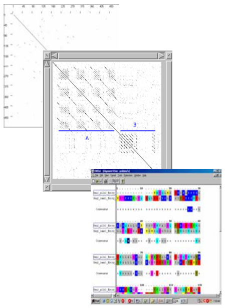
Algorithms in Bioinformatics

Sami Khuri
Department of Computer Science
San José State University
San José, California, USA
khuri@cs.sjsu.edu
www.cs.sjsu.edu/faculty/khuri

Pairwise Sequence Alignment



- **Homology**
- **Similarity**
- **Global string alignment**
- **Local string alignment**
- **Dot matrices**
- **Dynamic programming**
- **Scoring matrices**

Sequence Terminology

Computer Science		Biology
String, word	⟷	Sequence
Substring (contiguous)	⟷	Subsequence
Subsequence	⟷	N/A
Exact matching	⟷	N/A
Inexact Matching	⟷	Alignment

Sequence Alignment

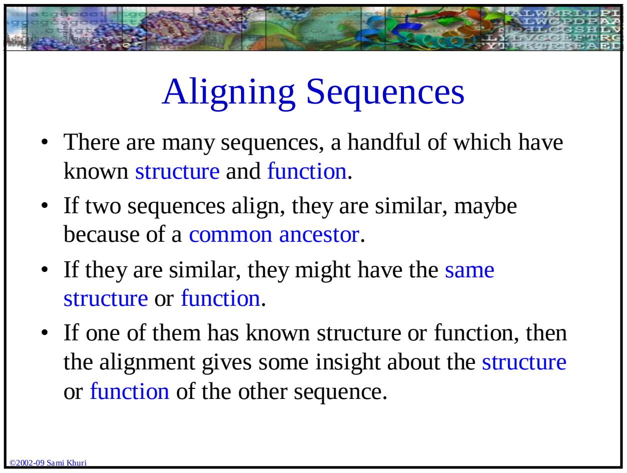
- Sequence alignment is the procedure of comparing sequences by searching for a series of individual characters or character patterns that are in the same order in the sequences.
  - Comparing two sequences gives us a pairwise alignment.
  - Comparing more than two sequences gives us multiple sequence alignment.

Why Do We Align Sequences?

- The basic idea of aligning sequences is that similar DNA sequences generally produce similar proteins.
- To be able to predict the characteristics of a protein using only its sequence data, the structure or function information of known proteins with similar sequences can be used.
- To be able to check and see whether two (or more) genes or proteins are evolutionarily related to each other.

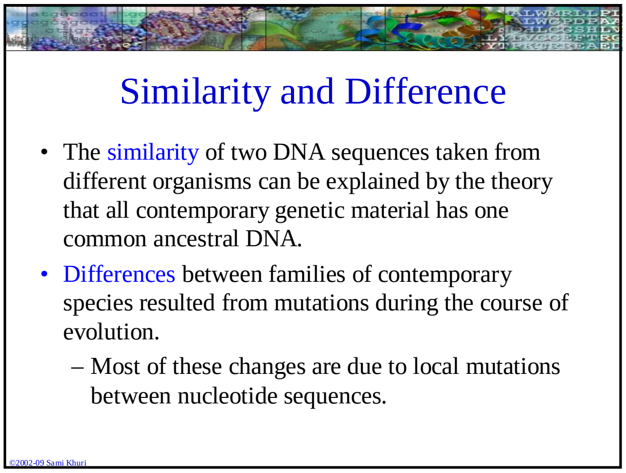
Query Sequence

If a query sequence is found to be significantly similar to an already annotated sequence (DNA or protein), we can use the information from the annotated sequence to possibly infer gene structure or function of the query sequence.



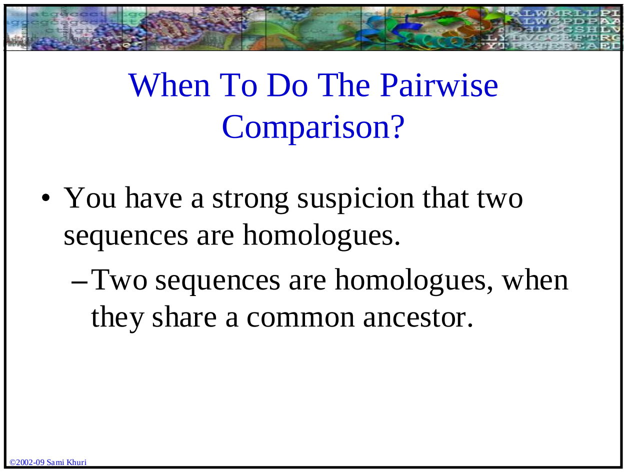
Aligning Sequences

- There are many sequences, a handful of which have known structure and function.
- If two sequences align, they are similar, maybe because of a common ancestor.
- If they are similar, they might have the same structure or function.
- If one of them has known structure or function, then the alignment gives some insight about the structure or function of the other sequence.



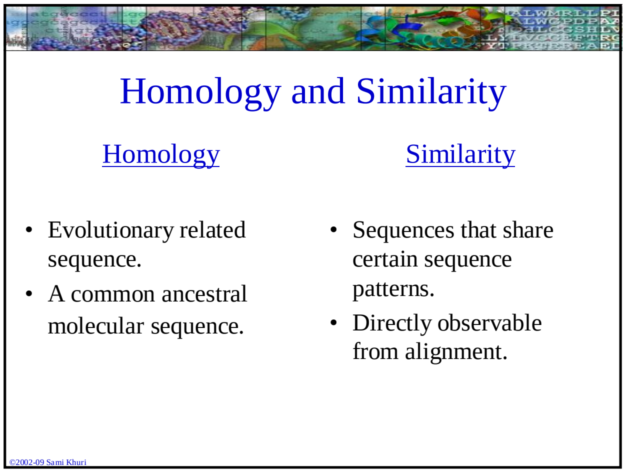
Similarity and Difference

- The similarity of two DNA sequences taken from different organisms can be explained by the theory that all contemporary genetic material has one common ancestral DNA.
- Differences between families of contemporary species resulted from mutations during the course of evolution.
  - Most of these changes are due to local mutations between nucleotide sequences.



When To Do The Pairwise Comparison?

- You have a strong suspicion that two sequences are homologues.
  - Two sequences are homologues, when they share a common ancestor.



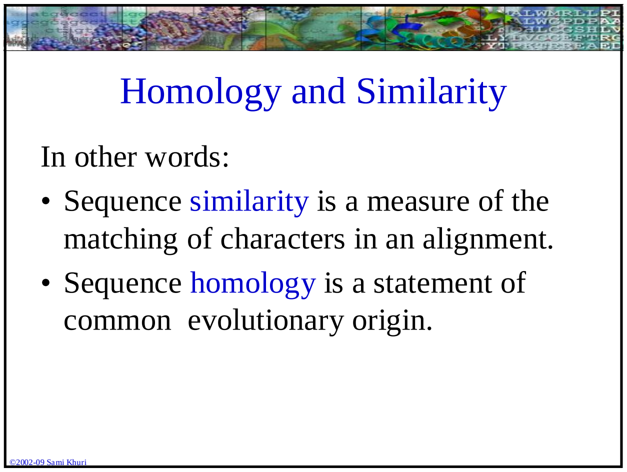
Homology and Similarity

Homology

- Evolutionary related sequence.
- A common ancestral molecular sequence.

Similarity

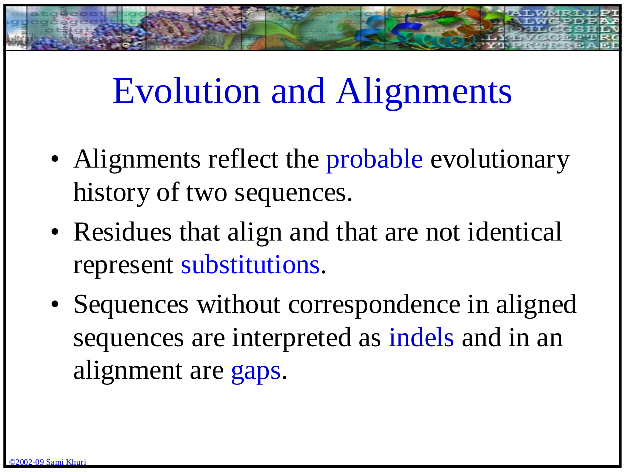
- Sequences that share certain sequence patterns.
- Directly observable from alignment.



Homology and Similarity

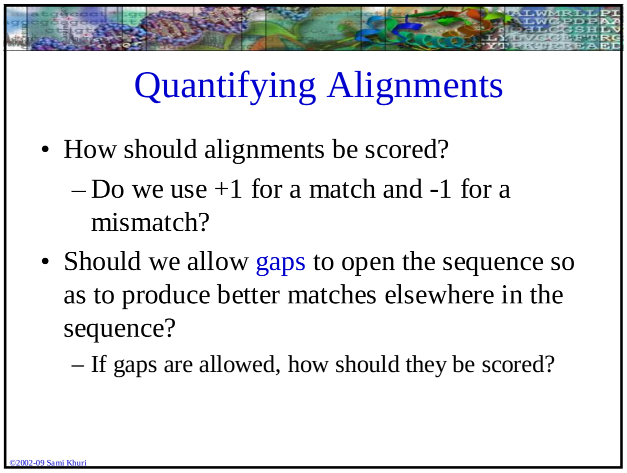
In other words:

- Sequence similarity is a measure of the matching of characters in an alignment.
- Sequence homology is a statement of common evolutionary origin.



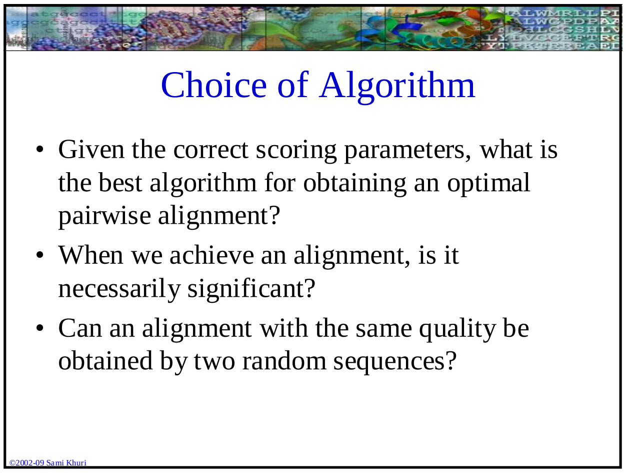
Evolution and Alignments

- Alignments reflect the probable evolutionary history of two sequences.
- Residues that align and that are not identical represent substitutions.
- Sequences without correspondence in aligned sequences are interpreted as indels and in an alignment are gaps.



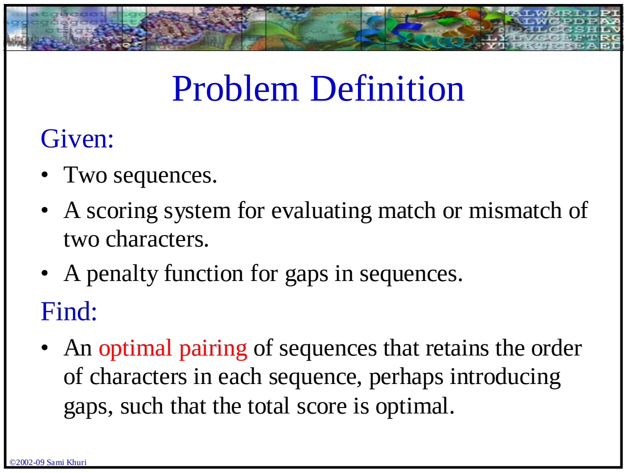
Quantifying Alignments

- How should alignments be scored?
  - Do we use +1 for a match and -1 for a mismatch?
- Should we allow gaps to open the sequence so as to produce better matches elsewhere in the sequence?
  - If gaps are allowed, how should they be scored?



Choice of Algorithm

- Given the correct scoring parameters, what is the best algorithm for obtaining an optimal pairwise alignment?
- When we achieve an alignment, is it necessarily significant?
- Can an alignment with the same quality be obtained by two random sequences?



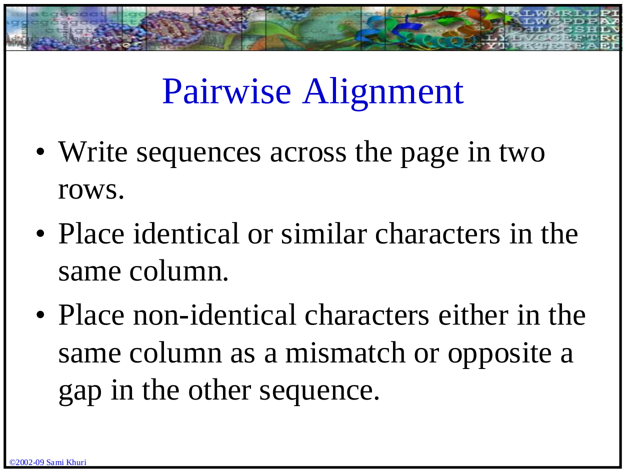
Problem Definition

Given:

- Two sequences.
- A scoring system for evaluating match or mismatch of two characters.
- A penalty function for gaps in sequences.

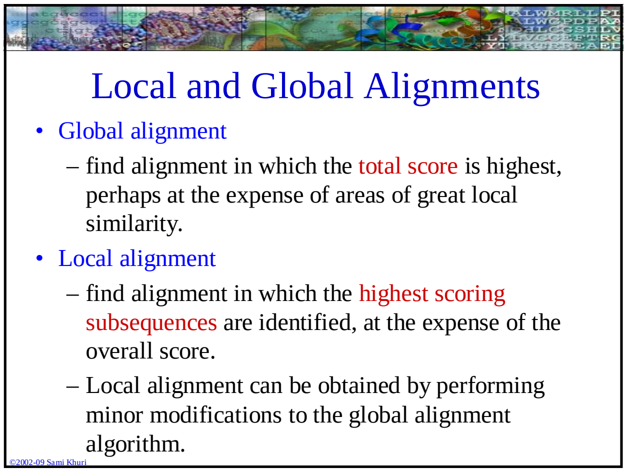
Find:

- An optimal pairing of sequences that retains the order of characters in each sequence, perhaps introducing gaps, such that the total score is optimal.



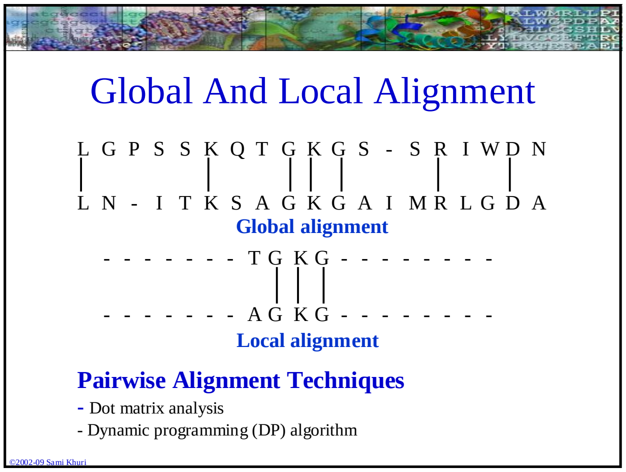
Pairwise Alignment

- Write sequences across the page in two rows.
- Place identical or similar characters in the same column.
- Place non-identical characters either in the same column as a mismatch or opposite a gap in the other sequence.



Local and Global Alignments

- Global alignment
  - find alignment in which the total score is highest, perhaps at the expense of areas of great local similarity.
- Local alignment
  - find alignment in which the highest scoring subsequences are identified, at the expense of the overall score.
  - Local alignment can be obtained by performing minor modifications to the global alignment algorithm.



Global And Local Alignment

```
L G P S S K Q T G K G S - S R I W D N
|         |       | | |       |     |
L N - I T K S A G K G A I M R L G D A
```

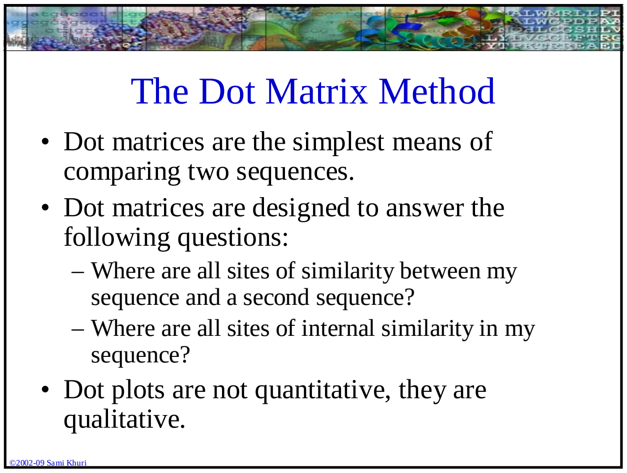
Global alignment

```
- - - - - - - T G K G - - - - - - - -
                | | |
- - - - - - - A G K G - - - - - - - -
```

Local alignment

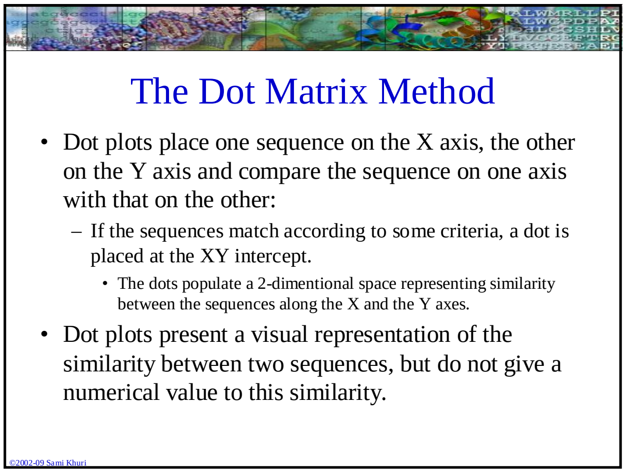
Pairwise Alignment Techniques

- Dot matrix analysis
- Dynamic programming (DP) algorithm



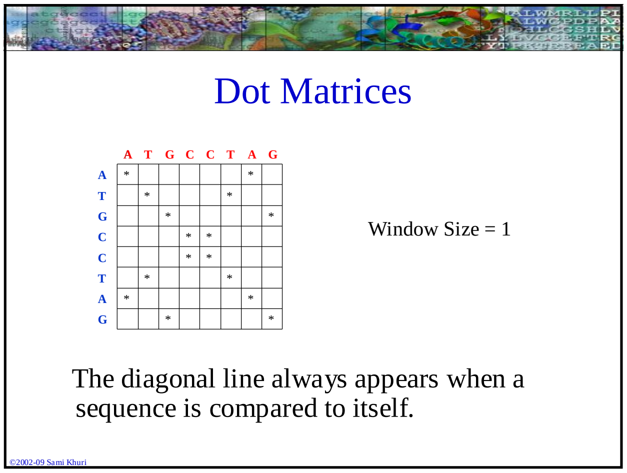
The Dot Matrix Method

- Dot matrices are the simplest means of comparing two sequences.
- Dot matrices are designed to answer the following questions:
  - Where are all sites of similarity between my sequence and a second sequence?
  - Where are all sites of internal similarity in my sequence?
- Dot plots are not quantitative, they are qualitative.



The Dot Matrix Method

- Dot plots place one sequence on the X axis, the other on the Y axis and compare the sequence on one axis with that on the other:
  - If the sequences match according to some criteria, a dot is placed at the XY intercept.
    - The dots populate a 2-dimentional space representing similarity between the sequences along the X and the Y axes.
- Dot plots present a visual representation of the similarity between two sequences, but do not give a numerical value to this similarity.

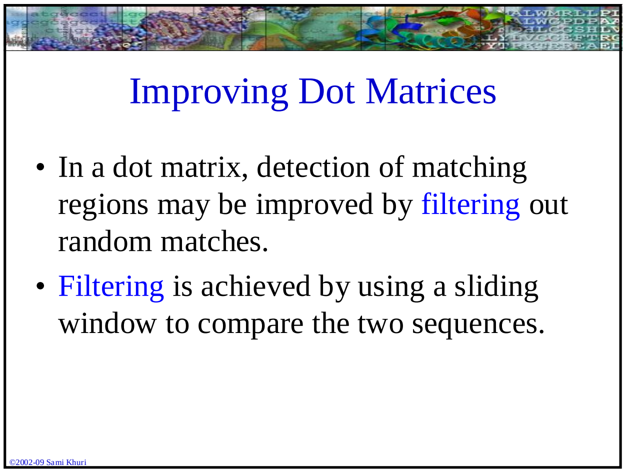


Dot Matrices

	A	T	G	C	C	T	A	G
A	*						*	
T		*				*		
G			*					*
C				*	*			
C				*	*			
T		*				*		
A	*						*	
G			*					*

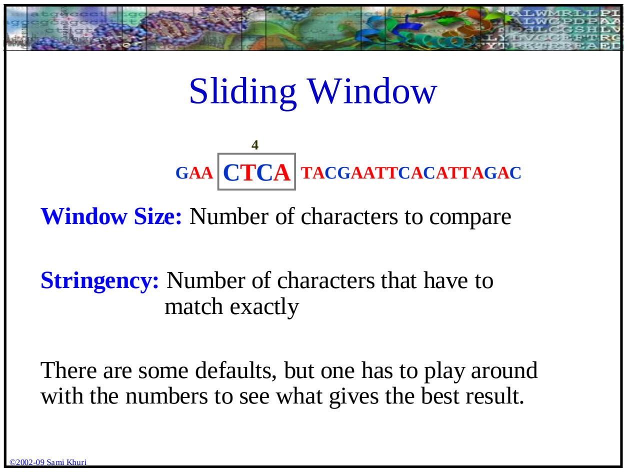
Window Size = 1

The diagonal line always appears when a sequence is compared to itself.



Improving Dot Matrices

- In a dot matrix, detection of matching regions may be improved by filtering out random matches.
- Filtering is achieved by using a sliding window to compare the two sequences.



Sliding Window

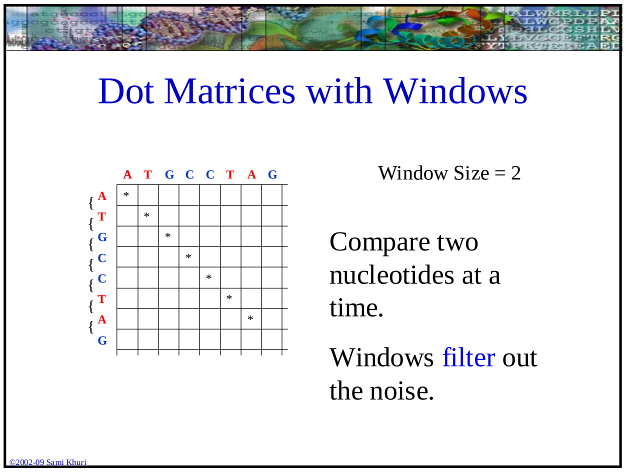
4

GAA CTCA TACGAATTCACATTAGAC

Window Size: Number of characters to compare

Stringency: Number of characters that have to match exactly

There are some defaults, but one has to play around with the numbers to see what gives the best result.



Dot Matrices with Windows

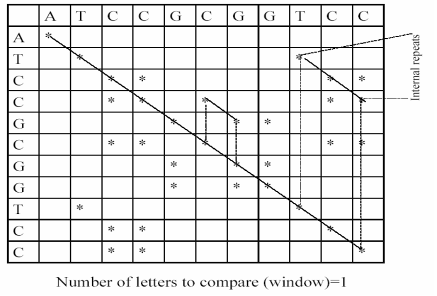
	A	T	G	C	C	T	A	G
{ A	*							
{ T		*						
{ G			*					
{ C				*				
{ C					*			
{ T						*		
{ A							*	
G								

Window Size = 2

Compare two nucleotides at a time.

Windows filter out the noise.

Internal Repeats



	A	T	C	C	G	C	G	G	T	C	C
A	*										
T		*							*		
C			*	*						*	*
C			*	*		*				*	*
G					*		*	*			
C			*	*		*				*	*
G					*		*	*			
G					*		*	*			
T		*							*		
C			*	*						*	
C			*	*							*

Number of letters to compare (window)=1

Determining The Window Size

A larger window size is generally used for DNA sequences than for protein sequences because:

the number of random matches is much larger due to the use of only four DNA symbols as compared to twenty amino acid symbols.

Determining The Window Size II

DNA Sequences

- A typical window size is15.
- A suitable match (stringency) requirement in this window is 10.

Protein Sequence

- Often the matrix is not filtered, but a window size requirement is 2 or 3.
- A match requirement of 2 will highlight matching regions.

Advantages of Dot Matrix

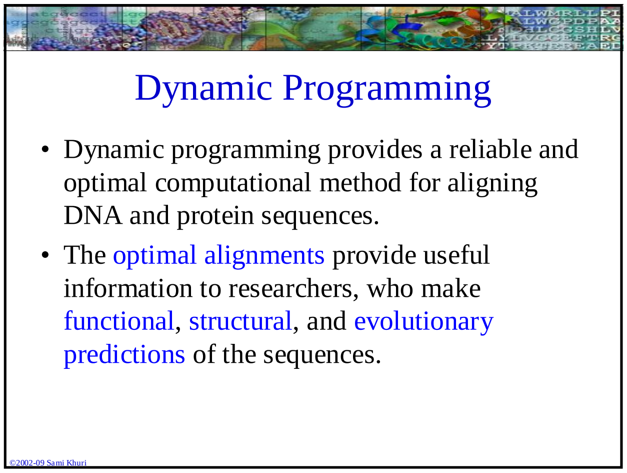
- All possible matches of residues between two sequences are found
  - The investigator now has the choice of identifying the most significant ones.
- The sequences of the actual regions that align can be detected by using one of two other methods for performing sequence alignments.
- The presence of insertions/deletions and direct and inverted repeats can be revealed.

Dot Matrices Shortcomings

- Most dot matrix computer programs do not show an actual alignment.
- Dot matrices rely on visual analysis.
- It is difficult to find optimal alignments.
  - We need scoring schemes more sophisticated than identical match.
- It is difficult to estimate the significance of alignments.
- Dot matrices do not allow gaps in the sequence alignments.

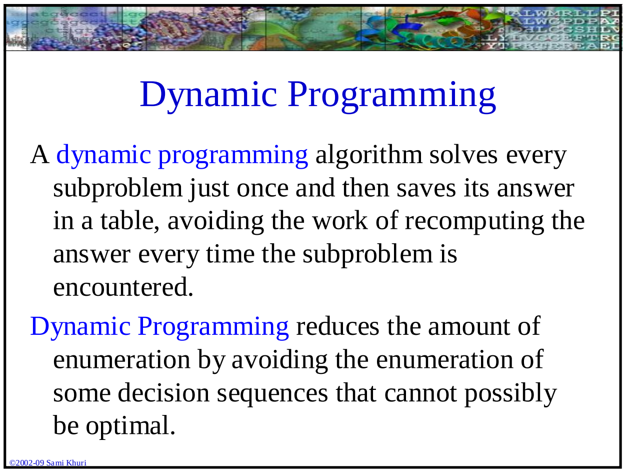
Other Applications of Dot Matrix

- Finding direct or inverted repeats in protein and DNA sequences.
- Predicting regions in RNA that are self-complementary and that have the potential of forming secondary structure.



Dynamic Programming

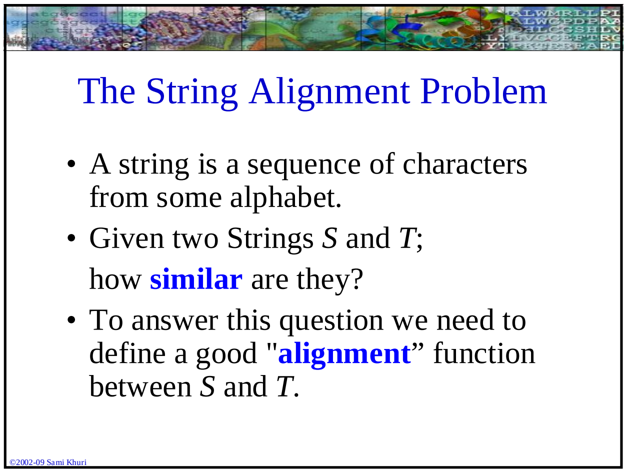
- Dynamic programming provides a reliable and optimal computational method for aligning DNA and protein sequences.
- The optimal alignments provide useful information to researchers, who make functional, structural, and evolutionary predictions of the sequences.



Dynamic Programming

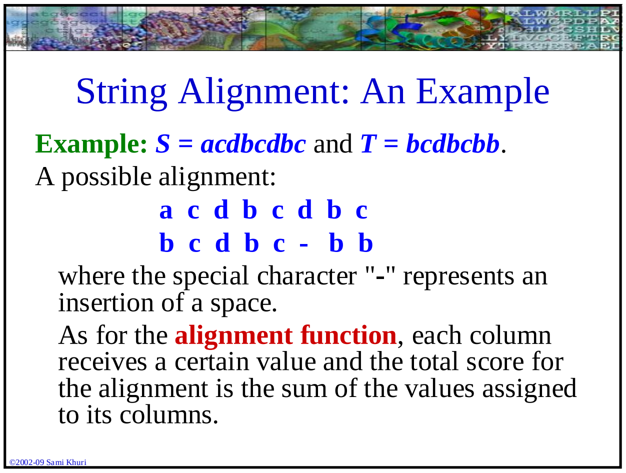
A dynamic programming algorithm solves every subproblem just once and then saves its answer in a table, avoiding the work of recomputing the answer every time the subproblem is encountered.

Dynamic Programming reduces the amount of enumeration by avoiding the enumeration of some decision sequences that cannot possibly be optimal.



The String Alignment Problem

- A string is a sequence of characters from some alphabet.
- Given two Strings *S* and *T*;
  how **similar** are they?
- To answer this question we need to define a good "**alignment**" function between *S* and *T*.



String Alignment: An Example

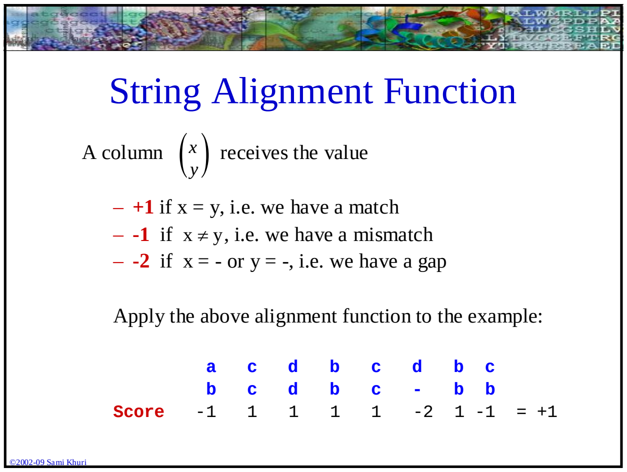
Example: ***S* = *acdbcdbc*** and ***T* = *bcdbcbb***.

A possible alignment:

```
a c d b c d b c
b c d b c - b b
```

where the special character "-" represents an insertion of a space.

As for the **alignment function**, each column receives a certain value and the total score for the alignment is the sum of the values assigned to its columns.



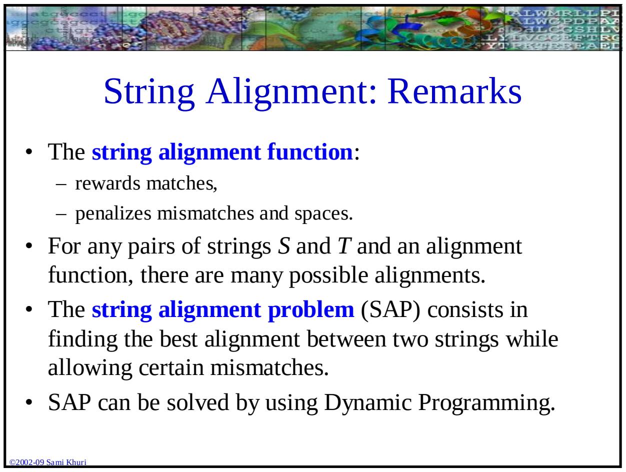
String Alignment Function

A column $\begin{pmatrix}x\\y\end{pmatrix}$ receives the value

- **+1** if x = y, i.e. we have a match
- **-1** if x ≠ y, i.e. we have a mismatch
- **-2** if x = - or y = -, i.e. we have a gap

Apply the above alignment function to the example:

```
         a    c    d    b    c    d    b   c
         b    c    d    b    c    -    b   b
Score   -1    1    1    1    1   -2    1  -1   = +1
```



String Alignment: Remarks

- The **string alignment function**:
  - rewards matches,
  - penalizes mismatches and spaces.
- For any pairs of strings *S* and *T* and an alignment function, there are many possible alignments.
- The **string alignment problem** (SAP) consists in finding the best alignment between two strings while allowing certain mismatches.
- SAP can be solved by using Dynamic Programming.

String Alignment Problem and DP

DP solves an instance of the String Alignment Problem by taking advantage of already computed solutions for smaller instances of the same problem.

- Given two sequences, *S* and *T*, instead of determining the similarity between *S* and *T* as whole sequences only, DP builds up the final solution by determining all similarities between arbitrary prefixes of *S* and *T*.
- DP starts with shorter prefixes and uses previously computed results to solve the problem for large prefixes until it finally finds the solution for *S* and *T*.

SAP: Optimal Alignments

- Use the alignment function previously seen.
- Given two strings *S* and *T* over some alphabet, with |*S*| = *n* and |*T*| = *m*.

  Define *a*(*i*, *j*) to be the value of an **optimal alignment** of strings:

  S[1], *S*[2], …, *S*[i] and

  T[1], *T*[2], ……, *T*[j]

  a(n, m) is the value of an optimal alignment of S and T.

SAP: Basis Relation

- The dynamic programming algorithm will compute each a(*i*, *j*), 0 ≤ *i* ≤ *n* and 0 ≤ *j* ≤ *m*, only **once**, by considering the values already computed for smaller indexes *i* and *j*.
- Define

$$a(i,0)=\sum_{k=1}^{i} p(S[k],-)$$

and

$$a(0,j)=\sum_{\mathrm{k=1}}^{\mathrm{j}} p(-,T[k])$$

where *p* is the alignment function.
a(*i*,0) means that the first *i* characters of *S* are aligned with no characters of *T*. In other words, the *i* characters of *S* are matched with *i* spaces (i.e. "-"). Similarly for *a*(0, *j*).

SAP: Recurrence Relation

In general:

$$a(i,j)=\max\begin{cases} a(i-1,\ j-1)+p(S[i],T[j]) \\ a(i-1,\ j)+p(S[i],-) \\ a(i,\ j-1)+p(-,T[j]) \end{cases}$$

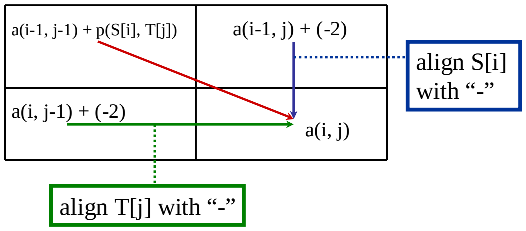
Recall: $p(S[i],T[j])=\begin{cases} +1 \text{ if } S[i]=T[j] \\ -1 \text{ if } S[i]\neq T[j] \end{cases}$

and *p*(*S*[*i*], -) = -2
p(-, *T*[*j*]) = -2.

SAP: Computing a(n,m)

- DP uses a table of size (n+1) × (m+1).
- *a*(*i*, *j*) corresponds to the optimal alignment of the i^{th} prefix of *S* with the j^{th} prefix of *T*.
- The dynamic programming algorithm fills in the entries of the table (matrix) by computing the values of *a*(*i*, *j*) from top to bottom, left to right.
- The value of the optimal alignment is given by *a*(*n*,*m*).

Filling Entry a(i,j) in the Table



a(i-1, j-1) + p(S[i], T[j])

a(i-1, j) + (-2)

align S[i] with "-"

a(i, j-1) + (-2)

a(i, j)

align T[j] with "-"

DP: Bookkeeping and Retracing

- Draw lines crossing the entries in the matrix to show from which entry in the matrix we derived the maximum score for each entry a(i, j).
- To determine the solution of the optimal alignment, simply retrace the steps from entry a(*n*, *m*) to entry a(0, 0).

Global Alignment

Example

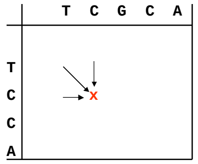
Align the following sequences:

S = TCCA
T = TCGCA

Solution

Use Needleman Wunsch Algorithm

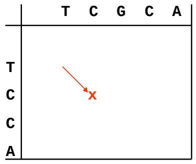
Three Possible Paths



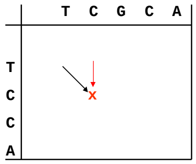
Any given point in the matrix can be reached from three possible positions.

=> Best scoring alignment ending in any given point in the matrix can be found by choosing the highest scoring of the three possibilities.

Computing the Score (I)

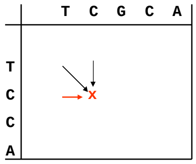


$$a(i,j) = \max \begin{cases} \mathbf{a(i-1,j-1) + p(i,j)} \\ \end{cases}$$



$$a(i,j) = \max \begin{cases} a(i-1,j-1) + p(i,j) \\ \mathbf{a(i-1,j) – (gap\_penalty)} \end{cases}$$

Computing the Score (II)

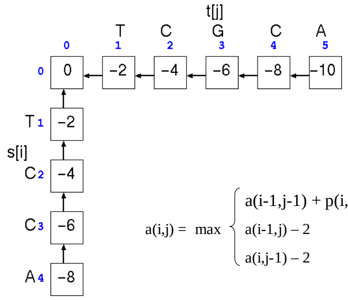


$$a(i,j) = \max \begin{cases} a(i-1,j-1) + p(i,j) \\ a(i-1,j) – (gap\_penalty) \\ \mathbf{a(i,j-1) – (gap\_penalty)} \end{cases}$$

Each new score is found by choosing the maximum of three possibilities. For each square in the matrix: keep track of where the best score came from.

Fill in scores one row at a time, starting in upper left corner of matrix, ending in lower right corner.

Needleman Wunsch: Example (I)



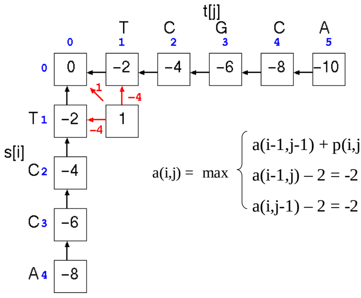
$$a(i,j) = \max \begin{cases} a(i-1,j-1) + p(i,j) \\ a(i-1,j) – 2 \\ a(i,j-1) – 2 \end{cases}$$

Penalty Function

	A	C	G	T
A	1	-1	-1	-1
C	-1	1	-1	-1
G	-1	-1	1	-1
T	-1	-1	-1	1

Gaps: -2

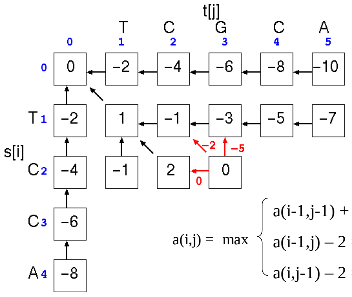
Needleman Wunsch: Example (II)



			T	C	G	C	A
		0	1	2	3	4	5
	0	0	−2	−4	−6	−8	−10
T	1	−2	1				
C	2	−4					
C	3	−6					
A	4	−8					

$$a(i,j) = \max \begin{cases} a(i-1,j-1) + p(i,j) = 0 + 1 \\ a(i-1,j) - 2 = -2 -2 \\ a(i,j-1) - 2 = -2 -2 \end{cases}$$

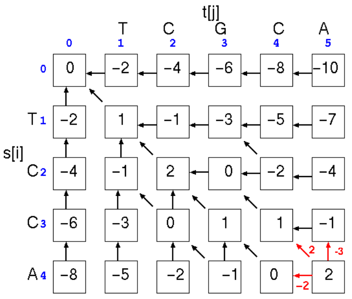
Needleman Wunsch: Example (III)



			T	C	G	C	A
		0	1	2	3	4	5
	0	0	−2	−4	−6	−8	−10
T	1	−2	1	−1	−3	−5	−7
C	2	−4	−1	2	0		
C	3	−6					
A	4	−8					

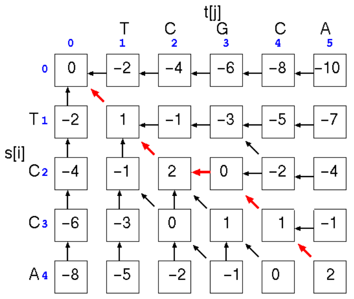
$$a(i,j) = \max \begin{cases} a(i-1,j-1) + p(i,j) = -1 + (-1) \\ a(i-1,j) - 2 = -3 + (-2) \\ a(i,j-1) - 2 = 2 + (-2) \end{cases}$$

Needleman Wunsch: Example (IV)



			T	C	G	C	A
		0	1	2	3	4	5
	0	0	−2	−4	−6	−8	−10
T	1	−2	1	−1	−3	−5	−7
C	2	−4	−1	2	0	−2	−4
C	3	−6	−3	0	1	1	−1
A	4	−8	−5	−2	−1	0	2

Needleman Wunsch: Example (V)



			T	C	G	C	A
		0	1	2	3	4	5
	0	0	−2	−4	−6	−8	−10
T	1	−2	1	−1	−3	−5	−7
C	2	−4	−1	2	0	−2	−4
C	3	−6	−3	0	1	1	−1
A	4	−8	−5	−2	−1	0	2

Solution:

```
T C - C A
: :   : :
T C G C A
1+1-2+1+1 = 2
```

Drawback of the DP for SAP

- The major drawback of dynamic programming is the fact that the table of size (n+1)×(m+1) uses *O*(*nm*) space.
- It is easy to compute a(*n*, *m*) in linear space since all we have to do at any given time during the computation is save two rows of the matrix, not more.
- The only values needed when computing *a*(*i*, *j*) are found in rows *i* and *i*-1.
- But it is not easy to find the optimal alignment in linear space.

Sub-Optimal Alignment

- The best alignment from a biological point of view, may not be the best alignment from a computational point of view.
- The ultimate goal is to align **functional** regions.
- The software can only align regions of sequence **similarity**.
- Sub-optimal alignments may not have the best sequence alignment, but may have helical regions or active sites aligned better than the “optimal” alignment.

Global Alignment

- The **dynamic programming** method we studied so far was designed by Needleman and Wunsch (1970).
- Their dynamic algorithm gives a **global alignment** of sequences.
- We now turn our attention to **local alignments**.

Local Alignment

- A modification of the dynamic programming algorithm for sequence alignment provides a **local sequence alignment** giving the highest-scoring local match between two sequences (Smith and Waterman 1981).
- Local alignments are usually more meaningful than global matches because they include patterns that are conserved in the sequences.

Local Alignment II

- The rules for calculating scoring values are slightly different with local alignment.
- The most important difference being:
  - Recall that the scoring system must include negative scores for mismatches
- With local alignment, when a dynamic programming scoring matrix value becomes negative, that value is set to zero, which has the effect of terminating any alignment up to that point.

Global and Local Alignments

- **Global Alignment**:
  - Are these two sequences generally the same?
- **Local Alignment**:
  - Do these two sequences contain high scoring subsequences?
- Local similarities may occur in sequences with different structure or function that share common substructure or subfunction.

Local Alignments

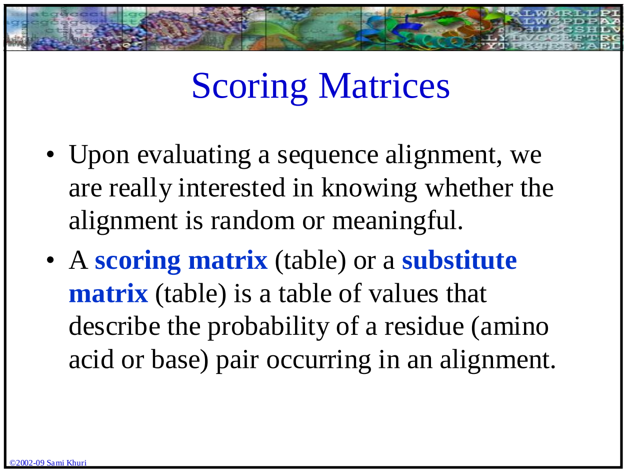
		G	A	A	C	G	T	A	G	G	C	G	T	A	T
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	1	1	0	0	0	1	0	0	0	0	0	1	0
T	0	0	0	0	0	0	1	0	0	0	0	0	1	0	2
A	0	0	1	1	0	0	0	2	0	0	0	0	0	2	0
C	0	0	0	0	2	0	0	0	1	0	1	0	0	0	1
T	0	0	0	0	0	1	1	0	0	0	0	0	1	0	1
A	0	0	1	1	0	0	0	2	0	0	0	0	0	2	0
C	0	0	0	0	2	0	0	0	0	0	1	0	0	0	0
G	0	1	0	0	0	3	1	0	1	1	0	2	0	0	0
G	0	1	0	0	0	1	2	0	0	2	0	1	1	0	0
A	0	0	2	1	0	0	0	3	1	0	0	0	0	2	0
G	0	1	0	0	0	1	0	0	4	2	0	1	0	0	0
G	0	1	0	0	0	1	0	0	1	5	3	1	0	0	0
G	0	1	0	0	0	1	0	0	1	2	4	4	2	0	0

Thus, the best local alignment achieved from the above Dynamic Programming is:

```
A C G G A G G
A C G T A G G
```

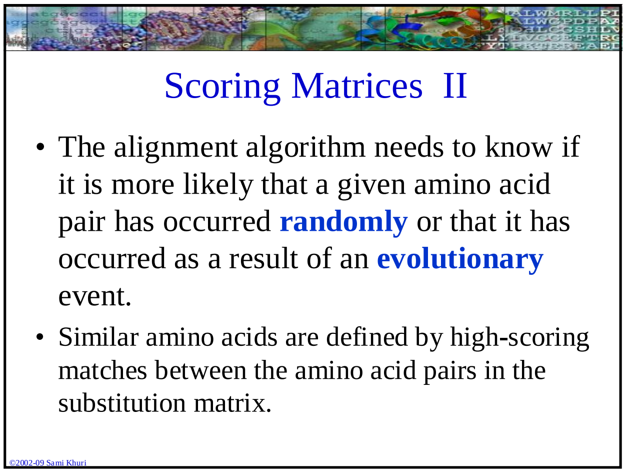
Scoring Systems

- Use of the **dynamic programming** method requires a scoring system for
  - the comparison of symbol pairs (**nucleotides** for DNA sequences & **amino acids** for protein sequences),
  - a scheme for insertion/deletion (gap) penalties.
- The most commonly used scoring systems for protein sequence alignments are the log odds form
  - of the **PAM250** matrix and
  - the **BLOSUM62** matrix.
- A number of other choices are available.



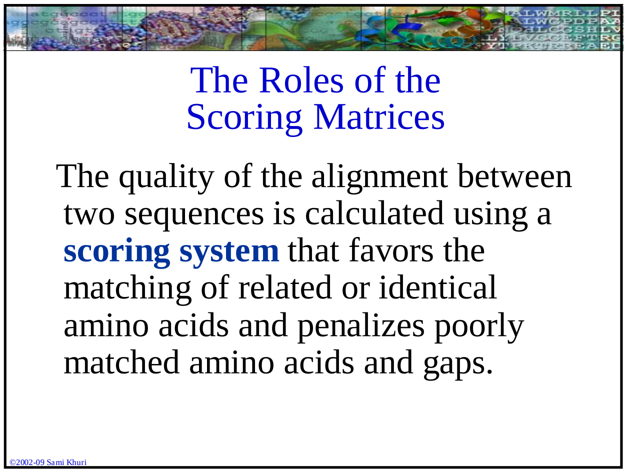
Scoring Matrices

- Upon evaluating a sequence alignment, we are really interested in knowing whether the alignment is random or meaningful.
- A **scoring matrix** (table) or a **substitute matrix** (table) is a table of values that describe the probability of a residue (amino acid or base) pair occurring in an alignment.



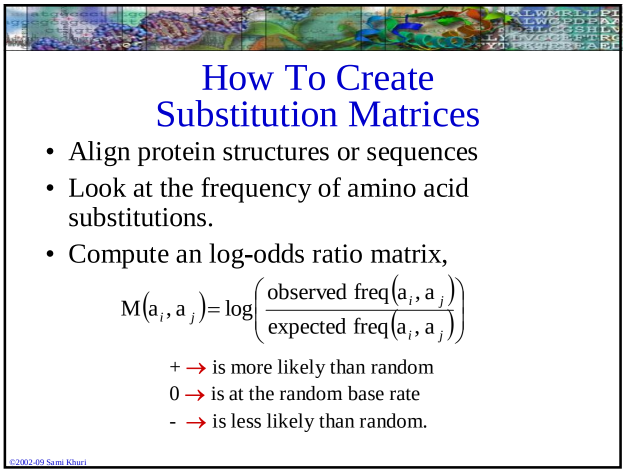
Scoring Matrices II

- The alignment algorithm needs to know if it is more likely that a given amino acid pair has occurred **randomly** or that it has occurred as a result of an **evolutionary** event.
- Similar amino acids are defined by high-scoring matches between the amino acid pairs in the substitution matrix.



The Roles of the Scoring Matrices

The quality of the alignment between two sequences is calculated using a **scoring system** that favors the matching of related or identical amino acids and penalizes poorly matched amino acids and gaps.

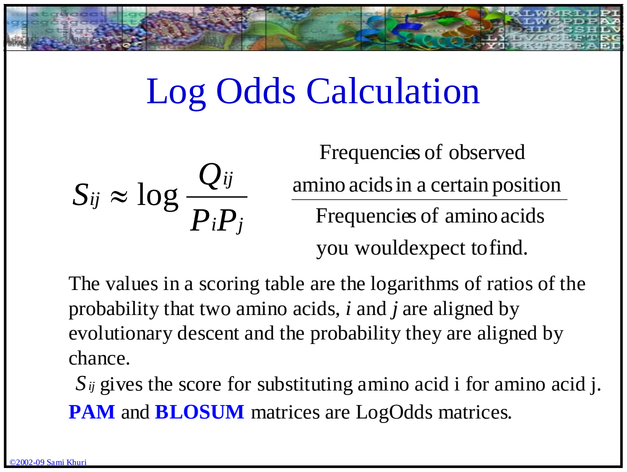


How To Create Substitution Matrices

- Align protein structures or sequences
- Look at the frequency of amino acid substitutions.
- Compute an log-odds ratio matrix,

$$M(a_i, a_j) = \log\left(\frac{\text{observed freq}(a_i, a_j)}{\text{expected freq}(a_i, a_j)}\right)$$

+ → is more likely than random
0 → is at the random base rate
- → is less likely than random.



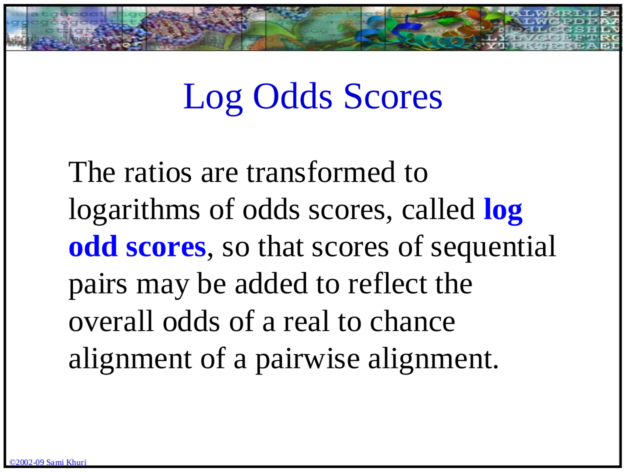
Log Odds Calculation

$$S_{ij} \approx \log \frac{Q_{ij}}{P_i P_j} \qquad \frac{\text{Frequencies of observed amino acids in a certain position}}{\text{Frequencies of amino acids you wouldexpect tofind.}}$$

The values in a scoring table are the logarithms of ratios of the probability that two amino acids, *i* and *j* are aligned by evolutionary descent and the probability they are aligned by chance.

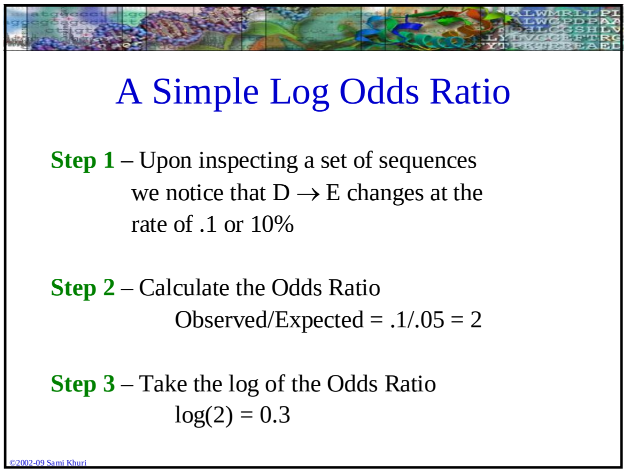
S_{ij} gives the score for substituting amino acid i for amino acid j.

PAM and **BLOSUM** matrices are LogOdds matrices.



Log Odds Scores

The ratios are transformed to logarithms of odds scores, called **log odd scores**, so that scores of sequential pairs may be added to reflect the overall odds of a real to chance alignment of a pairwise alignment.

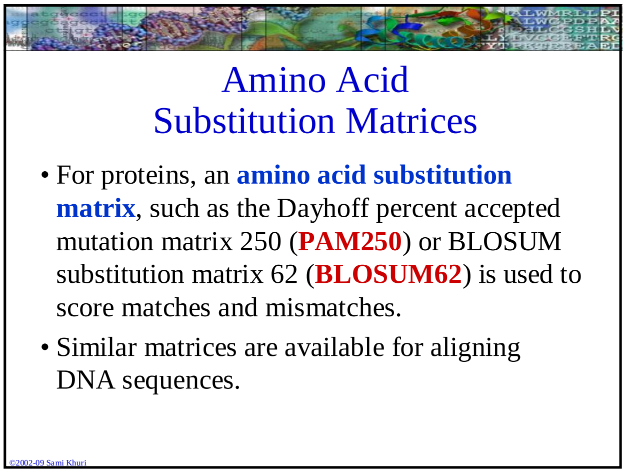


A Simple Log Odds Ratio

Step 1 – Upon inspecting a set of sequences we notice that D → E changes at the rate of .1 or 10%

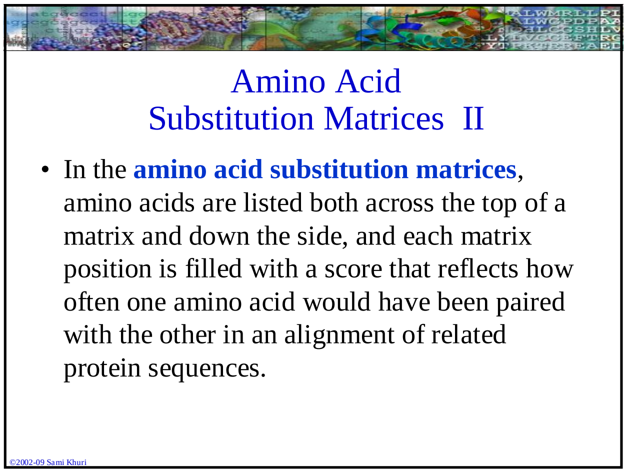
Step 2 – Calculate the Odds Ratio
Observed/Expected = .1/.05 = 2

Step 3 – Take the log of the Odds Ratio
log(2) = 0.3



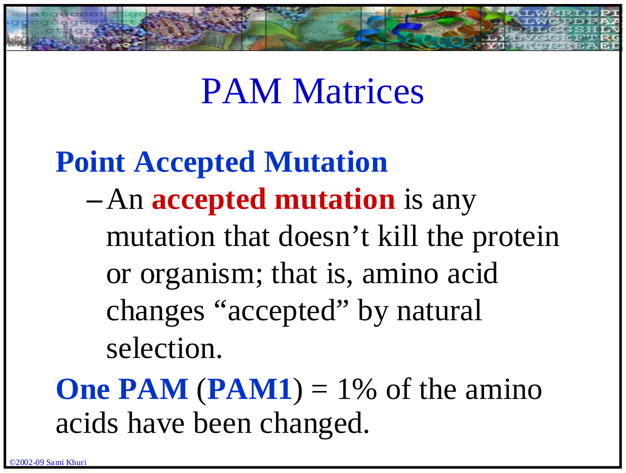
Amino Acid Substitution Matrices

- For proteins, an **amino acid substitution matrix**, such as the Dayhoff percent accepted mutation matrix 250 (**PAM250**) or BLOSUM substitution matrix 62 (**BLOSUM62**) is used to score matches and mismatches.
- Similar matrices are available for aligning DNA sequences.



Amino Acid Substitution Matrices II

- In the **amino acid substitution matrices**, amino acids are listed both across the top of a matrix and down the side, and each matrix position is filled with a score that reflects how often one amino acid would have been paired with the other in an alignment of related protein sequences.

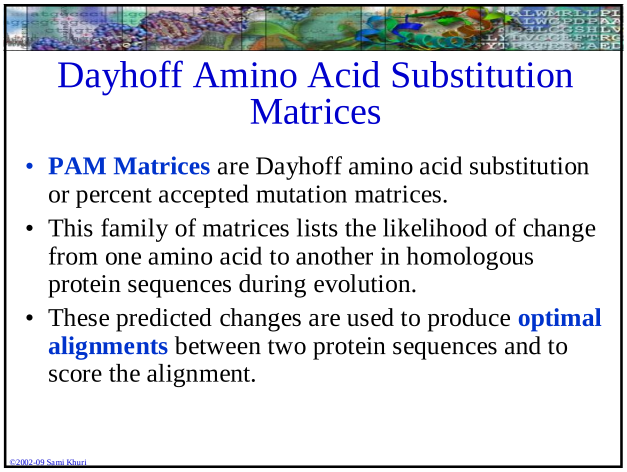


PAM Matrices

Point Accepted Mutation

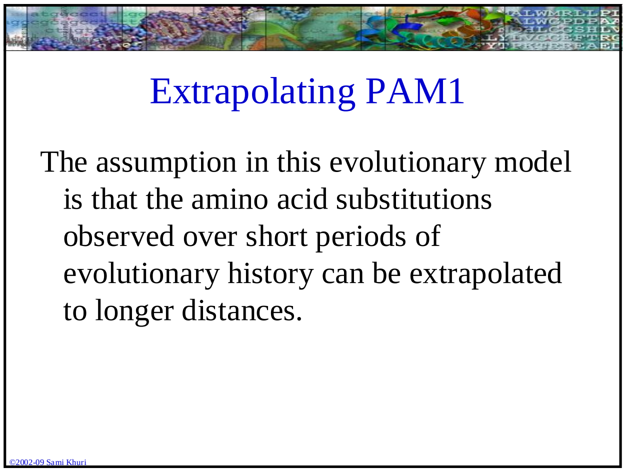
– An **accepted mutation** is any mutation that doesn't kill the protein or organism; that is, amino acid changes "accepted" by natural selection.

One PAM (**PAM1**) = 1% of the amino acids have been changed.



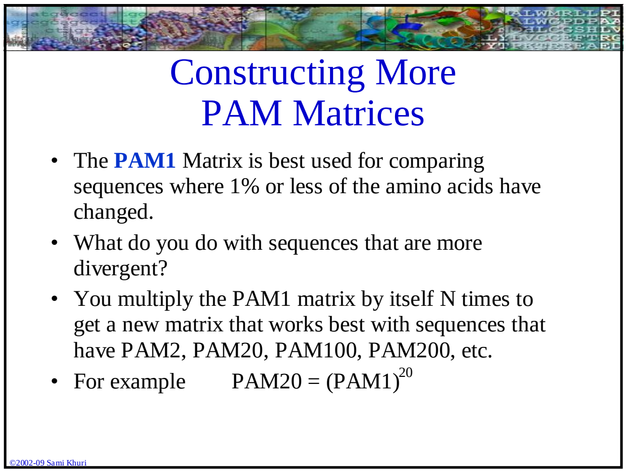
Dayhoff Amino Acid Substitution Matrices

- **PAM Matrices** are Dayhoff amino acid substitution or percent accepted mutation matrices.
- This family of matrices lists the likelihood of change from one amino acid to another in homologous protein sequences during evolution.
- These predicted changes are used to produce **optimal alignments** between two protein sequences and to score the alignment.



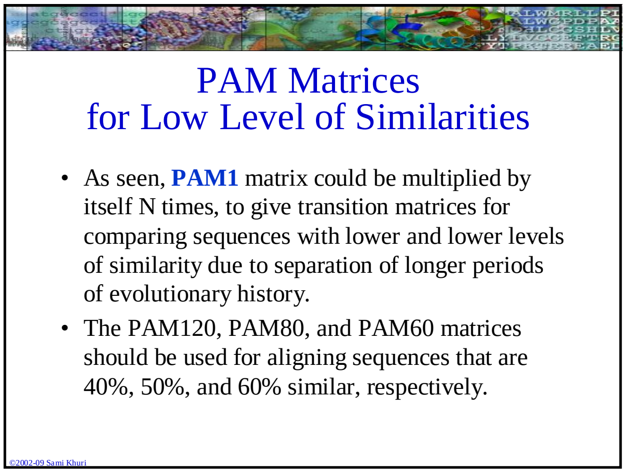
Extrapolating PAM1

The assumption in this evolutionary model is that the amino acid substitutions observed over short periods of evolutionary history can be extrapolated to longer distances.



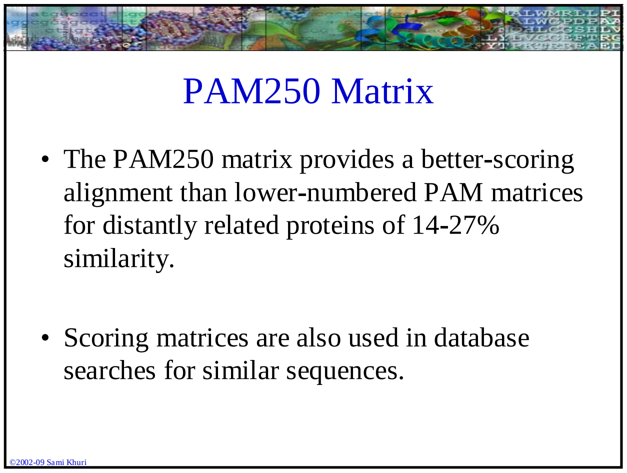
Constructing More PAM Matrices

- The **PAM1** Matrix is best used for comparing sequences where 1% or less of the amino acids have changed.
- What do you do with sequences that are more divergent?
- You multiply the PAM1 matrix by itself N times to get a new matrix that works best with sequences that have PAM2, PAM20, PAM100, PAM200, etc.
- For example PAM20 = $(\text{PAM1})^{20}$



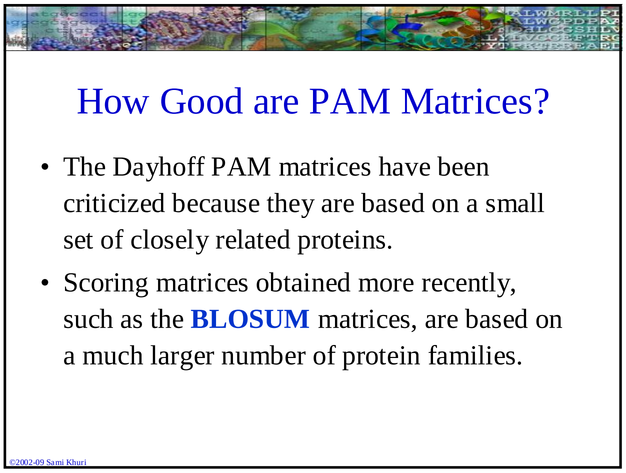
PAM Matrices for Low Level of Similarities

- As seen, **PAM1** matrix could be multiplied by itself N times, to give transition matrices for comparing sequences with lower and lower levels of similarity due to separation of longer periods of evolutionary history.
- The PAM120, PAM80, and PAM60 matrices should be used for aligning sequences that are 40%, 50%, and 60% similar, respectively.



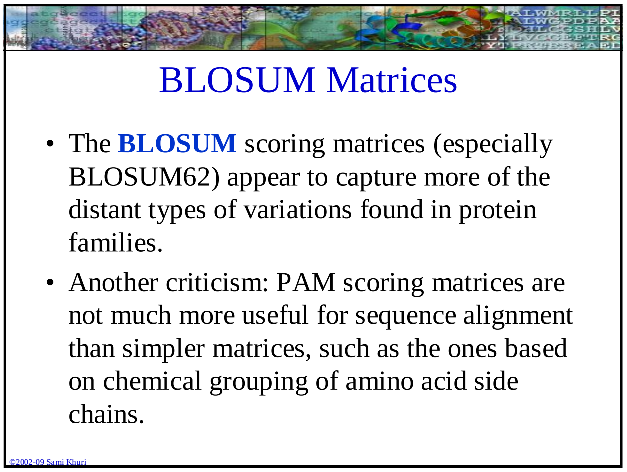
PAM250 Matrix

- The PAM250 matrix provides a better-scoring alignment than lower-numbered PAM matrices for distantly related proteins of 14-27% similarity.
- Scoring matrices are also used in database searches for similar sequences.



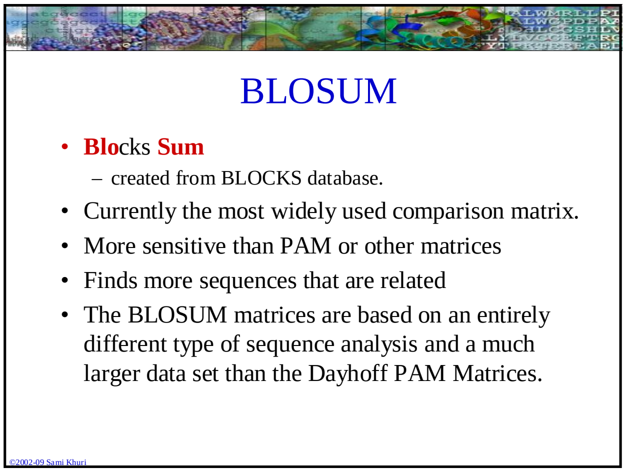
How Good are PAM Matrices?

- The Dayhoff PAM matrices have been criticized because they are based on a small set of closely related proteins.
- Scoring matrices obtained more recently, such as the **BLOSUM** matrices, are based on a much larger number of protein families.



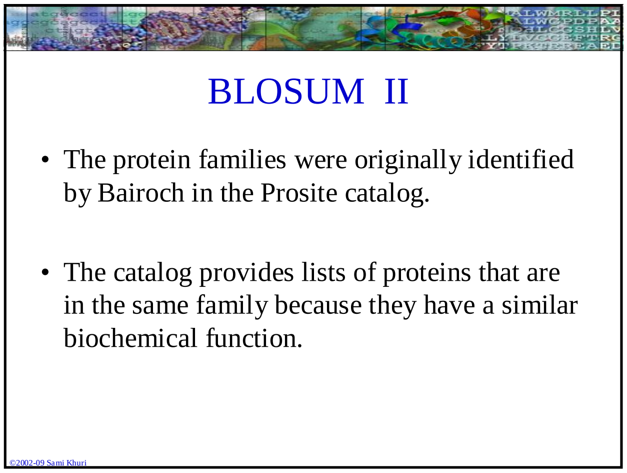
BLOSUM Matrices

- The **BLOSUM** scoring matrices (especially BLOSUM62) appear to capture more of the distant types of variations found in protein families.
- Another criticism: PAM scoring matrices are not much more useful for sequence alignment than simpler matrices, such as the ones based on chemical grouping of amino acid side chains.



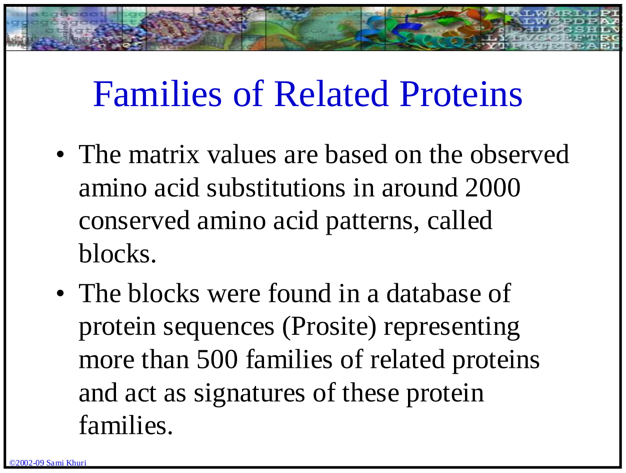
BLOSUM

- **Blo**cks **Sum**
  - created from BLOCKS database.
- Currently the most widely used comparison matrix.
- More sensitive than PAM or other matrices
- Finds more sequences that are related
- The BLOSUM matrices are based on an entirely different type of sequence analysis and a much larger data set than the Dayhoff PAM Matrices.



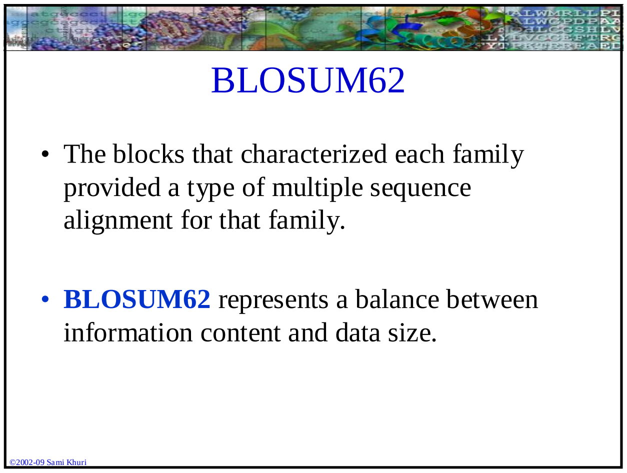
BLOSUM II

- The protein families were originally identified by Bairoch in the Prosite catalog.
- The catalog provides lists of proteins that are in the same family because they have a similar biochemical function.



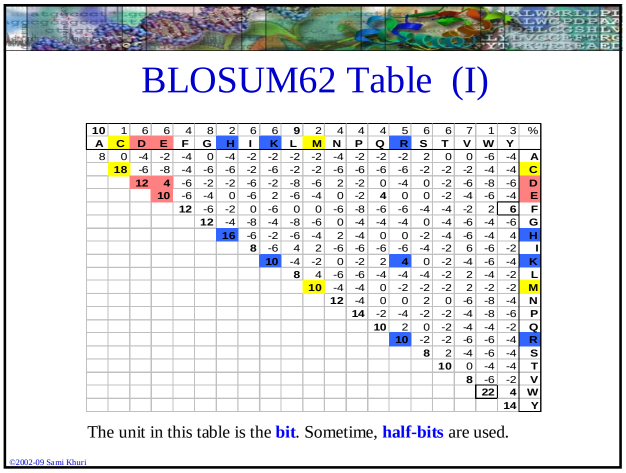
Families of Related Proteins

- The matrix values are based on the observed amino acid substitutions in around 2000 conserved amino acid patterns, called blocks.
- The blocks were found in a database of protein sequences (Prosite) representing more than 500 families of related proteins and act as signatures of these protein families.



BLOSUM62

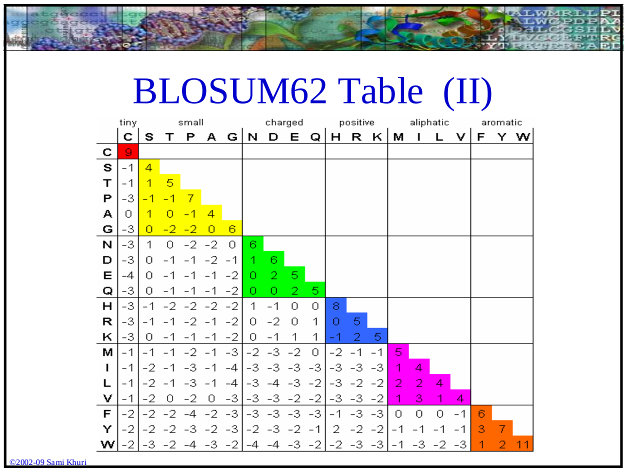
- The blocks that characterized each family provided a type of multiple sequence alignment for that family.
- **BLOSUM62** represents a balance between information content and data size.



BLOSUM62 Table (I)

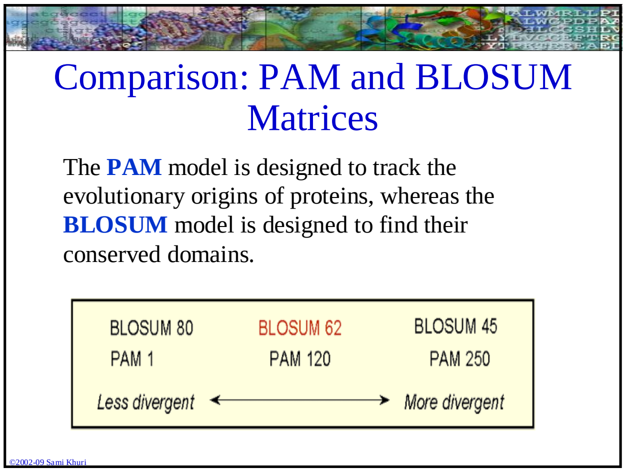
10	1	6	6	4	8	2	6	6	9	2	4	4	4	5	6	6	7	1	3	%
A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y	
8	0	-4	-2	-4	0	-4	-2	-2	-2	-2	-4	-2	-2	-2	2	0	0	-6	-4	A
	18	-6	-8	-4	-6	-6	-2	-6	-2	-2	-6	-6	-6	-6	-2	-2	-2	-4	-4	C
		12	4	-6	-2	-2	-6	-2	-8	-6	2	-2	0	-4	0	-2	-6	-8	-6	D
			10	-6	-4	0	-6	2	-6	-4	0	-2	4	0	0	-2	-4	-6	-4	E
				12	-6	-2	0	-6	0	0	-6	-8	-6	-6	-4	-4	-2	2	6	F
					12	-4	-8	-4	-8	-6	0	-4	-4	-4	0	-4	-6	-4	-6	G
						16	-6	-2	-6	-4	2	-4	0	0	-2	-4	-6	-4	4	H
							8	-6	4	2	-6	-6	-6	-6	-4	-2	6	-6	-2	I
								10	-4	-2	0	-2	2	4	0	-2	-4	-6	-4	K
									8	4	-6	-6	-4	-4	-4	-2	2	-4	-2	L
										10	-4	-4	0	-2	-2	-2	2	-2	-2	M
											12	-4	0	0	2	0	-6	-8	-4	N
												14	-2	-4	-2	-2	-4	-8	-6	P
													10	2	0	-2	-4	-4	-2	Q
														10	-2	-2	-6	-6	-4	R
															8	2	-4	-6	-4	S
																10	0	-4	-4	T
																	8	-6	-2	V
																		22	4	W
																			14	Y

The unit in this table is the **bit**. Sometime, **half-bits** are used.



BLOSUM62 Table (II)

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

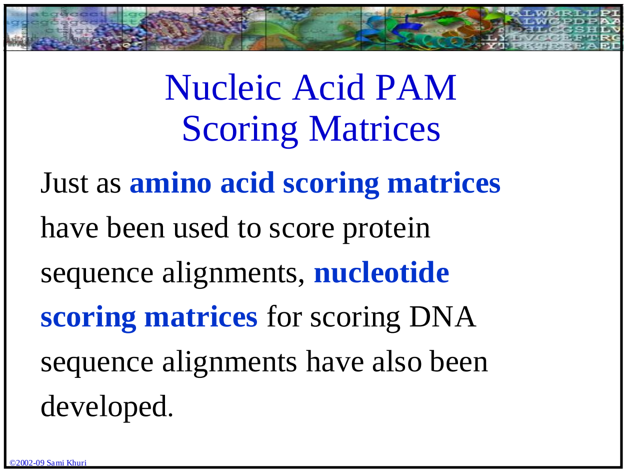


Comparison: PAM and BLOSUM Matrices

The **PAM** model is designed to track the evolutionary origins of proteins, whereas the **BLOSUM** model is designed to find their conserved domains.

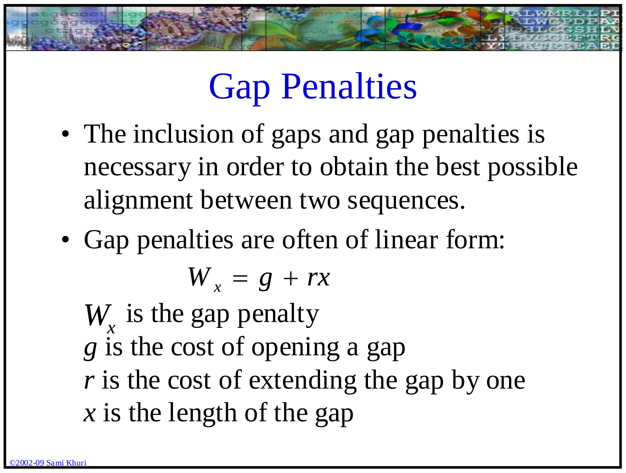
BLOSUM 80	BLOSUM 62	BLOSUM 45
PAM 1	PAM 120	PAM 250

Less divergent ⟵⟶ *More divergent*



Nucleic Acid PAM Scoring Matrices

Just as **amino acid scoring matrices** have been used to score protein sequence alignments, **nucleotide scoring matrices** for scoring DNA sequence alignments have also been developed.

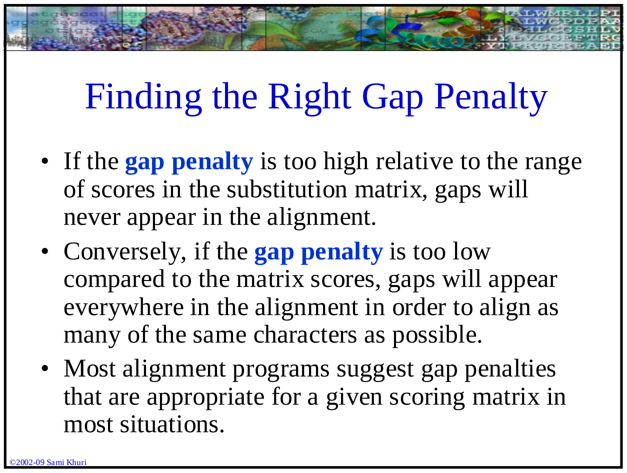


Gap Penalties

- The inclusion of gaps and gap penalties is necessary in order to obtain the best possible alignment between two sequences.
- Gap penalties are often of linear form:

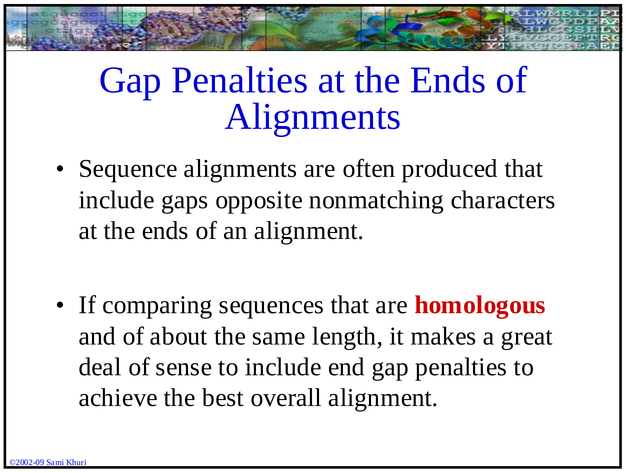
$$W_x = g + rx$$

W_x is the gap penalty
g is the cost of opening a gap
r is the cost of extending the gap by one
x is the length of the gap



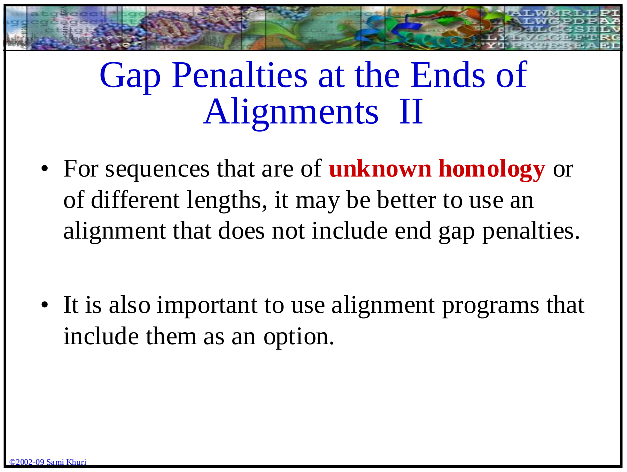
Finding the Right Gap Penalty

- If the **gap penalty** is too high relative to the range of scores in the substitution matrix, gaps will never appear in the alignment.
- Conversely, if the **gap penalty** is too low compared to the matrix scores, gaps will appear everywhere in the alignment in order to align as many of the same characters as possible.
- Most alignment programs suggest gap penalties that are appropriate for a given scoring matrix in most situations.



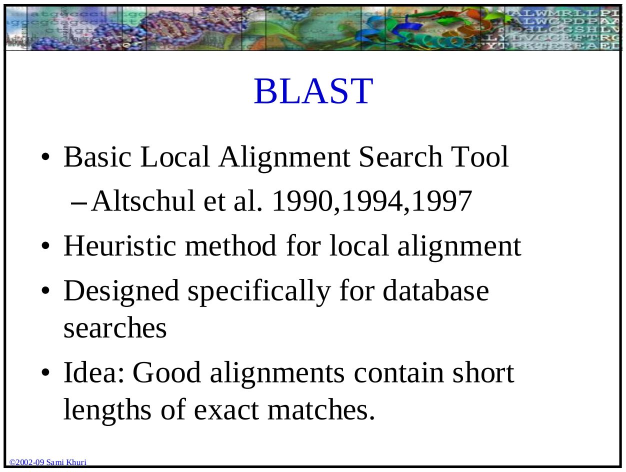
Gap Penalties at the Ends of Alignments

- Sequence alignments are often produced that include gaps opposite nonmatching characters at the ends of an alignment.
- If comparing sequences that are **homologous** and of about the same length, it makes a great deal of sense to include end gap penalties to achieve the best overall alignment.



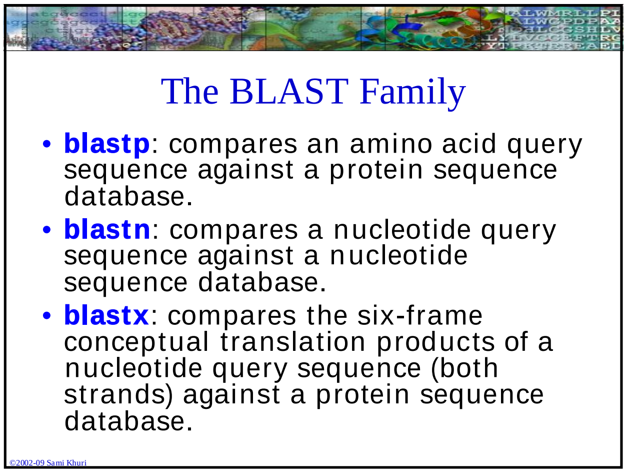
Gap Penalties at the Ends of Alignments II

- For sequences that are of **unknown homology** or of different lengths, it may be better to use an alignment that does not include end gap penalties.
- It is also important to use alignment programs that include them as an option.



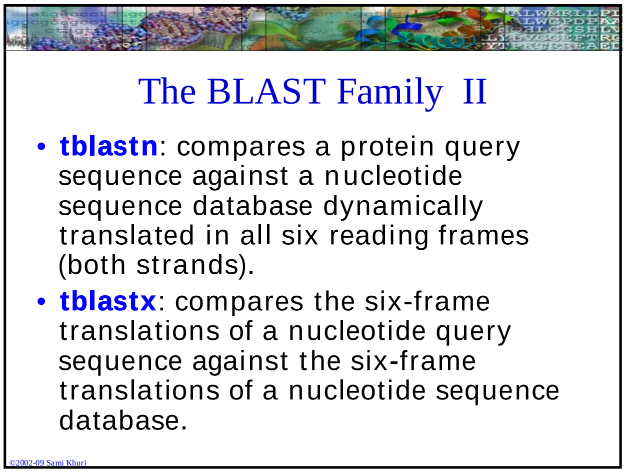
BLAST

- Basic Local Alignment Search Tool
  - Altschul et al. 1990,1994,1997
- Heuristic method for local alignment
- Designed specifically for database searches
- Idea: Good alignments contain short lengths of exact matches.



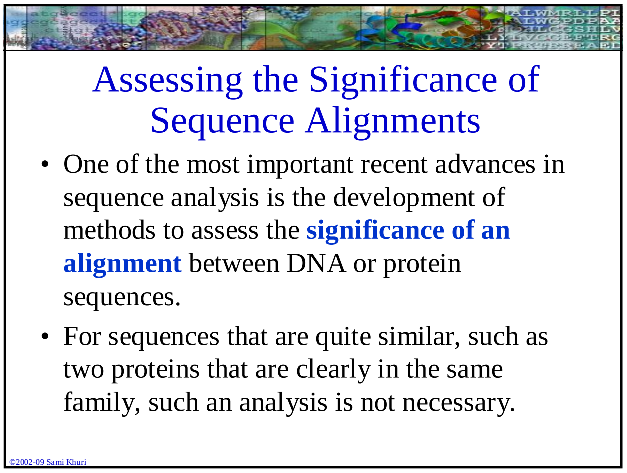
The BLAST Family

- **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 the six-frame conceptual translation products of a nucleotide query sequence (both strands) against a protein sequence database.



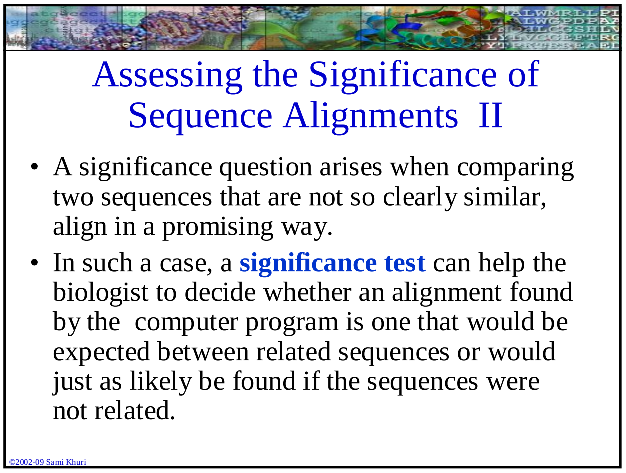
The BLAST Family II

- **tblastn**: compares a protein query sequence against a nucleotide sequence database dynamically translated in all six reading frames (both strands).
- **tblastx**: compares the six-frame translations of a nucleotide query sequence against the six-frame translations of a nucleotide sequence database.



Assessing the Significance of Sequence Alignments

- One of the most important recent advances in sequence analysis is the development of methods to assess the **significance of an alignment** between DNA or protein sequences.
- For sequences that are quite similar, such as two proteins that are clearly in the same family, such an analysis is not necessary.



Assessing the Significance of Sequence Alignments II

- A significance question arises when comparing two sequences that are not so clearly similar, align in a promising way.
- In such a case, a **significance test** can help the biologist to decide whether an alignment found by the computer program is one that would be expected between related sequences or would just as likely be found if the sequences were not related.