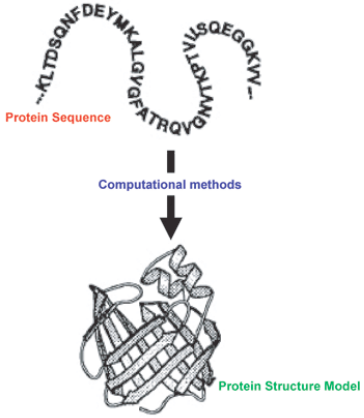
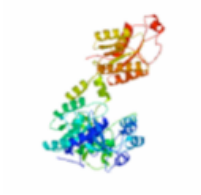
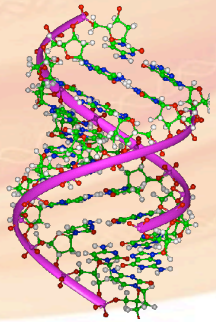




Bioinformatics Computational Methods 1 - BIOL 6308



November 14th 2013

http://155.33.203.128/cleslin/home/teaching6308F2013.php

Last Time

- Describing Features Using Frequency Matrices
- Position Specific Score Matrix (PSSM)
- Knowledge-Based Motif Identification
  - Consensus v.s. Patterns v.s. PSSMs
- Steps used to build a PSSM
- Advantages of PSSM
- How to Determine Significance in a PSSM
  - p-Values
  - The null Hypothesis
- Rescaling Values
- Approximation of the Distribution of Scores (Staden Technique)
- Prokaryotic Promoters

The Null Hypothesis

- When performing statistical hypothesis tests:
  - one-sample t-test or the Anderson-Darling test for normality
  - Investigator either:
    - ***reject*** *or* ***fail to reject*** *the null hypothesis*
  - ***But, should never accept the null hypothesis***
- More at stake than semantics when investigators use misleading language that implies "acceptance" of the null hypothesis

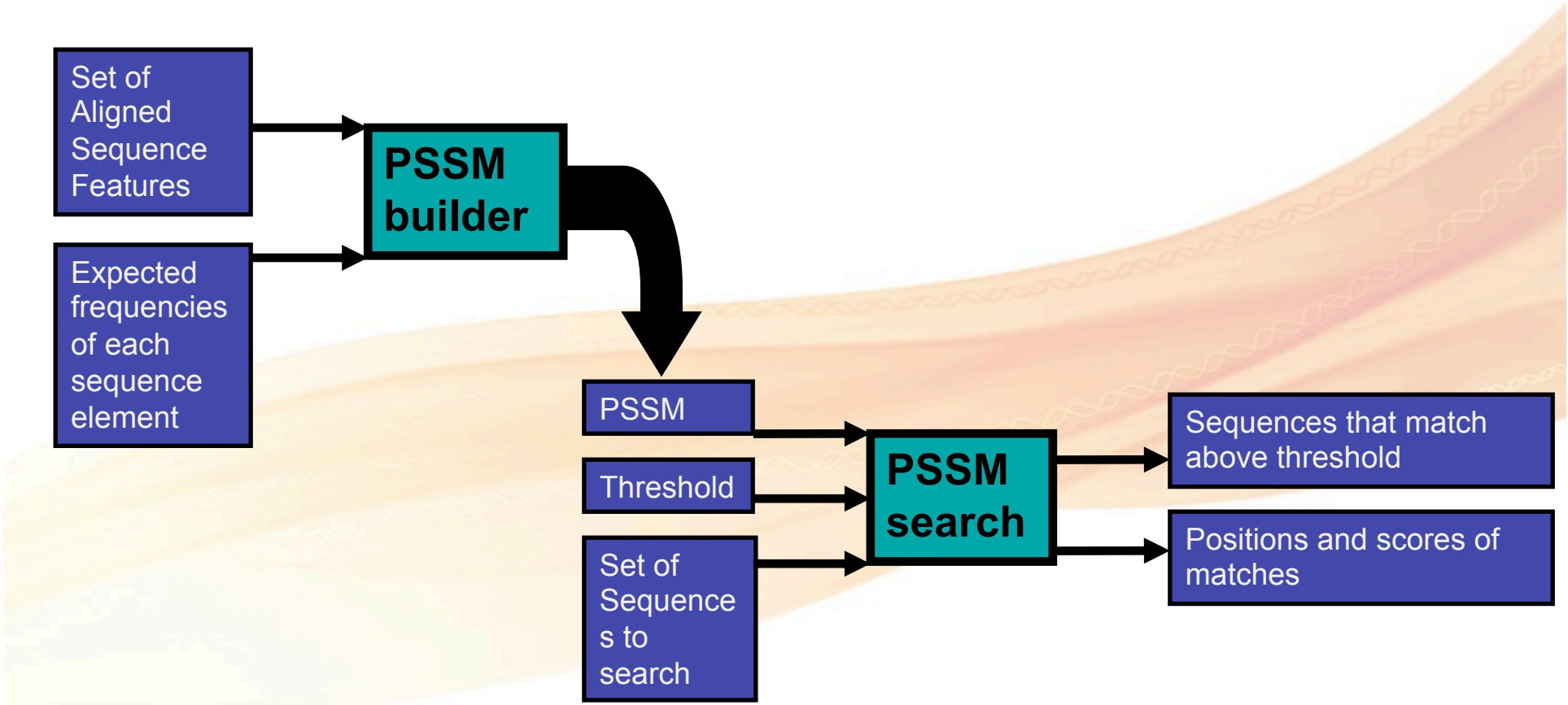
- **We Don't "Accept" the Null Hypothesis**

http://www.minitab.com/uploadedFiles/Shared_Resources/Documents/Articles/not_accepting_null_hypothesis.pdf

Adopted form Keith M. Bower

High Level View of How to use PSSM

Block Diagram for Searching for Sequences Related to a Family with a PSSM



Sequence Analysis

Intrinsic Sequence Analysis

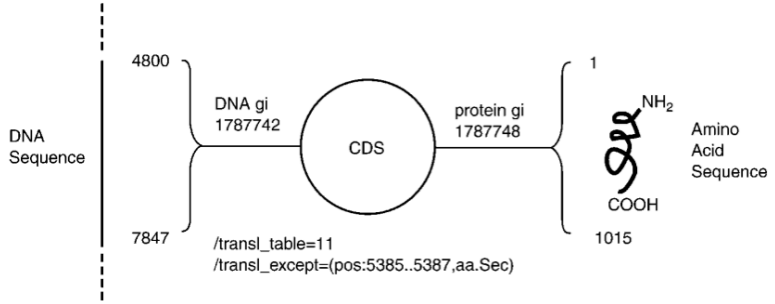
- **Intrinsic Sequence Analysis:**
  - Evaluating sequence properties without explicitly referring to other sequences
  - Using a derived formula, sequence or value
  - Sequences:
    - Includes calculating %G+C values (DNA), or
    - Comparing the sequence against itself for the presence of direct or inverted repeats

Sequence Analysis

- **Extrinsic Sequence Analysis:**
  - Evaluating sequence properties by explicitly comparing them to other sequences or sets of sequences
  - Sequences:
    - Includes sequence comparison, either:
      - Pairwise dot plots
      - Pairwise optimal (Needleman-Wunsch or Smith-Waterman)
      - Pairwise heuristic (BLAST, BLAT, FASTA)
      - Multiple sequence alignment to detect homology, insertions or deletions, as well as statistical or phylogenetic analysis
- **A common mistake in sequence analysis is to use only one of these approaches**
- Combining intrinsic and extrinsic computational approaches can give results neither alone can approach

Remember, if You Can Use Proteins

- The NCBI Data Model
  - the CDS feature link



- If the sequence of interest encodes a protein
  - Compare at the protein sequence level:
    - Many changes in DNA sequences do not change the encoded protein
    - As we've see substitution matrices for protein sequences represent biochemical information

Using Proteins for Searches

- **Protein** similarity searches are **more sensitive** than comparisons of DNA sequences
  - Alphabet
    - DNA contains only 4 lettersa
    - a.a. have 20 letters
      - **Probability of chance matches is much greater with DNA-DNA comparisons**
- DNA bases scored as a match or a mismatch
- a.a can share varying degrees of similarity, based on:
  - **Their physical and chemical properties**
  - **Similarity of DNA codons**
- Protein DBs much smaller than DNA DBs
  - So searches can be more sensitive without incurring too many **false positives**

Exact vs. Heuristic

- **Exact methods:** the result is guaranteed to be (mathematically) optimal
  - Pairwise alignment
    - Needleman-Wunsch (global)
    - Smith-Waterman (local)
  - Multiple alignment
    - Rarely used; quickly becomes computationally intractable with increasing number of sequences and/or their length
- **Heuristic methods:** make some assumptions that hold most of the time, but not all of the time
  - Pairwise alignment – BLAST
  - Multiple alignment – all widely used multiple alignment programs (ClustalW, Tcofee, Muscle, MAFFT…)

Optimal Alignments

- In the mathematical sense:
  - Provide the best or highest-scoring alignment for a given set of scoring functions
- In the biological sense:
  - Provide an alignment in which each aligned sequence residue descended from the same ancestral residue
  - Each aligned sequence residue plays the same functional role for the two proteins

- Finding optimal alignments is straightforward, determining homology is not!

A True Set of Related Sequences

- A total of approximately 2 billion years of evolution amongst these proteins

```
CLUSTAL W(1.4) multiple sequence alignment      * = identical in all
                                                . = conservative changes

worm CaMKII  EARICRKLQHPNIVRLHDSIQEESFHYLVFDLVTGGELFEDIVAREFYSEADASHCIQQI
fly  CaMKII  EARICRKLHHPNIVRLHDSIQEENYHYLVFDLVTGGELFEDIVAREFYSEADASHCIQQI
rat  CaMKII  EARICRLLKHPNIVRLHDSISEEGHHYLIFDLVTGGELFEDIVAREYYSEADASHCIQQI
             ****** *.*********** **  ***.*****************.**************

worm CaMKII  LESIAYCHSNGIVHRDLKPENLLLASKAKGAAVKLADFGLAIEVN-DSEAWHGFAGTPGY
fly  CaMKII  LESVNHCHQNGVVHRDLKPENLLLASKAKGAAVKLADFGLAIEVQGDHQAWFGFAGTPGY
rat  CaMKII  LEAVLHCHQMGVVHRDLKPENLLLASKLKGAAVKLADFGLAIEVEGEQQRWFGFAGTPGY
             **..  **  *.*************** ****************  . . * ********
```

A Dubious Case

- Are these sequences evolutionarily related?

```
CLUSTAL W(1.4) multiple sequence alignment

ATTSTDGLISNGAERLRLQGSRLQTSRFACFRCCGNIITYLVRLRSTPEELEQRYKSKEI
EARICRKLHHPNIVRLHDSIQEENYHYLVFDLVTGGELFEDIVAREFYSEADASHCIQQI
 .  .  *   .  **.      .         .*. .   .  *    * .     ..*

DKFLE--KEKHTFRRQVK--LLLLGAGESGKSTFLKQMRIIHGVNFDYELLLEYQS---V
LESVNHCHQNGVVHRDLKPENLLLASKAKGAAVKLADFGLAIEVQGDHQAWFGFAGTPGY
   .   ..    .* .*   *** .   * .  *    .   *. * .    .
```

- No – sequence data is from a G-protein alpha subunit and a CaM kinase

Remember, Distances Between Sequences?

Distance Between Sequences

- Mentioned both Hamming and Levenshtein distance
- Consider two sequences of symbols form the alphabet A = {A,B,C,D}
- How similar are the sequences:
  - S_1= ABCD
  - S_2= ABD
- Find the lowest Distance *D*
- Intuitively S_2 is obtained from S_1 by deleting C
- But……?

Distance Between Sequences

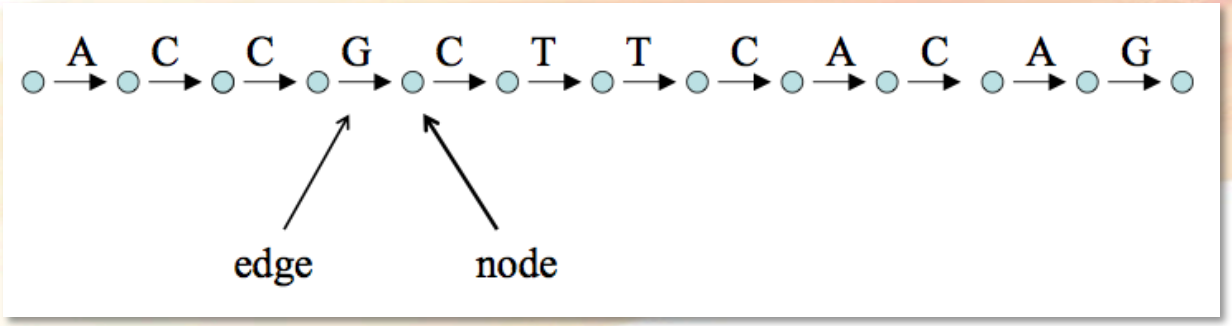
  - S_1 = ABCD
  - S_2 = ABD
- Alternative explanations:
  - S_2 was obtained from S_1 by substituting D for C and then deleting D
  - S_2 was obtained from S_1 by deleting ABCD and inserting ABD
  - ….
- Why is the first explanation intuitively "correct" and the "Alternative explanations" intuitively wrong?
  - We favor the simplest explanation
  - Which involves the minimum number of insertions, deletions, and substitutions

FYI - Occam's Razor

- Commonly attributed to William of Ockham (1290—1349)
  - ``It is vain to do with more what can be done with fewer''
- It states: Entities should not be multiplied beyond necessity
- Commonly explained as: when have choices, choose the simplest theory
- How can this be applied to sequence analysis

Graph Theory and Terminology

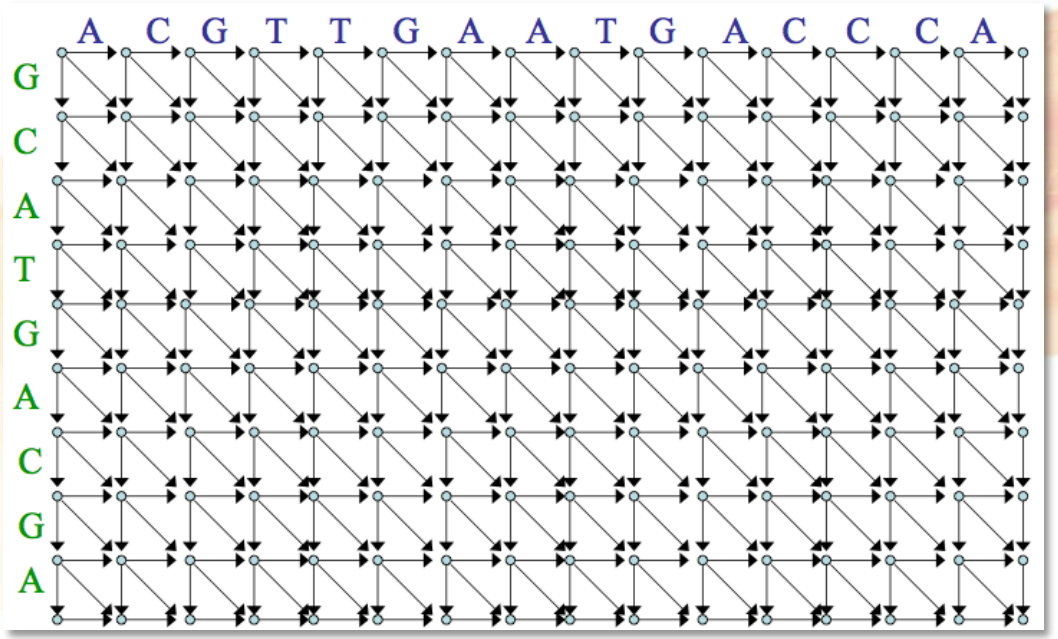
- A sequence graph represents residues as:
  - "**Edges**" between "**nodes**"
  - Also called **vertices**
- Such a graph can represent a single sequence
- Or can be 2-dimensional (or more) and represent a comparison between two sequences



Phil Green

Graphs and Sequence Alignments

- Paths through the graph correspond to alignments of the sequences
  - With each edge on the path corresponding to a column of the alignment
  - Diagonal edges correspond to two aligned residues
  - Horizontal and vertical edges correspond to a residue in one sequence and a gap in the other

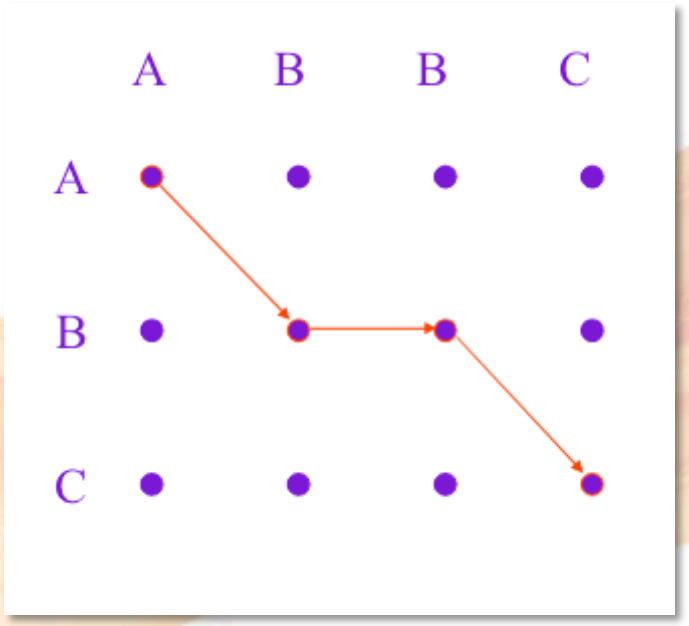


2-D Graph Representing Alignments

Sequence Alignments

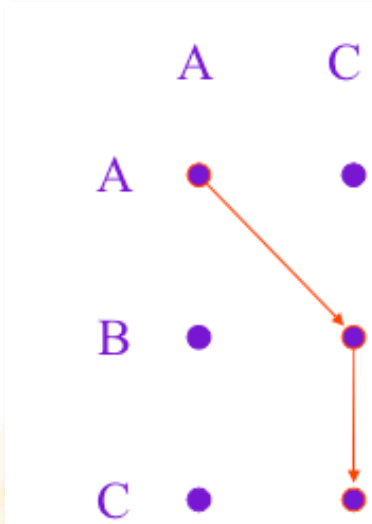
- The relationship between two sequences can be expressed as an alignment between their elements

Insertion/Deletion

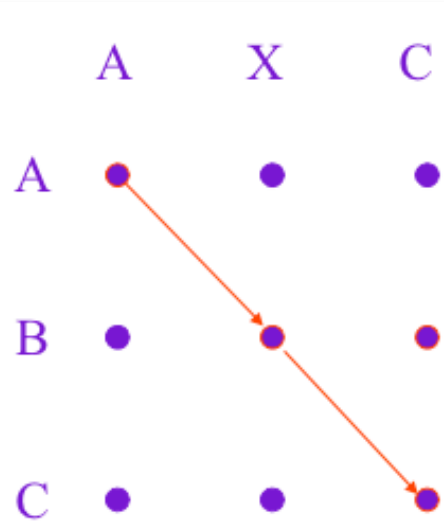


Sequence Alignments

Insertion/Deletion

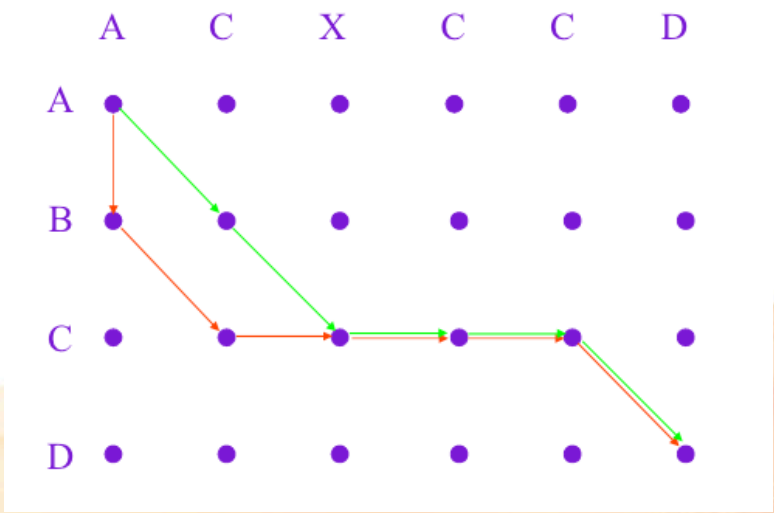


Substitution



Know how to differentiate the two!

General Alignment Path



Which Alignment is best, i.e. which has the lowest *D*?

Edge Weights on Graphs

- Edge weights correspond to scores for an aligned residue or gap (a simple version would be 1 for a match, 0 otherwise)
  - The **weight of a path is the sum of weights for each edge on the path**
  - The **highest weight path corresponds to the highest scoring alignment for that scoring system**
    - **The Optimal Alignment!**
- **Where do the weights come from?**
- Weights are assigned using a substitution score matrix
  - Such as the BLOSUM62 matrix
  - The Needleman-Wunsch paper just uses 1 for each match

Why Score, The Way We Score?

- Over course of evolution, some positions undergo:
  - Base or amino acids substitutions
  - Or bases or amino acids can be indels
- Any measurement of distance/similarity must therefore be done with respect to the best possible alignment between two sequences
- Because indel events are rare compared to base substitutions
  - Makes sense to penalize gaps more heavily than mismatches when calculating the score

Sequence Comparison Overview

- Problem: Find the “best” alignment between a query sequence and a target sequence
- To solve this problem, we need
  - A method for scoring alignments, and
  - An algorithm for finding the alignment with the best score
- The alignment score is calculated using
  - A substitution matrix and gap penalties
  - Let's take a closer look at what you should know how to do:

- **The algorithm for finding the "optimal alignment" is dynamic programming**

Sequence Alignment Algorithms

- Aim:
  - Find the alignment associated with the smallest D (or largest S) from among all possible alignments
  - Without gaps, there are are NxM possible alignments between sequences of length N and M
- The Problem:
  - Once we start allowing gaps, there are many possible arrangements to consider:
  - The number of possible alignments may be astronomical, For example:
    - When two sequences 300 residues long each are compared
    - There are more possible alignments, in comparison, to the number of elementary particles in the universe $\sim 10^{80}$

Too many to enumerate!

$$\frac{(2n)!}{(n!)^2}$$

Luckily, there are computer algorithms for finding the optimal alignment between two sequences that do not require an exhaustive search of all the possibilities; besides not all possibilities are necessary to compute

Dynamic Programming

- Dynamic programming (DP) algorithms are a general class of algorithms typically applied to **optimization problems**
- For DP to be applicable, an optimization problem must have two key ingredients:
  - **Optimal substructure** – an optimal solution to the problem contains within it optimal solutions to sub-problems
  - **Overlapping sub-problems** – the pieces of larger problem have a sequential dependency

Intro to Dynamic Programming in Sequence Alignments

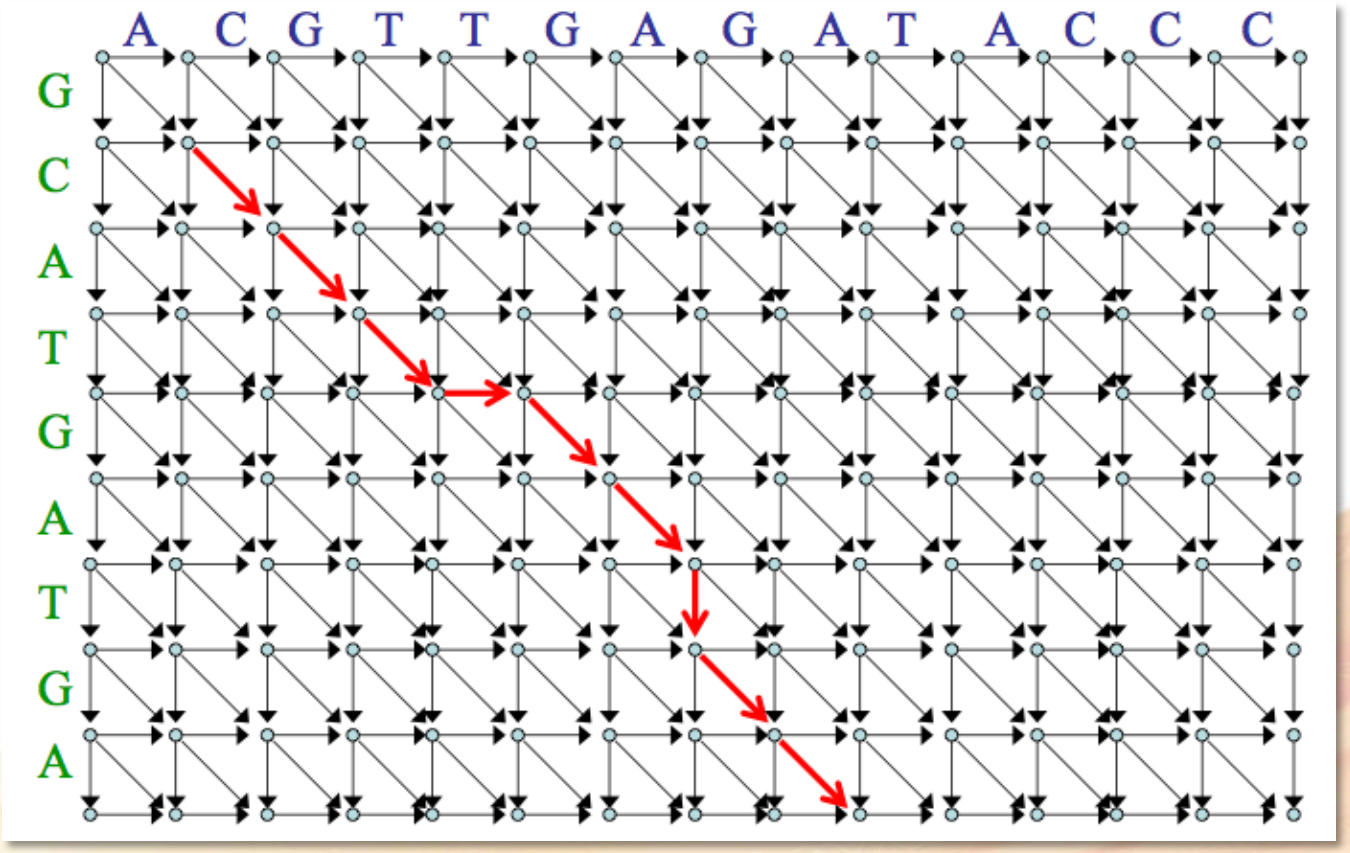
- Two types of alignment
  - Global (Needleman-Wunsch)
    - Attempt to align every residue in the sequences
    - Most useful when sequences are similar in size and sequence
  - Local (Smith-Waterman)
    - Finds an alignment for parts of the two strings
    - Most useful for dissimilar sequences that share regions of similarity or contain similar motifs

Why Use Dynamic Programming?

- Global/Local optimal alignment is a difficult problem
- Where does the difficulty come from?
  - One cannot simply slide one sequence along another and sum over the similarity scores found in the appropriate mutation data matrix
  - This will not work, why?
    - Because biological sequences may have gaps or insertions of sequences relative to each other

Needleman-Wunsch / Smith-Waterman Algorithm

- Start at the top left corner on the graph (or the bottom right corner)
- Proceed across and down the graph, adding up match scores for every path and recording in a 2-D array
- At the end, choose the path that gives the best score at the bottom row or right edge
- For alignment, traceback through best (optimal) score path
- It is possible to prove that this is the best alignment

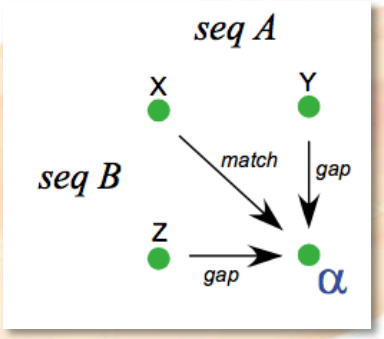


- The above **path corresponds to the following alignment:**

```
...CGTTGA-GA...
...CAT-GATGA...
```

Informal Inductive Proof of Optimal Alignment Path

- Consider the last step in the best alignment path to node α below
- This path must come from one of the three nodes shown, where X, Y, and Z are the cumulative scores of the optimal alignments up to those nodes
- We can reach node α by three possible paths: an A-B match, a gap in sequence A or a gap in sequence B:



The optimal-scoring path to α is the maximum of:
X + match
Y + gap
Z + gap

BUT the optimal paths to X, Y, and Z are analogously the max of their three upstream possibilities, etc. Inductively Q.E.D.

Needleman-Wunsch - Dynamic Programming (DP)

The following is an example of global sequence alignment using Needleman/Wunsch techniques. For this example, the two sequences to be globally aligned are

G A A T T C A G T T A (sequence #1)
G G A T C G A (sequence #2)

So M = 11 and N = 7 (the length of sequence #1 and sequence #2, respectively)

A simple scoring scheme is assumed where

- $S_{i,j} = 1$ if the residue at position i of sequence #1 is the same as the residue at position j of sequence #2 (match score); otherwise
- $S_{i,j} = 0$ (mismatch score)
- w = 0 (gap penalty)

http://www.avatar.se/molbioinfo2001/dynprog/dynamic.html

Three Steps in DP

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

Initialization Step

- The first step is to create a matrix:
  - M + 1 columns
  - N + 1 rows
  - Assuming 0 gap opening or gap extension penalty for simplicity, so:

		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 Step

- Find maximum global alignment score:
  - Start in the upper left hand corner in the matrix and find the maximal score $M_{i,j}$ for each position in the matrix
  - In order to find $M_{i,j}$ for any i,j it is minimal to know the score for the matrix positions to the left, above and diagonal to i, j
  - In terms of matrix positions, it is necessary to know
    - $M_{i-1,\, j-1}$ (diagonal)
    - $M_{i-1,j}$ (left)
    - $M_{i,j-1}$ (above)
- For each position, $M_{i,j}$ is defined to be the maximum score:

$$
\begin{aligned}
M_{i,j} = \mathrm{MAXIMUM}[& \\
& M_{i-1,\, j-1} + S_{i,j} \text{ (match/mismatch in the diagonal)}, \\
& M_{i,j-1} + w \text{ (gap in sequence \#1)}, \\
& M_{i-1,j} + w \text{ (gap in sequence \#2)}]
\end{aligned}
$$

Matrix Fill Step

Note that in the example, $M_{i-1,j-1}$ will be red, $M_{i,j-1}$ will be green and $M_{i-1,j}$ will be blue.

Using this information, the score at position 1,1 in the matrix can be calculated. Since the first residue in both sequences is a G, $S_{1,1} = 1$, and by the assumptions stated at the beginning, $w = 0$. Thus, $M_{1,1} = MAX[M_{0,0} + 1, M_{1,0} + 0, M_{0,1} + 0] = MAX\,[1, 0, 0] = 1$.

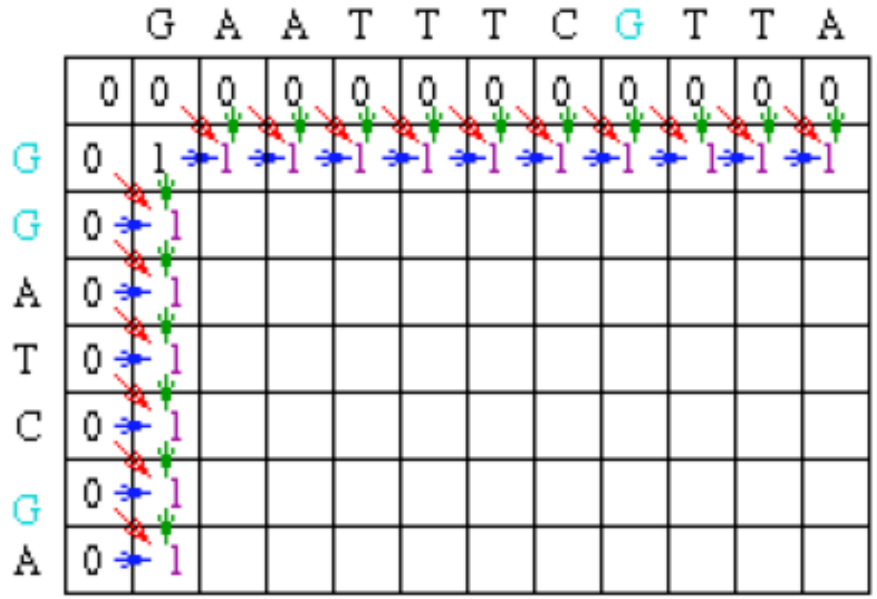
A value of 1 is then placed in position 1,1 of the scoring matrix.

		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	1										
G	0											
A	0											
T	0											
C	0											
G	0											
A	0											

$M_{i,j}$ = **MAXIMUM[**
$M_{i-1,\,j-1} + S_{i,j}$ (match/mismatch in the diagonal),
$M_{i,j-1} + w$ (gap in sequence #1),
$M_{i-1,j} + w$ (gap in sequence #2)]

Matrix Fill Step

Since the gap penalty (w) is 0, the rest of row 1 and column 1 can be filled in with the value 1. Take the example of row 1. At column 2, the value is the max of 0 (for a mismatch), 0 (for a vertical gap) or 1 (horizontal gap). The rest of row 1 can be filled out similarly until we get to column 8. At this point, there is a G in both sequences (light blue). Thus, the value for the cell at row 1 column 8 is the maximum of 1 (for a match), 0 (for a vertical gap) or 1 (horizontal gap). The value will again be 1. The rest of row 1 and column 1 can be filled with 1 using the above reasoning.



$$M_{i,j} = \text{MAXIMUM}[\ M_{i-1,\,j-1} + S_{i,j}\ \text{(match/mismatch in the diagonal)},\ M_{i,j-1} + w\ \text{(gap in sequence \#1)},\ M_{i-1,j} + w\ \text{(gap in sequence \#2)}]$$

Matrix Fill Step

Now let's look at column 2. The location at row 2 will be assigned the value of the maximum of 1(mismatch), 1(horizontal gap) or 1 (vertical gap). So its value is 1.

At the position column 2 row 3, there is an A in both sequences. Thus, its value will be the maximum of 2(match), 1 (horizontal gap), 1 (vertical gap) so its value is 2.

Moving along to position colum 2 row 4, its value will be the maximum of 1 (mismatch), 1 (horizontal gap), 2 (vertical gap) so its value is 2. Note that for all of the remaining positions except the last one in column 2, the choices for the value will be the exact same as in row 4 since there are no matches. The final row will contain the value 2 since it is the maximum of 2 (match), 1 (horizontal gap) and 2(vertical gap).

		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	1	1	1	1	1	1	1	1	1	1	1
G	0	1	1									
A	0	1	2									
T	0	1	2									
C	0	1	2									
G	0	1	2									
A	0	1	2									

$$M_{i,j} = \text{MAXIMUM}[$$
$$M_{i-1,\,j-1} + S_{i,j} \text{ (match/mismatch in the diagonal)},$$
$$M_{i,j-1} + w \text{ (gap in sequence \#1)},$$
$$M_{i-1,j} + w \text{ (gap in sequence \#2)}]$$

Matrix Fill Step

Using the same techniques as described for column 2, we can fill in column 3.

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

$$M_{i,j} = \text{MAXIMUM}[$$
$$M_{i-1,\,j-1} + S_{i,j} \text{ (match/mismatch in the diagonal)},$$
$$M_{i,j-1} + w \text{ (gap in sequence \#1)},$$
$$M_{i-1,j} + w \text{ (gap in sequence \#2)}]$$

Matrix Fill Step

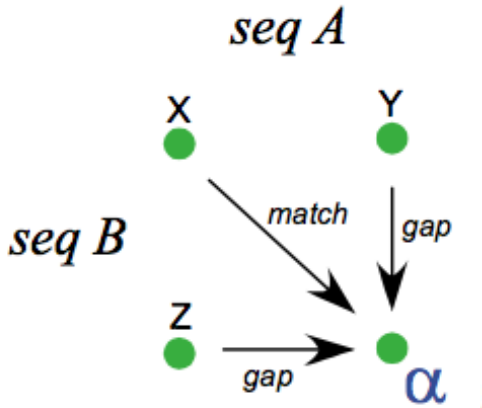
After filling in all of the values the score matrix is as follows:

		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	1	1	1	1	1	1	1	1	1	1	1
G	0	1	1	1	1	1	1	1	2	2	2	2
A	0	1	2	2	2	2	2	2	2	2	2	3
T	0	1	2	2	3	3	3	3	3	3	3	3
C	0	1	2	2	3	3	4	4	4	4	4	4
G	0	1	2	2	3	3	4	4	5	5	5	5
A	0	1	2	3	3	3	4	5	5	5	5	**6**

Traceback Step

- After the matrix fill step:
  - The maximum alignment score for the two test sequences is 6
  - The traceback step determines the actual alignment(s) that result in the maximum score
  - Note:
    - With a simple scoring scheme such as one that is used here, there are likely to be multiple maximal alignments
- Traceback begins in the M,J position in the matrix:
  - i.e. the position that leads to the maximal score
  - In our case, there is a 6 in that location

Traceback for Optimal Alignment Path



- For alignment, node α must have two associated values – the score of the best path leading to α and which node that score came from (X, Y, or Z)
  - A similar pair of values can be kept for every node in the graph
- At the end, search along the bottom and right edges for the highest scoring node and start traceback from that node
  - Follow the node pointers backward from there

Traceback Step

- Traceback takes the current cell and looks to the neighbor cells that could be direct predecessors
  - Looks to:
    - neighbor to the left (gap in sequence #2)
    - diagonal neighbor (match/mismatch)
    - neighbor above it (gap in sequence #1).
  - Traceback chooses highest scoring predecessors

		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	1	1	1	1	1	1	1	1	1	1	1
G	0	1	1	1	1	1	1	1	2	2	2	2
A	0	1	2	2	2	2	2	2	2	2	2	3
T	0	1	2	2	3	3	3	3	3	3	3	3
C	0	1	2	2	3	3	4	4	4	4	4	4
G	0	1	2	2	3	3	4	4	5	5	5	5
A	0	1	2	3	3	3	4	5	5	5	5	6

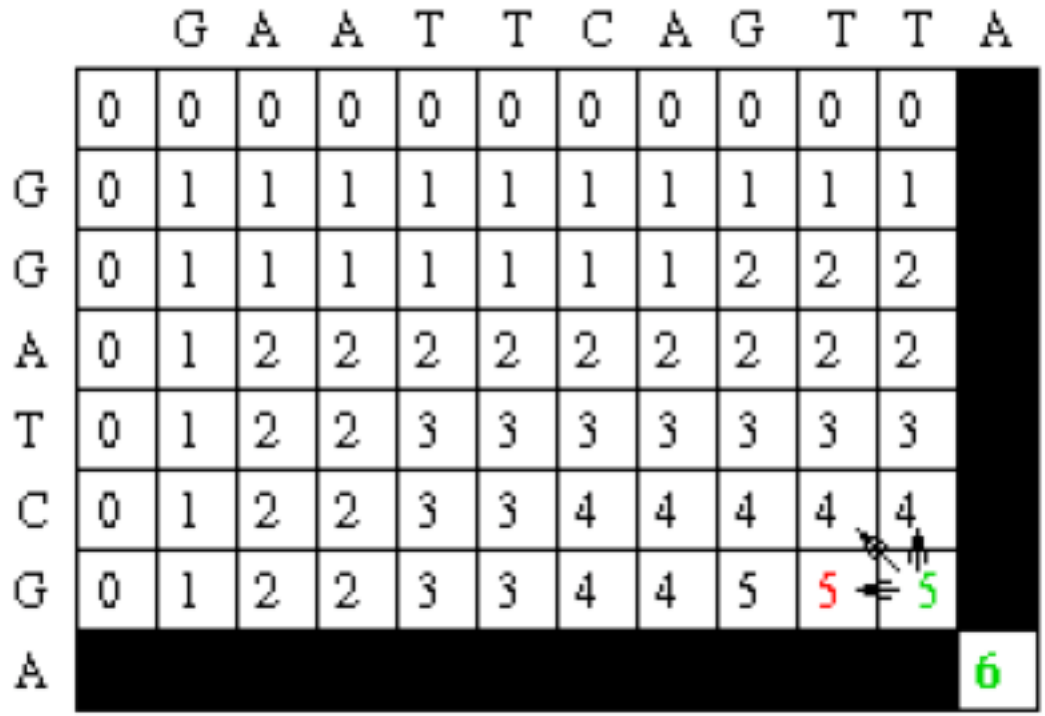
Traceback Step

- Current cell has a value of 6:
  - And since scores:
    - Match = 1
    - 0 for anything else
  - Only possible predecessor is the diagonal match/mismatch neighbor
  - If more than one possible predecessor exists:
    - Any can be chosen
  - This gives us a current alignment of

```
(Seq #1)      A
                |
(Seq #2)      A
```

Traceback Step

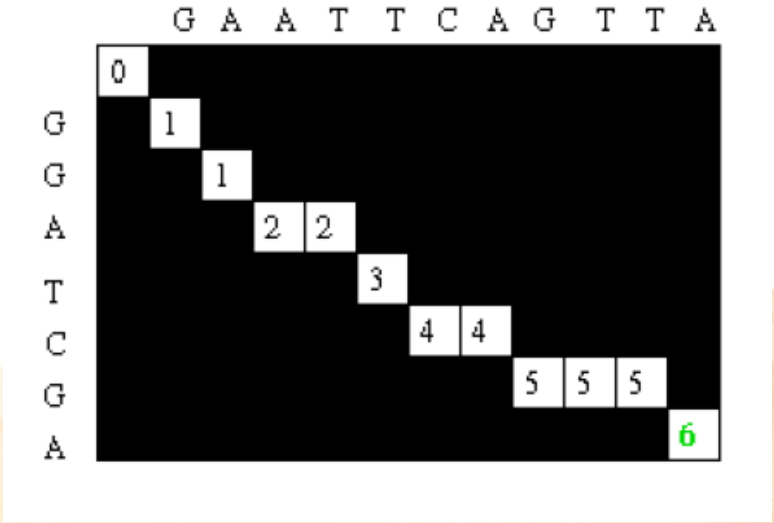
- Now we look at the current cell and determine which cell is its direct predecessor



		G	A	A	T	T	C	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	
G	0	1	1	1	1	1	1	1	1	1	1	
G	0	1	1	1	1	1	1	1	2	2	2	
A	0	1	2	2	2	2	2	2	2	2	2	
T	0	1	2	2	3	3	3	3	3	3	3	
C	0	1	2	2	3	3	4	4	4	4	4	
G	0	1	2	2	3	3	4	4	5	5	5	
A												6

Traceback Step

- Continuing, until we get to column 0 row 0

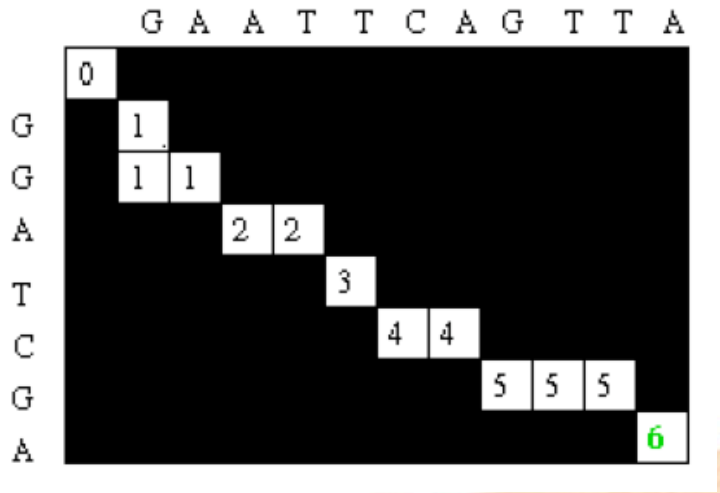


- Giving us an alignment of:

```
G A A T T C A G T T A
|   |   | |   |     |
G G A _ T C _ G _ _ A
```

Traceback Step

- Alternate solution



- Giving us an alignment of:

```
G _ A A T T C A G T T A
|   |   | |   |     |
G G _ A _ T C _ G _ _ A
```

- Our simplistic model didn't take gaps into account
- Traceback is a set of pointers telling where each score cam from
  - See: "Traceback for Optimal Alignment Path"

Dynamic Programming

- Both **global** and **local** types of alignments may be made by simple changes in the basic DP algorithm
  - We'll see this next
- Alignments depend on the choice of a **scoring system** for comparing character pairs and penalty scores
  - e.g. PAM and BLOSUM matrices
  - Our example used a basic scoring scheme,
    - But it's the same for using a scoring matrix
      - Except match and mismatch have varying score associated with them
      - **And then the use of Gap penalty**

Need to Penalize Gaps!

```
TACTTGGATCCGATCAGGAGGAACCGATTC
T--TT---T--G----GG-G-------TT-
```

- Convince yourself that if we allow gaps without affecting score, we can always generate a nearly perfect alignment
  - Constrained only by residue frequency and sequence length
- Thinking biologically, like an amino acid change, an indel indicates evolutionary divergence

Lowering The Gap Penalties

```
scoring matrix: BLOSUM50, gap penalties: -14/-8
14.8% identity;          Global alignment score: -6

               10         20         30         40         50
Seq1   VADCFNITVTEYSIGPAAKKNTSEAVAAANQTEVEMENKVVTKVIREMCVQQYREYRLA-
         . . ..  .. . :...  .... ... .: . :.. .:...::. ::..    :
Seq2   DEYSNQNNFVHDCVNITIKQHTV-TTTTKGENFTETDVKMMERVVEQMCITQYERESQAY
               10         20         30         40         50

        60         70         80
Seq1   --SGIQLHPADTWLAVLLLLLTTLFAMH
         : ..    ..  ..::. .  .. .
Seq2   YQRGSSMVLFSSPPVILLISFLIFLIVG
      60         70         80
```

Low gap penalties -> artificially high percent of identity between sequences

```
scoring matrix: BLOSUM50, gap penalties: -2/-2
28.3% identity;          Global alignment score: 115

                         10         20         30         40
Seq1   ---------VADCFNITVTEYSIGPAAK-KNTSEAVAAANQTEVEM-ENKVVTKVIREMC
               : :: :::. ....  ..: .: .:        :.:.: :     .:...::
Seq2   DEYSNQNNFVHDCVNITIKQHTVTTTTKGENFTE-------TDVKMME-----RVVEQMC
               10         20         30                 40

      50                60          70         80
Seq1   VQQY-RE---Y--RLASGIQLH--PADTWLAVLLLLLTTL-FAM-H
       . :: ::    :  : .:.. :    :       :.::. . : : .
Seq2   ITQYERESQAYYQR-GSSMVLFSSPP-----VILLI-SFLIFLIVG
       50         60          70                 80
```

Results in a different optimal alignment with higher % identity, so be careful!

Dynamic Programming

- Global Dynamic programming matrix

Can we improve?

Smith and Waterman at Los Alamos, New Mexico

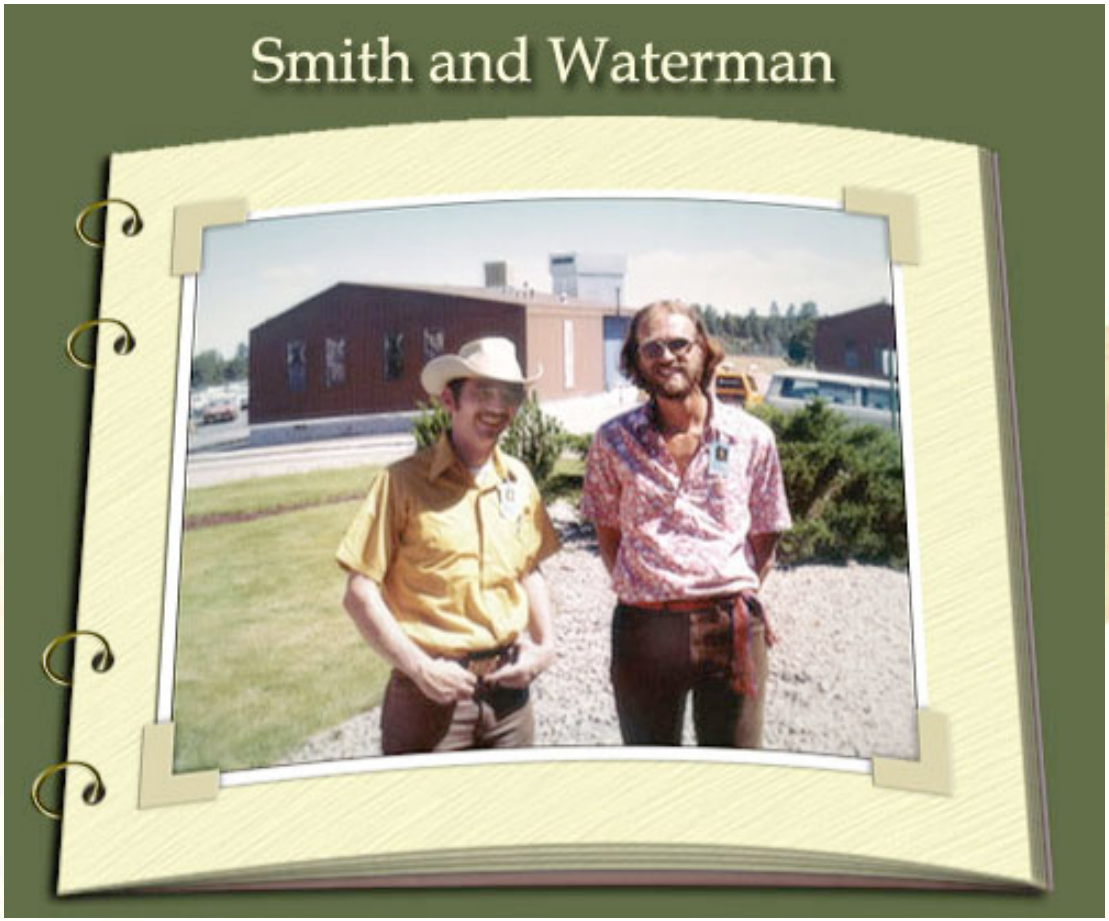


Photo by **David Lipman**, Taken Summer 1980

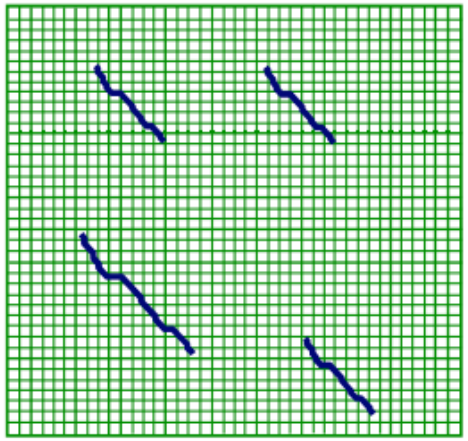
The Smith-Waterman Algorithm

- Idea: Ignore badly aligning regions

Modifications to Needleman-Wunsch:

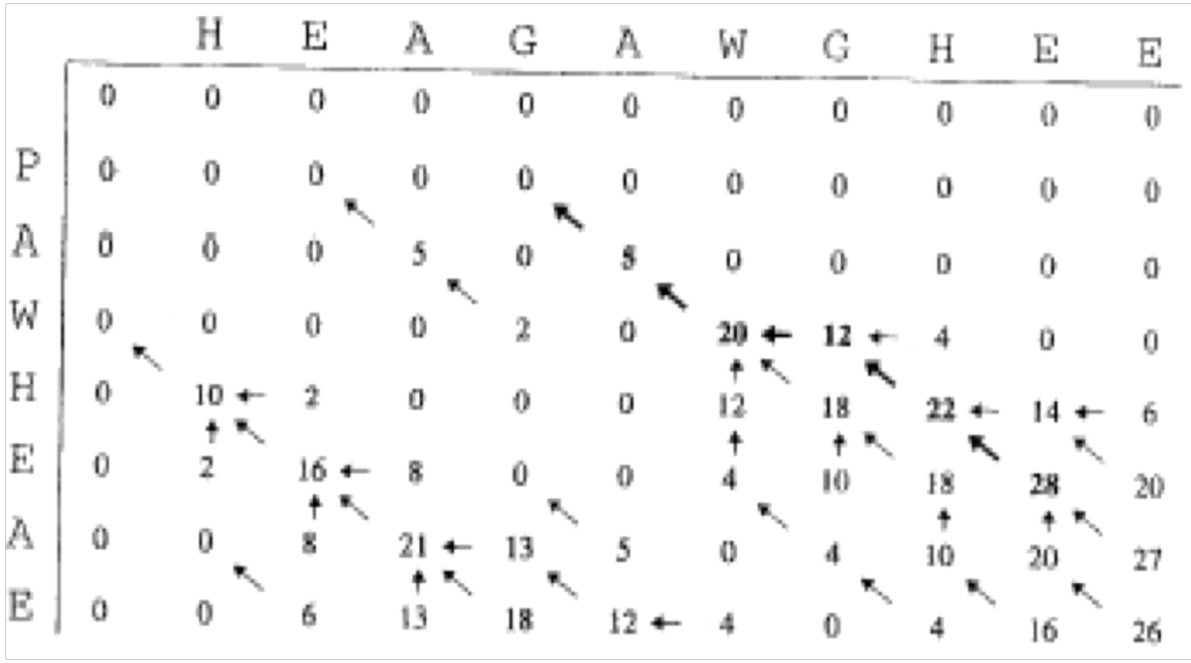
Initialization: $F(0, j) = F(i, 0) = 0$

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



Dynamic Programming

- Local Dynamic programming matrix



		H	E	A	G	A	W	G	H	E	E
	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	5	0	**5**	0	0	0	0	0
W	0	0	0	0	2	0	**20** ←	**12** ←	4	0	0
H	0	10 ←	2	0	0	0	12	18	**22** ←	14 ←	6
E	0	2	16 ←	8	0	0	4	10	18	**28**	20
A	0	0	8	21 ←	13	5	0	4	10	20	27
E	0	0	6	13	18	12 ←	4	0	4	16	26

Smith-Waterman Algorithm

- 2-dimensional matrix with one sequence along the top and the other sequence down the left side
  - "Edge rows" along the top and left side
- All possible alignments are represented by the paths through the matrix
  - A diagonal step is an alignment between the query and the subject sequences at that position
  - a vertical step is a gap in the query sequence
  - a horizontal step is a gap in the subject sequence
- Use the BLOSUM62 matrix, with a linear gap penalty of -6
- Initialize the edge rows to scores of 0

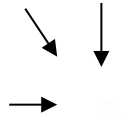
		I	S	A	L	I	G	N	E	D	
	0	0	0	0	0	0	0	0	0	0	
T	0										
H	0										
I	0										
S	0										
L	0										
I	0										
N	0										
E	0										

BLOSUM62 with Positive Scores Marked

	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

Calculating Cell Scores

- Using BLOSUM62 for substitutions, and a -6 linear gap penalty
- The cell at row i and column j has a score F(i, j)
- Starting at top left cell, proceed row-by-row, calculating each cell's score F(i, j). F(i, j) is the maximum of:
  - 0 (i.e. set to 0 if the calculated score is less than 0)
  - F(i-1, j-1) + match/mismatch score for cell (i, j)
  - F(i, j-1) + gap penalty
  - F(i-1, j) + gap penalty



	T	A
T	9	8
P	2	?

For the cell in question, the amino acids don't match. On the BLOSUM62 matrix, this is a -1. There are 3 possible alignment paths to this cell:

1. diagonal (query/subject alignment). Score = 9 – 1 = 8.
2. vertical (query gap). Score = 8 – 6 = 2
3. horizontal (subject gap). Score = 2 – 6 = -4 (set to 0)

Since 8 is the maximum, the cell's value is set to 8.

Smith-Waterman Details

- Start at the first row: T doesn't match anything, and looking at BLOSUM62, the only positive score for a mismatch is +1 with S.
  - We keep track of the 0 -> 1 diagonal
- Second row: H matches N = +1, but nothing else..
  - The diagonal staring with the 1 in the previous row is a H-A mismatch = -2, so 1 -2 = -1, which is scored as 0
- Third row: I gives positive scores with M. L, and V. But, nothing builds on the previous row

		I	S	A	L	I	G	N	E	D	
	0	0	0	0	0	0	0	0	0	0	
T	0	0	1	0	0	0	0	0	0	0	
H	0	0	0	0	0	0	0	1	0	0	
I	0	4	0	0	2	4	0	0	0	0	
S	0										
L	0										
I	0										
N	0										
E	0										

More S-W

- Fourth row: S has positive scores with N, A, and T
  - S-S = +4 match, added to 4 from the diagonal = 8
  - S-A = 1. For a horizontal move (subject gap), 8 – 6 = 2
  - S-I is -2 mismatch, added to 2 from the diagonal = 0
  - S-G = 0 mismatch, added to 4 from the diagonal

		I	S	A	L	I	G	N	E	D	
	0	0	0	0	0	0	0	0	0	0	
T	0	0	1	0	0	0	0	0	0	0	
H	0	0	0	0	0	0	0	1	0	0	
I	0	4	0	0	2	4	0	0	0	0	
S	0	0	8	2	0	0	4	1	0	0	
L	0										
I	0										
N	0										
E	0										

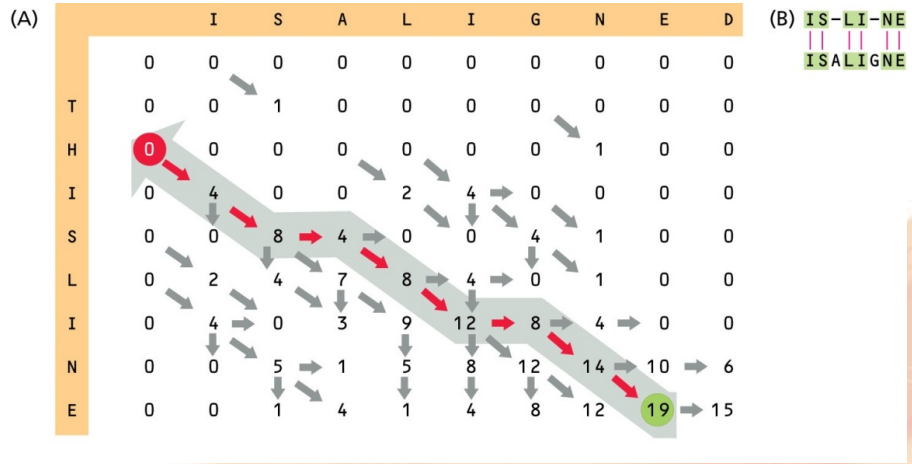
Traceback

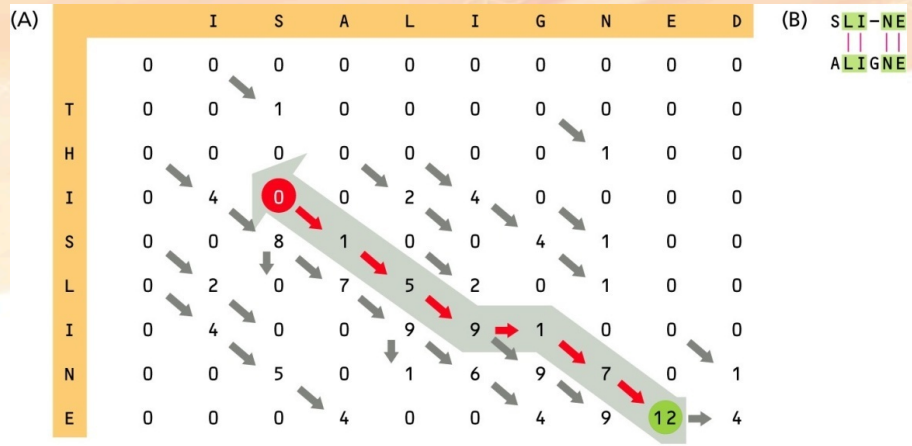
		I	S	A	L	I	G	N	E	D	
	0	0	0	0	0	0	0	0	0	0	
T	0	0	1	0	0	0	0	0	0	0	
H	0	0	0	0	0	0	0	1	0	0	
I	0	4	0	0	2	4	0	0	0	0	
S	0	0	8	3	0	2	4	1	0	0	
L	0										
I	0										
N	0										
E	0										

- As an exercise, you go through and fill in the rest of the values!
- Then do the Traceback
- Begin with highest score in the matrix
- Trace back the path leading through the highest previous scores to 0
- Go left and up only, preferring the diagonal path if a choice needs to be made
- Write the alignment, what do you end up with?

Changing the Gap Penalty

- The top one has a -4 gap penalty and the bottom one has a -8 gap penalty (both linear)
- They give somewhat different alignments





Global vs. Local Illustration

		G	C	T	G	G	A	A	G	G	C	A	T
	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	5	0	0	5	5	0	0	5	5	0	0	0
C	0	0	10	3	0	1	1	0	0	1	10	3	0
A	0	0	3	6	0	0	0	6	.	0	3	15	8
G	0	5	0	0	11	5	0	2	11	5	0	8	11
A	0	0	1	0	4	7	10	5	4	7	1	5	4
G	0	5	0	0	5	9	3	6	10	9	3	0	1
C	0	0	10	3	0	2	5	0	3	6	14	7	0
A	0	0	3	6	0	0	7	10	3	0	7	19	12
C	0	0	5	0	2	0	0	3	6	0	5	12	15
T	0	0	0	10	3	0	0	0	0	2	0	5	17

DYNAMIC PROGRAMMING STEPS

1. **Create matrix and fill with best scores (keeping pointers)**
2. **Find highest score (global in final row and column; local anywhere)**
3. **Trace pointers back to start and generate sequence alignment**

Scores are from a local dynamic programming example in Gibas & Jambeck

```
G C T G G A A G G C A . T
| |   |       |   | | |   |
G C A G . . A . G C A C T
```

GLOBAL

Needleman-Wunsch (GCG Gap)

Starts at final row and column in lower right (17), retraces path

```
  G A A G . G C A
  |   |   |   | | |
  G C A G A G C A
```

LOCAL

Smith-Waterman (GCG BestFit)

Starts at highest score in matrix (19), retraces path

Global vs. Local:

- ***Use global alignment if***
  - You expect, based on some biological information, that your sequences will match over the entire length
  - Your sequences are of similar length
- ***Use local alignment if***
  - You expect that only certain parts of two sequences will match (as in the case of conserved segment that can be found in many different proteins)
  - Your sequences are very different in length
  - You want to search a sequence database

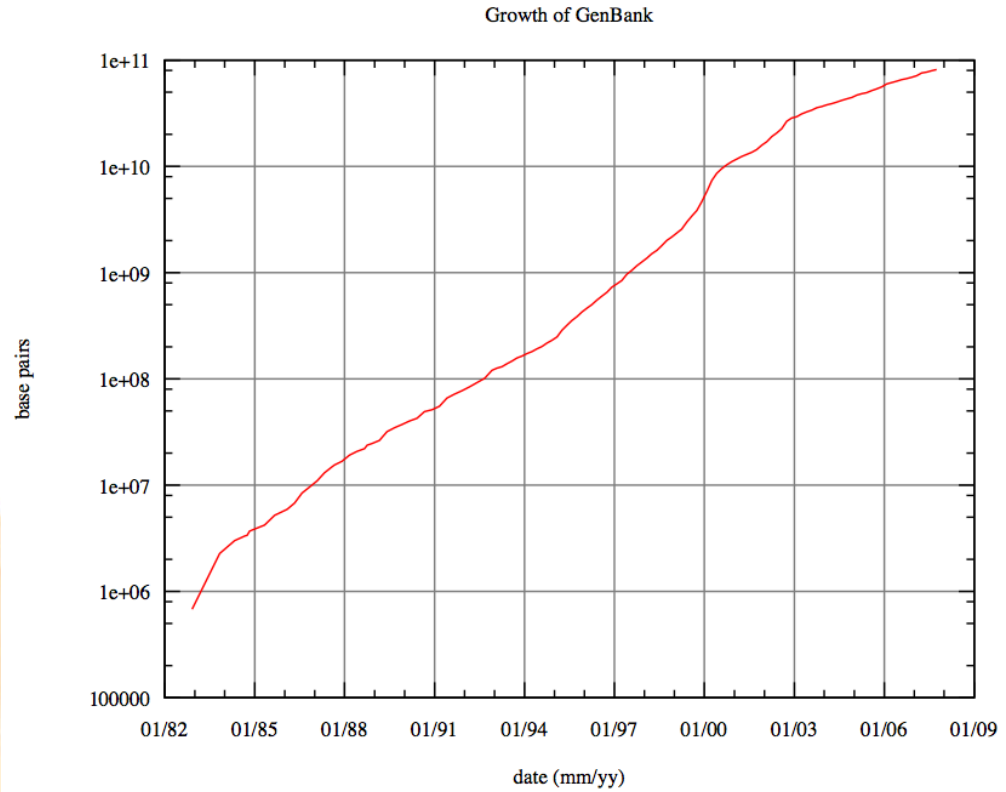
Dynamic Programming

- Advantages:
  Guaranteed in a mathematical sense to provide the optimal (very best or highest-scoring) alignment for a given set of scoring functions

- Disadvantages:
  - **Slow** due to the very large number of computational steps
  - Computer **memory requirement** also increases as the square of the sequence lengths
- Therefore, it is difficult to use the method for very long sequences, or for many pairwise comparisons
  - see upcoming Slides

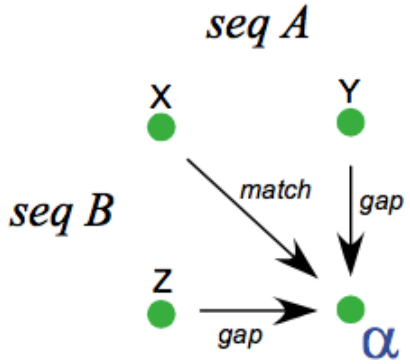
A Common Task

- Look for the best aligning sequences in huge datasets
- Great, we have our tool!
  - Dynamic programming to the rescue?
  - What's the problem?



Search Complexity for Dynamic Programming

- For two sequences of lengths M and N, the sequence alignment graph has M * N nodes and 3 * M * N edges:
  - Search time complexity is of order M * N
  - Search space complexity to reconstruct the highest-scoring alignment is of order M * N
  - Using a 1 Kb DNA sequence to search the GenBank **nr database (~132 billion nucleotides in 2011) would require calculation of about 300 trillion edge weights (3 x 10^{14})**
    - **Even running on fast clusters, this is far too slow for general use**
    - Remember, sequence data sets are growing at least as fast as Moore's law



In step Heuristic methods (word methods)

Let's take 5 minutes…

Heuristic Sequence Comparison

- Heuristic Algorithms:
  - Solve a problem by using rules of thumb to reach a solution
  - The solution is not guaranteed to be an optimal solution
  - Generally arrived at far faster than using an optimal solution approach such as dynamic programming
- Heuristic Sequence Comparison Algorithms:
  - In sequence comparison, commonly used heuristic approaches include the BLAT algorithm, developed by Jim Kent in 2002, the BLAST algorithm, developed by Stephen Altschul in 1990, and the FASTA algorithm, developed by William Pearson in 1988
  - The most widely know and used of these is the BLAST algorithm

FASTA and BLAST

- Sequence alignment methods based on Words
  - Work at the level of words
    - Multiple polypeptides or nucleic acids
    - Instead of individual polypeptides and nucleic acids
- Both methods are fast enough to support searching for alignments of query sequences sequences against entire DBs
- The target dataset is pre-indexed to indicate the positions in all dataset sequences that match each search word above some scoring threshold (using a global score matrix such as BLOSUM62)
  - Here's where there is a major speedup!
- Now we'll begin to understand what gives them their superior speed when compared to DP

Speeding Things Up

- A lot of time is wasted calculating the whole matrix
  - Why?
  - Much of the matrix is far away from the main diagonal
- So, only calculate diagonals where the score is above certain minimum
  - Note that by **not** calculating the entire matrix, you risk missing an optimal alignment
  - Heuristic!
- **Need to start with a well-aligned region on the main diagonal** - we will see how to do this soon
- The FASTA alignment method uses the "diagonal" method

Short Exact Matches

- Smith-Waterman is exact but slow
- The best improvement on S-W involves a list of short sequences that match exactly between the query and subject
- Start by making a list of the position of all possible “k-mers” or “k-tuples” in both sequences
  - K = length
  - Set at:
    - 2 for protein and 6 for DNA in the FASTA program
    - 3 for protein, 11 for DNA in BLAST
- Find all matching k-mers between query and subject, and combine them to form regions of exact ungapped matches
- Computer science algorithms for indexing positions of k-mers:
  - Suffix trees and hashing/chaining

K-mers

…AGCTATGCCATCGACTGCTCCAGTCGCACACACAAAGATTTGAG
GCTATAGCTACTTTATAAAGGGGCTACGGCAAATT…

K-mers

…AGCTATGCCATCGACTGCTCCAGTCGCACACACAAAGATTTGAG
GCTATAGCTACTTTATAAAGGGGCTACGGCAAATT…

k-mers (k≤7)
AGCTATG

K-mers

…AGCTATGCCATCGACTGCTCCAGTCGCACACACAAAGATTTGAG
GCTATAGCTACTTTATAAAGGGGCTACGGCAAATT…

<u>k-mers (k≤7)</u>

AGCTATG

GCTATGC

K-mers

…AGCTATGCCATCGACTGCTCCAGTCGCACACACAAAGATTTGAG
GCTATAGCTACTTTATAAAGGGGCTACGGCAAATT…

<u>k-mers (k≤7)</u>

AGCTATG
GCTATGC
CTATGCC

K-mers

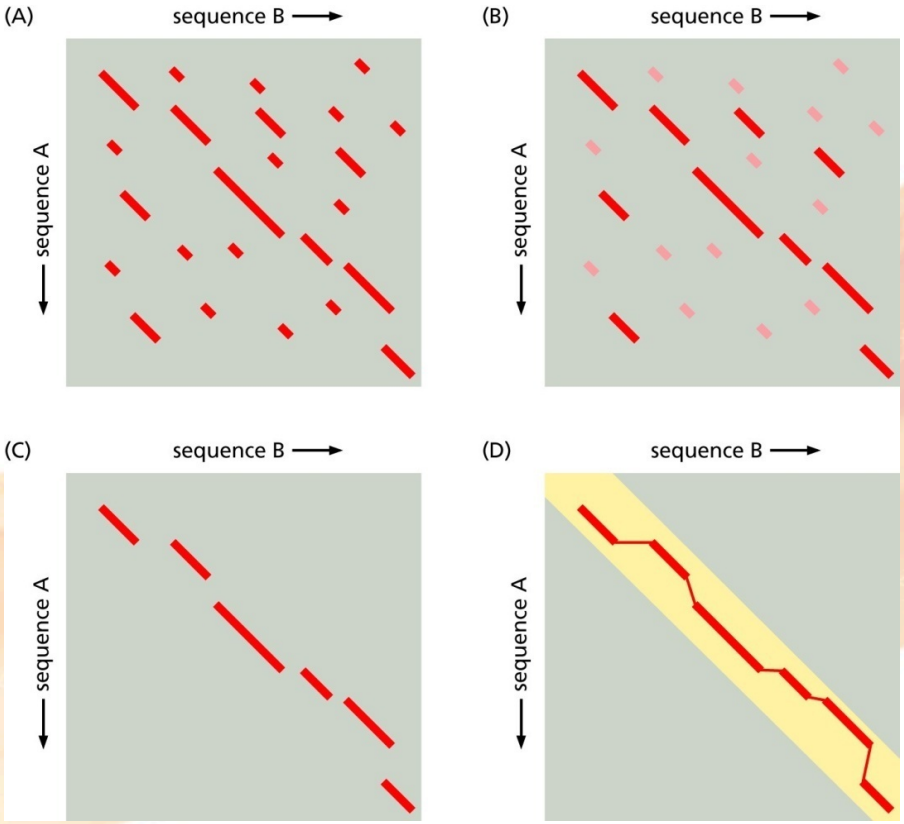
…AGCTATGCCATCGACTGCTCCAGTCGCACACACAAAGATTTGAG
GCTATAGCTACTTTATAAAGGGGCTACGGCAAATT…

<u>k-mers (k≤7)</u>

AGCTATG
GCTATGC
CTATGCC
⋮

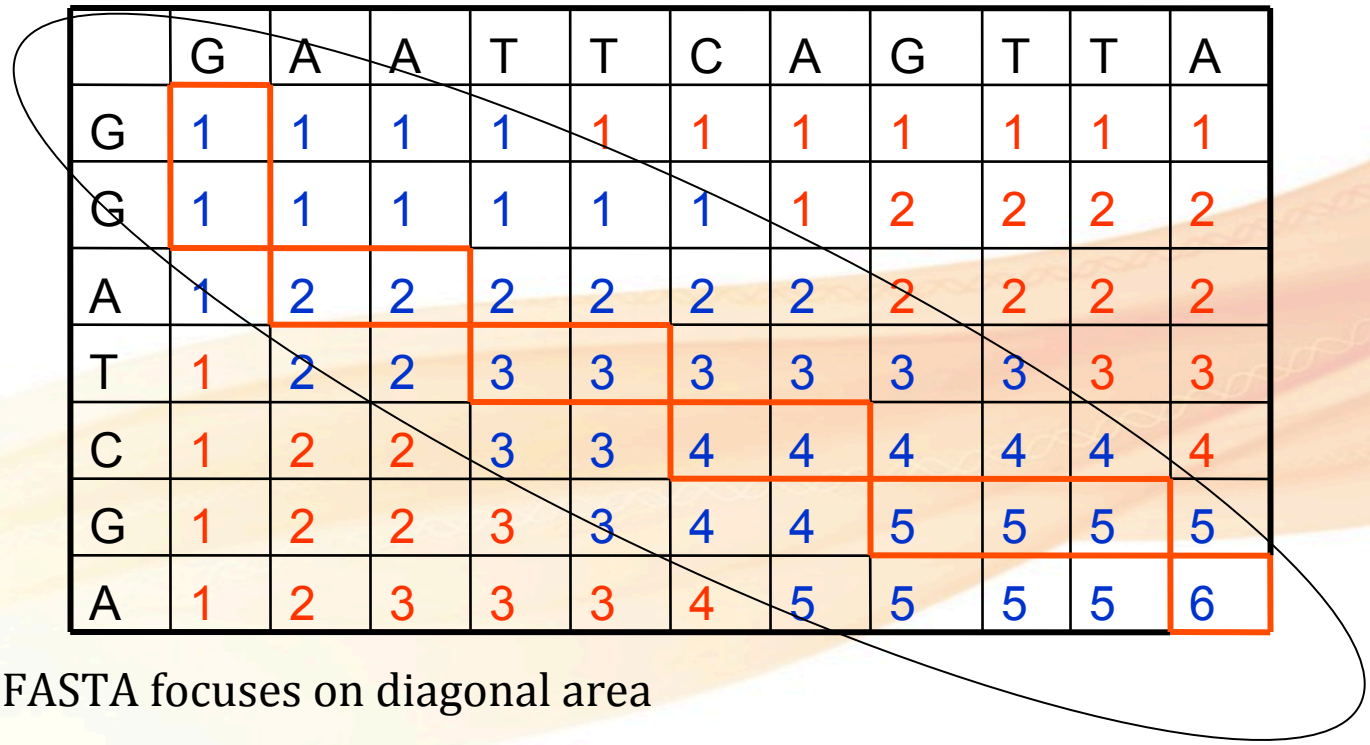
FASTA – the Method

- Best ungapped perfect matching alignments are found
- Connect them with gaps, using a linear gap penalty
  - This is fast
- Then take a band around the best connected chain and use the Smith-Waterman dynamic programming method to produce an optimum alignment
  - Using a substitution matrix
  - And affine gap penalty



FASTA - the Basic Idea

- Problem of Dynamic Programming, computes the scores in a lot of useless area for optimal sequence

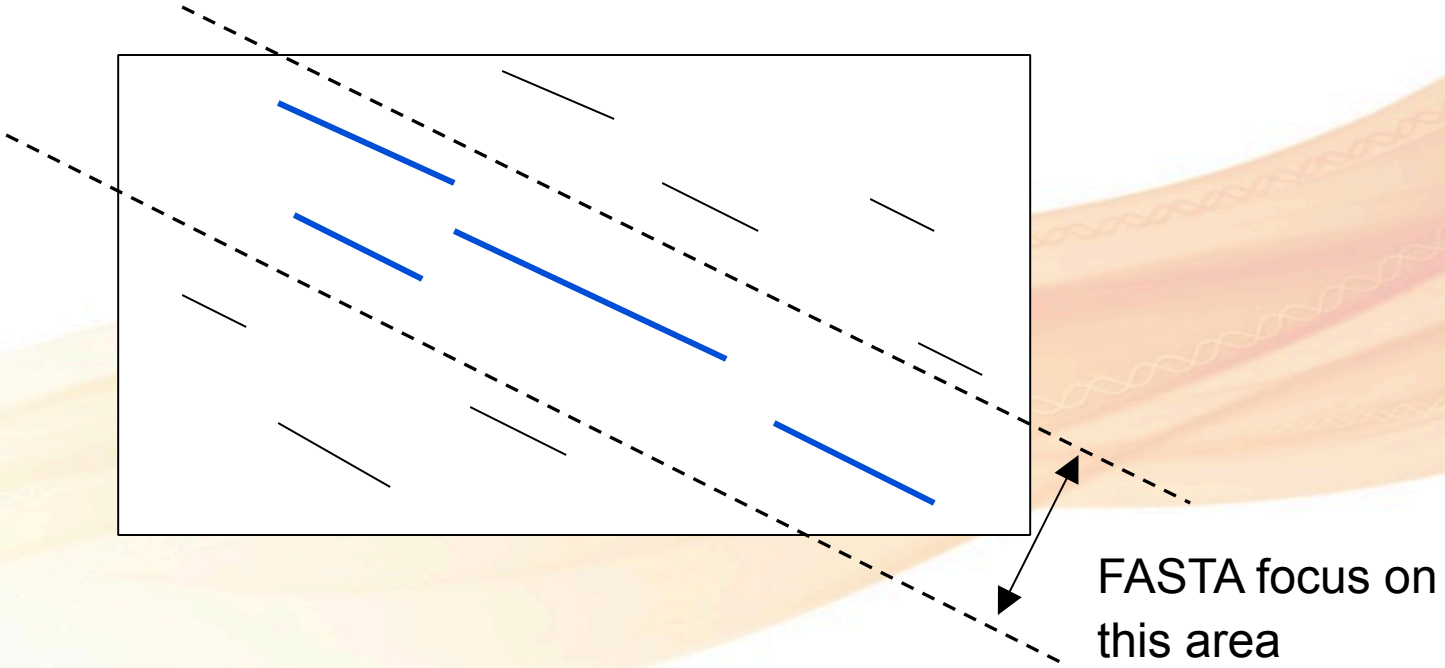


	G	A	A	T	T	C	A	G	T	T	A
G	1	1	1	1	1	1	1	1	1	1	1
G	1	1	1	1	1	1	1	2	2	2	2
A	1	2	2	2	2	2	2	2	2	2	2
T	1	2	2	3	3	3	3	3	3	3	3
C	1	2	2	3	3	4	4	4	4	4	4
G	1	2	2	3	3	4	4	5	5	5	5
A	1	2	3	3	3	4	5	5	5	5	6

- FASTA focuses on diagonal area

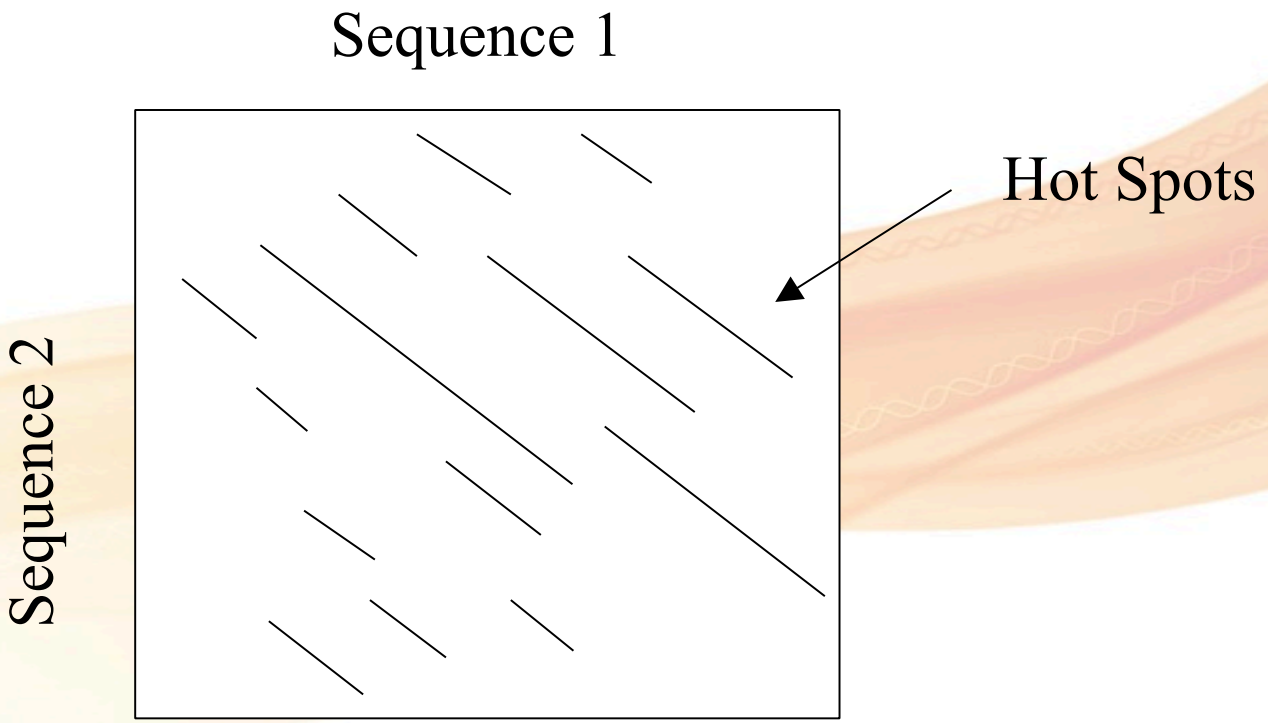
FASTA - the Heuristic

- Heuristic
- Good local alignment should have some exact match subsequence.



FASTA - the Algorithm

- Step 1 - Find all hot-spots
- Hot spots are pairs of words of length k that exactly match



FASTA - the Algorithm

- Step 1 in more detail
- Use look-up Table

  Query : G A A T T C A G T T A

  Sequence: G G A T C G A

Look-up Table

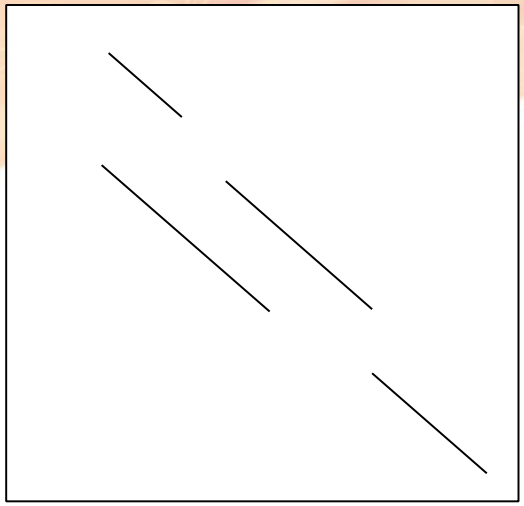
Q	Location
A	2,3,7,11
C	6
G	1,8
T	4,5,9,10

Dot—Matrix

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

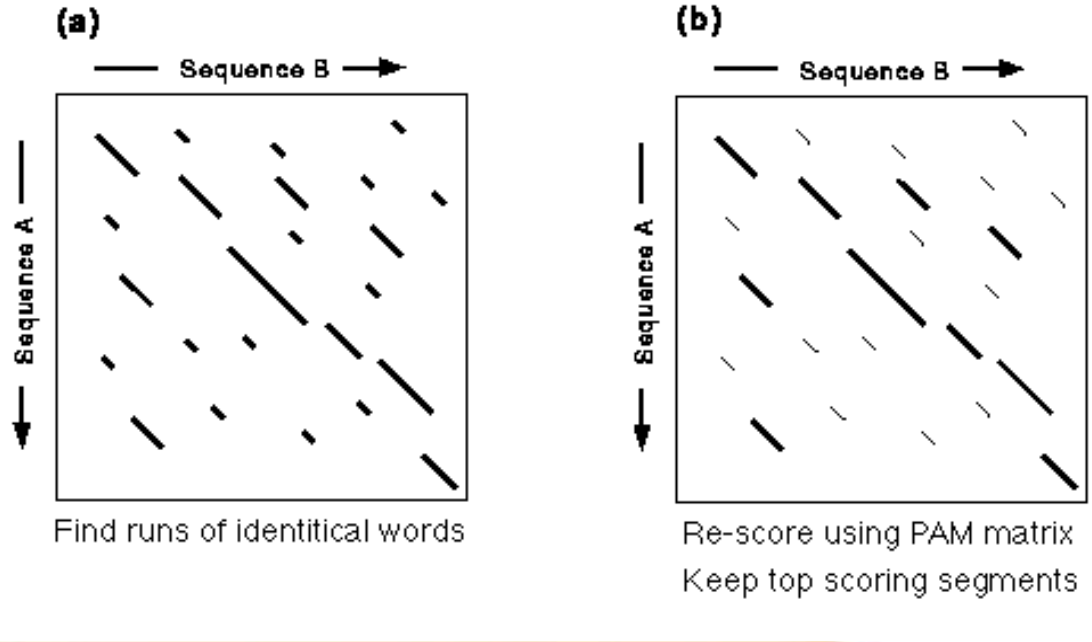
FASTA - the Algorithm

- Step 2
- Score the Hot-spot and locate the **ten** best diagonal runs
  - A diagonal run is a sequence of nearby hot spots on the same diagonal
  - Not necessarily adjacent along the diagonal, i.e. spaces between hot spots are allowed
- Then rescored using a scoring matrix
- ex. PAM250



FASTA - the Algorithm

- Step 2 & 3

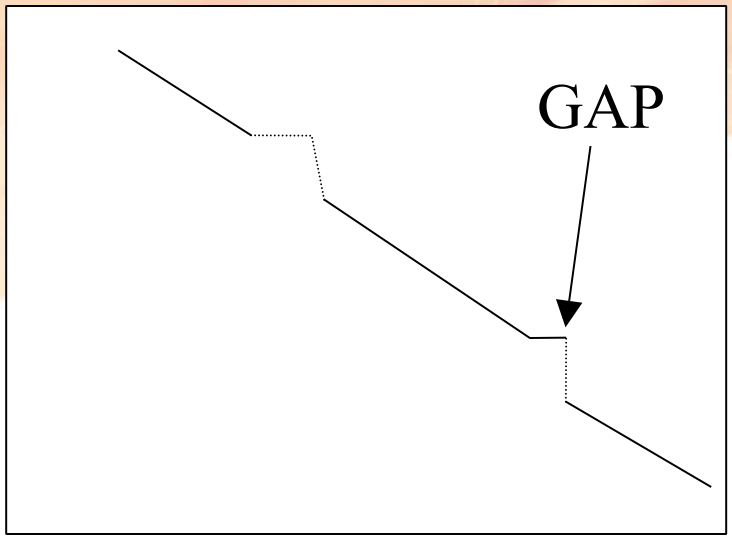


FASTA – the Algorithm

- Step 3
- A diagonal run specifies an alignment, which is composed of matches (the hot spots) and mismatches (from the interspot regions), but does not contain any indels because it is derived from a single diagonal

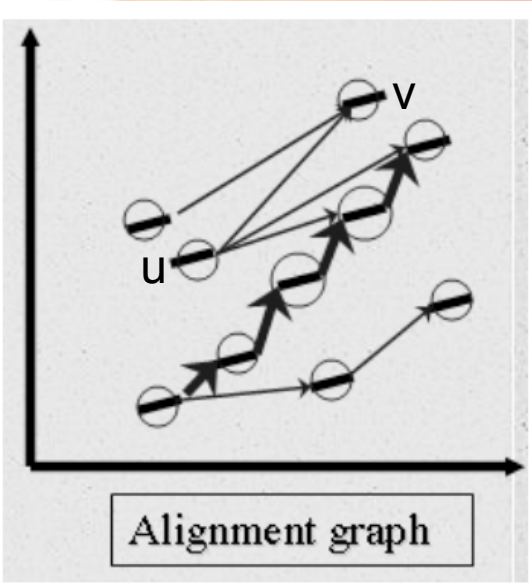
  Next evaluate the runs using an amino acid (or nucleotide) substitution matrix, and pick the best scoring run – $init_1$

  A Filtration is performed and we discard the diagonal runs achieving relatively low scores



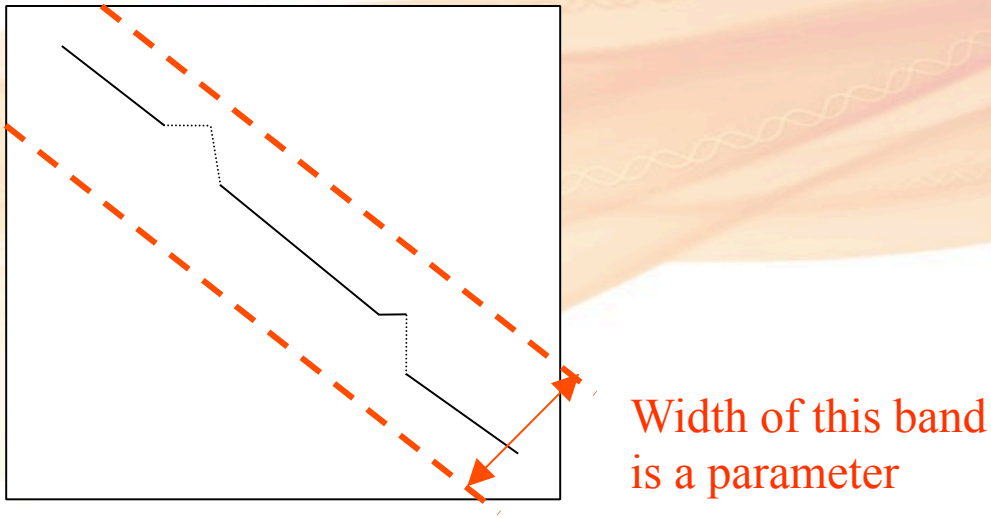
FASTA - the Algorithm

- Step 4 - Combine close diagonal runs and compute $init_n$
- Construct a directed weighted graph
  - Vertices are the sub- alignments found in the previous stage, and the weight of each vertex is the score of the subalignment it represents
- Extend an edge from subalignment ***u*** to subalignment ***v*** if ***v*** starts at a higher row and column than those at which ***u*** ends
  - Give the edge a negative weight
  - Depends on the number of gaps that would be created by aligning according to subalignment ***u*** followed by subalignment ***v***
- Score all
- FASTA then finds a maximum weight path in **this acyclic graph**
- This alignment is known as $init_n$



FASTA - the Algorithm

- Step 5
- Use the dynamic programming in restricted area around the best-score alignment found previously to find out if there are **higher-scoring alignments** than the best-score alignment
- The best local alignment computed in this stage is called *opt.*



FASTA - the Algorithm

- Step 6
- In the last stage, the database sequences are ranked according to $init_n$ scores or *opt* scores, and the full dynamic programming algorithm is used to align the query sequence against each of the highest ranking result sequences

FYI - FASTA – the Algorithm Summary

- 1: Find all hot-spots. Hot spots is pairs of words of length k that exactly match
- 2: Score the Hot-spot and locate the ten best diagonal run
- 3: Combine sub-alignments into one alignment
- 4: Score Each alignment with gap penalty and pick up the best-score alignment
- 5: Use the dynamic programming in restricted area around the best-score alignment to find out the alignment greater than the best-score alignment
- 6: The database sequences are ranked according to $init_n$ scores or *opt* scores, and the full dynamic programming algorithm is used to align the query sequence against each of the highest ranking result sequences

FASTA Results - Histogram

```
!!SEQUENCE_LIST 1.0
(Nucleotide) FASTA of: b2.seq  from: 1 to: 693  December 9, 2002 14:02
TO: /u/browns02/Victor/Search-set/*.seq  Sequences:      2,050  Symbols:
913,285  Word Size: 6
 Searching with both strands of the query.
 Scoring matrix: GenRunData:fastadna.cmp
 Constant pamfactor used
 Gap creation penalty: 16  Gap extension penalty: 4

Histogram Key:
 Each histogram symbol represents 4 search set sequences
 Each inset symbol represents 1 search set sequences
 z-scores computed from opt scores
z-score obs    exp
        (=)    (*)
< 20      0      0:
  22      0      0:
  24      3      0:=
  26      2      0:=
  28      5      0:==
  30     11      3:*==
  32     19     11:==*==
  34     38     30:=======*==
  36     58     61:===============*
  38     79    100:==================    *
  40    134    140:==================================*
  42    167    171:==========================================*
  44    205    189:================================================*====
  46    209    192:================================================*=====
  48    177    184:============================================*
```

FASTA Results - List

```
The best scores are:                      init1 initn   opt    z-sc E(1018780)..

SW:PPI1_HUMAN    Begin: 1  End: 269
! Q00169 homo sapiens (human). phosph... 1854  1854  1854  2249.3  1.8e-117
SW:PPI1_RABIT    Begin: 1  End: 269
! P48738 oryctolagus cuniculus (rabbi... 1840  1840  1840  2232.4  1.6e-116
SW:PPI1_RAT    Begin: 1  End: 270
! P16446 rattus norvegicus (rat). pho... 1543  1543  1837  2228.7  2.5e-116
SW:PPI1_MOUSE    Begin: 1  End: 270
! P53810 mus musculus (mouse). phosph... 1542  1542  1836  2227.5  2.9e-116
SW:PPI2_HUMAN    Begin: 1  End: 270
! P48739 homo sapiens (human). phosph... 1533  1533  1533  1861.0  7.7e-96
SPTREMBL_NEW:BAC25830    Begin: 1  End: 270
! Bac25830 mus musculus (mouse). 10, ... 1488  1488  1522  1847.6  4.2e-95
SP_TREMBL:Q8N5W1    Begin: 1  End: 268
! Q8n5w1 homo sapiens (human). simila... 1477  1477  1522  1847.6  4.3e-95
SW:PPI2_RAT    Begin: 1  End: 269
! P53812 rattus norvegicus (rat). pho... 1482  1482  1516  1840.4  1.1e-94
```

FASTA Results - Alignment

```
SCORES   Init1: 1515  Initn: 1565  Opt: 1687  z-score: 1158.1 E(): 2.3e-58
>>GB_IN3:DMU09374                                          (2038 nt)
 initn: 1565 init1: 1515 opt: 1687 Z-score: 1158.1 expect(): 2.3e-58
  66.2% identity in 875 nt overlap
 (83-957:151-1022)

                  60        70        80        90       100       110
u39412.gb_pr CCCTTTGTGGCCGCCATGGACAATTCCGGGAAGGAAGCGGAGGCGATGGCGCTGTTGGCC
                                        || |||  | ||||| |    ||| |||||
DMU09374     AGGCGGACATAAATCCTCGACATGGGTGACAACGAACAGAAGGCGCTCCAACTGATGGCC
                 130       140       150       160       170       180

                 120       130       140       150       160       170
u39412.gb_pr GAGGCGGAGCGCAAAGTGAAGAACTCGCAGTCCTTCTTCTCTGGCCTCTTTGGAGGCTCA
             |||||||||   ||  |||   |   | ||  |||  |       || || ||||| ||
DMU09374     GAGGCGGAGAAGAAGTTGACCCAGCAGAAGGGCTTTCTGGGATCGCTGTTCGGAGGGTCC
                 190       200       210       220       230       240

                 180       190       200       210       220       230
u39412.gb_pr TCCAAAATAGAGGAAGCATGCGAAATCTACGCCAGAGCAGCAAACATGTTCAAAATGGCC
               |||  | ||||| ||   |||   ||||    | || |  |||||||| || ||| ||
DMU09374     AACAAGGTGGAGGACGCCATCGAGTGCTACCAGCGGGCGGGCAACATGTTTAAGATGTCC
                 250       260       270       280       290       300

                 240       250       260       270       280       290
u39412.gb_pr AAAAACTGGAGTGCTGCTGGAAACGCGTTCTGCCAGGCTGCACAGCTGCACCTGCAGCTC
             ||||||||||     |||||  |    |||||| |||| |||   || |||  || |
DMU09374     AAAAACTGGACAAAGGCTGGGGAGTGCTTCTGCGAGGCGGCAACTCTACACGCGCGGGCT
                 310       320       330       340       350       360
```

FASTA Webservers

EMBL, Heidelberg, Germany
http://www.ebi.ac.uk/Tools/sss/fasta/

PIR – Protein Information Resource
http://pir.georgetown.edu/pirwww/search/fasta.shtml

Kyoto University Bioinformatics Center
http://fasta.genome.jp/

FASTA Sequence Comparison at the U. of Virginia
http://fasta.bioch.virginia.edu/fasta_www2/fasta_list2.shtml

"Sequence database searches must routinely compare hundreds of thousands of sequences, and each comparison requires tens of thousand of steps. FASTA uses computer science techniques and statistical methods to speed things a hundredfold by quickly finding places that are significant." - Dr William Pearson

FASTA on the Web

Many websites offer **FASTA** searches

- Various databases and various other services
- Each server has its limits
- Be aware that you are depending on the kindness of strangers
- **So Best if you can do it through the command line**
- **We'll learn how through online lab**

For Next Tuesday

- Read about how to use gaps in DP
  - http://www.avatar.se/molbioinfo2001/dynprog/adv_dynamic.html
- Go here:
- Read the Introduction and go through these pages
  - http://teacher.bmc.uu.se/UPPSALA06/2can/tutorials/protein/align.html
  - http://teacher.bmc.uu.se/UPPSALA06/2can/tutorials/protein/align1.html
  - http://teacher.bmc.uu.se/UPPSALA06/2can/tutorials/protein/align2.html
- The Statistics of Sequence Similarity Scores
  - http://www.ncbi.nlm.nih.gov/BLAST/tutorial/Altschul-1.html