better. It tells the possibility that this is a random hit

Links

BLASTn

- The default BLAST
- DNA query
- DNA database
- About 100 times faster than DP

BLAST 2

```
>gi|1044939|emb|X92422.1|ZMUBIS27F  Z.mays mRNA for ubiquitin/ribosomal protein S27a fusion protein
          Length = 777

 Score =  165 bits (83), Expect = 7e-38
 Identities = 146/167 (87%)
 Strand = Plus / Plus


Query: 1   gagtcctcggacacgatcgacaatgtgaagtccaagattcaggacaaggagggaattccc 60
           ||||||||||||| |||||||| |||||| ||||||| |||||||||||||| || ||
Sbjct: 142 gagtcctcggacaccatcgacaacgtgaaggccaagatccaggacaaggagggcatcccg 201


Query: 61  ccggaccaacagcgtcttatctttgctggcaagcagctggaggacggccgtactctctcc 120
           |||||||| ||||| || ||||| || |||||||||| ||||| ||||| || ||| ||
Sbjct: 202 ccggaccagcagcgcctcatcttcgccggcaagcagctcgaggatggccgcaccctcgcc 261


Query: 121 gactacaacatccagaaggagtcgaccctccacttggtcctccgcct 167
           ||||||||||||||||||||||||| || ||| ||||| |||||||
Sbjct: 262 gactacaacatccagaaggagtcgacgctgcacctggtcgtccgcct 308
```

Query line: the segment from query sequence.

Subj line: the segment from hit (subject) sequence.

Middle line: the consensus bases

MEGABLAST

- MEGABLAST is the tool of choice to identify a nucleotide sequence.
- Long, exact match

Discontiguous MEGABLAST

- Discontiguous MEGABLAST is better at finding nucleotide sequences similar, but not identical, to your nucleotide query.

Search for short nearly exact matches

- "Search for short nearly exact matches" is useful for primer or short nucleotide motif searches.

BLASTP

- Standard protein BLAST is designed for protein searches

PSI-BLAST

- PSI-BLAST is designed for more sensitive protein-protein similarity searches
- Position-Specific Iterated

PHI-BLAST

- PHI-BLAST can do a restricted protein pattern search.
- Pattern-Hit Initiated

BLASTX

- The “Nucleotide query - Protein db [blastx]” is useful for finding similar proteins to those encoded by a nucleotide query
- useful for new sequence query, frame is unknown, could has frame shift error.
- First analysis

TBLASTN

- The “Protein query - Translated db [tblastn]” search is useful for finding protein homologs in unnannotated nucleotide data.
- New genome draft

TBLASTX

- The "Nucleotide query - Translated db [tblastx]" is useful for identifying novel genes in error prone query sequences.

Genome BLAST

- Human, mouse, rat, chimp , cow, pig, dog, sheep, cat
- Chicken, puffer fish, zebrafish
- Malaria
- Insects, nematodes, plants, fungi, microbial genomes, other eukaryotic genomes

If your sequence is NUCLEOTIDE

Length	DB	Purpose	Program
20 bp or longer	Nucl	Identify the query sequence	MegaBlast blastn
		Find sequences similar to query sequence	blastn
		Find similar proteins to translated query in a translated database	tblastx
	Prot	Find similar proteins to translated query in a protein database	blastx
7-20 bp	Nucl	Find primer binding sites or map short contiguous motifs	Search for short, nearly exact matches

If Your Sequence is PROTEIN

Length	DB	Purpose	Program
15 residue or longer	Prot	Identify the query sequence or find protein sequences similar to query	blastp
		Find members of a protein family or build a custom position-specific score matrix	PSI-blast
	Nucl	Find similar proteins in a translated nucleotide database	tblastn
5-15 residue	Prot	Search for peptide motifs	Search for short, nearly exact matches

BLAST Family

Program	Description
blastp	Compares an amino acid query sequence against a protein sequence database.
blastn	Compares a nucleotide query sequence against a nucleotide sequence database.
blastx	Compares a nucleotide query sequence translated in all reading frames against a protein sequence database. You could use this option to find potential translation products of an unknown nucleotide sequence.
tblastn	Compares a protein query sequence against a nucleotide sequence database dynamically translated in all reading frames.
tblastx	Compares the six-frame translations of a nucleotide query sequence against the six-frame translations of a nucleotide sequence database. Please note that the tblastx program cannot be used with the nr database on the BLAST Web page because it is computationally intensive.

Choose the BLAST program

Program	Input		Database
blastn	DNA	1	DNA
blastp	protein	1	protein
blastx	DNA	6	protein
tblastn	protein	6	DNA
tblastx	DNA	36	DNA

Topic 4 Review

- Global alignments vs. local alignments
- Dynamic Programming
- PAM and BLOSUM
- Using BLAST, choosing the right one