### Moving through the matrix

Align these Protein sequences
- GAATTCAGTTA, GGATCGA
- Approximately 18 choose 7 possibilities
- Represent all in a matrix
- When Gaps = 0 and Matches = 1, finding the longest common sequence

\[
F_{ij} = \max\{F_{i-1,j} - \sigma |F_{i,j-1} - \sigma |F_{i-1,j-1} + S_{i,j}\}
\]

<table>
<thead>
<tr>
<th></th>
<th>v</th>
<th>G</th>
<th>A</th>
<th>A</th>
<th>T</th>
<th>T</th>
<th>C</th>
<th>A</th>
<th>G</th>
<th>T</th>
<th>T</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td>5</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

GAATTCAGTTA
GGA_TC_G__A
Global Alignment: Scoring

- Align these Protein sequences
- GAATTCAGTTA, GGATCGA
- GGATCGA
- In General Gap penalty, Mismatch penalty

\[ F_{ij} = \max \{ F_{i-1,j} - \sigma | F_{i,j-1} - \sigma | F_{i-1,j-1} + S_{i,j} \} \]
Save your moves

- Align these Protein sequences
- GAATTCAGTTA, GGATCGA
- Have a score for the alignment, but what is the alignment?
- During recursion, data structure saves the previous cell
- Upon termination, can extract aligned sequences

\[
B_{ij} = \begin{cases} \uparrow & \text{if } S_{i-1,j} \\ \leftarrow & \text{if } S_{i,j-1} \\ \& & \text{if } S_{i,j} \end{cases}
\]

GAATTCAGTTA
GGA_TC_G__A
Local Alignment

- Want to identify conserved sub-regions
- Ignore those regions with little similarity
- Biologically irrelevant scores might have higher alignment score
- Find longest local path between arbitrary vertices \((i,j)\) and \((i',j')\)
- Could compute all of these -> Very inefficient
- Instead, connect source \((0,0)\) to every other vertex, with zero weighting
- Or, as a recursion statement?

Why Global Alignment Misses Local:

\[
\text{Diagonal} = n \ast \left( \frac{1}{4} - \frac{3}{4} \ast \mu \right)
\]

\[
\text{Local} = -\frac{2}{3} n \ast \sigma + \frac{1}{3} n - \frac{2}{3} n \ast \sigma
\]

\[
= n \ast \left( \frac{1}{3} - \frac{4}{3} \ast \sigma \right)
\]
Extending Recursion: Local Alignment

- Recursion Statement
- Termination: Anywhere!
- Traceback
  - Maximum alignment score
  - Trace back with stored movements
  - Or all scores higher than some threshold ‘t’
- Initialization $F(i,0)=0$, $F(0,j)=0$

$$F_{ij} = \max\{0|F_{i-1,j} - \sigma|F_{i,j-1} - \sigma|F_{i-1,j-1} + S_{i,j}\}$$
Another Extension: Affine Gap Penalties

- Mutations are frequently insertions and deletions of entire strings
- So why punish a gap as $x \times \text{penalty}$?
- One fix: Penalize $x$ spaces with a slower growing function
- Along those lines, let’s impose a penalty for opening $p$, and a smaller factor for extending gap, $o$
- Adding ‘Long’ vertical and horizontal edges to the graph $(i,j)$ to $(i+x,j)$ with weight $(p+o)$
- Did we increase the run time by adding edges? From $O(n^2)$ to $O(n^3)$?
- Modify the recursion to keep run time down

**Initialization:** same

**Iteration:**

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

**Termination:** same
Affine Gap Penalties: Additional info

- Now when we score a single entry, need to know
  - Are we continuing in a gap in v?
  - Are we continuing a gap in w?
  - Are we continuing a match?

- Again, return to dynamic programming framework to use information about previous optimizations in current optimal score

- Design 3 different matrices, that you can ‘jump’ between
  - A ‘gapping’ matrix for v
  - A ‘gapping’ matrix for w
  - A matching/mismatching matrix for v/w
Affine Gap Penalties: Recursion 1

- Design 3 different matrices, that you can ‘jump’ between:
  - A ‘gapping’ matrix for v
  - A ‘gapping’ matrix for w
  - A matching/mismatching matrix for v/w

- Each Matrix depends on entries from 2 other matrices and itself

\[
a(i, j) = \text{score}(s[i], t[j]) + \max \left\{ \begin{array}{l} a(i-1, j-1) \\ b(i-1, j-1) \\ c(i-1, j-1) \end{array} \right. \\
\]

\[
b(i, j) = \max \left\{ \begin{array}{l} a(i, j-1) - (p+q) \\ b(i, j-1) - q \end{array} \right. \quad \text{starting a gap in s}
\]

\[
c(i, j) = \max \left\{ \begin{array}{l} a(i-1, j) - (p+q) \\ c(i-1, j) - q \end{array} \right. \quad \text{extending a gap in s}
\]
Affine Gap Penalties: Recursion 2

- As One Recursion Statement

\[
F_{ij} = \max \left( F_{i-1,j-1} + S_{i,j}, F_{i,j}^{\rightarrow}, F_{i,j}^{\downarrow} \right)
\]

\[
F_{ij}^{\downarrow} = \max \left( F_{i-1,j}^{\downarrow} - \sigma \left| F_{ij} - (p + \sigma) \right| \right)
\]

\[
F_{ij}^{\rightarrow} = \max \left( F_{i,j-1}^{\rightarrow} - \sigma \left| F_{ij} - (p + \sigma) \right| \right)
\]
Where do scores come from?

- Not all amino acid substitutions are equal
  - Some don’t disrupt protein structure
  - Some impair function
  - Mutable: Asn, Asp, Glu, Ser
  - Not: Cys, Trp

- Amino Acid alignments not matching, but biologically meaningful

- $S(i,j) = \text{how often aa substituted for another}$
  - Likelihood one amino acid is substituted for another

- Trusted alignments of related sequences provide information about biologically permissible mutations

- But, mutation probability time dependent
  - Substitutions likelihood for proteins diverging 15 million years ago different than for those 80 million years ago

- Pam Matrix

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>R</th>
<th>N</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5</td>
<td>-2</td>
<td>-1</td>
<td>-2</td>
</tr>
<tr>
<td>R</td>
<td>-2</td>
<td>7</td>
<td>-1</td>
<td>-2</td>
</tr>
<tr>
<td>N</td>
<td>-1</td>
<td>-1</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>8</td>
</tr>
</tbody>
</table>

\[
f(i,j) = \frac{N(i,j)}{T}
\]
\[
g(i,j) = \frac{f(i,j)}{f(i)}
\]
\[
S(i,j) = \log \left( \frac{f(i,j)}{f(j) \times f(i)} \right)
\]
Hashing

- Let’s say, you want to find all the L-mers in a sequence or database and store their positions
- Sliding window, convert strings to numbers/bit masks
- If want to store the positions for future look up
- Need to create table with indexes for $4^l$ possibilities
  - Some of the L-mers will never be seen
- Problem: $l$ can be 40+ long. Not practical to create indexes for that many L-mers – Use hashing
- Create series of bins with indexes in small range
- In lecture, scan sequence for match, convert strings to numbers, use dependencies on sequential etc..
- Numbers after conversion can be quite long, slow down comparison
- Hashing functions solve this by compressing large numbers into smaller numbers, using smaller set of numbers to match (fall into same bin)
- Problem: Collisions
  - Reduced space, will bin some dissimilar patterns
  - Solution 1: Chaining collisions together – Analyze later
  - But chance of such a collision is low – Randomized algorithms
- Hash tables allow us to quickly look up location of L-mers in database, hash functions useful for pattern matching

$$h(x) = \text{remainder}, x/b$$

Algorithm in Lecture: Finding Matches
Matches bin together
Hashing with Combs

- For BLAST, we may want to extend our L-mers to have non-consecutive symbols.
- Want to store these symbols in a hash table as well, use them to match query.
- **RGIKW** -> **R*IK**, **RG**\*W
  - Indexing is the same as for L-mers
  - Use Bit-Mask to convert letter/wildcard info to a number
  - Store the list of positions in database where they occur
  - If pattern matching, as in lecture, bin matches together
  - Perform lookups to answer query.
- This extend hash table, so can search positions of non-consecutive L-mers, find matches.
- **Why can we do this? Randomized Projection**
  - Two Sequences match if Random subset of their positions match
  - What about false negatives?
  - Example in Lecture slide
- Assume select k positions at random, with replacement
  - Probability of False Negative
  - $1 - (7/9)^4 = 0.63$
  - Repeat several times? Prob missing falls to less than 10%
Quickly searching for matches: BLAST

- Can’t do global alignment with a query and a database of sequences
- But, want to find relevant matches for our query sequence
- How do we Search for pattern in a database?
- This is an approximate pattern finding problem

- Receive query, q
  - Break into sub-sequences, aa-len=3, nuc=11
  - Make a list of ‘neighborhood’ words
    - Cut-off defined by hamming distance
  - Search your database for matches to those words
BLAST: Extend ‘hits’

- Find a set of matches to your L-mer subsequences
- Filters Out Sequences that have no chance anyway
- Now iteratively extend match, without gaps to the query sequence
- Stop at maximum score extension
- Find High scoring Pairs
Problem set up to find L-mers above some threshold

Number of matches with scores above some threshold, T

This follows a poisson distribution

Probability score greater than T from randomly aligned sequences

Probability of seeing the score randomly (p-value0)

Motivation behind BLAST algorithm
  - Set up problem to find matches above some threshold