Lecture 17: Approximate Pattern Matching

Study Chapter 9.6 – 9.8
Approximate vs. Exact Pattern Matching

• Last time we discussed exact pattern matching algorithms
• Usually, because of mutations, it makes much more biological sense to find approximate pattern matches
• Biologists often use fast heuristic approaches (rather than local alignment) to find approximate matches
Heuristic Similarity Searches

- Genomes are huge: Smith-Waterman quadratic alignment algorithms are too slow

- Good alignments of two sequences usually have short identical or highly similar subsequences

- Many heuristic methods (i.e., BLAST, FASTA) are based on the idea of filtration
  - Find short exact matches, and use them as seeds for potential match extension
  - “Filter” out positions with no extendable matches
Dot Matrices

- Dot matrices show similarities between two sequences
- FASTA makes an implicit dot matrix from short exact matches, and tries to find long diagonals (allowing for some mismatches)

\[
l = 1
\]
Dot Matrices

- Dot matrices show similarities between two sequences.
- FASTA makes an implicit dot matrix from short exact matches, and tries to find long diagonals (allowing for some mismatches).

<table>
<thead>
<tr>
<th></th>
<th>G</th>
<th>A</th>
<th>T</th>
<th>T</th>
<th>C</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>T</th>
<th>A</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></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></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></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>*</td>
<td></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>*</td>
<td></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></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></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></td>
<td></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></td>
<td></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></td>
<td></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></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></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></td>
<td></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></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></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></td>
<td></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></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></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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ l = 2 \]
Dot Matrices

- Identify diagonals above a threshold length

- Diagonals in the dot matrix indicate exact substring matching

\[
l = 2
\]
Diagonals in Dot Matrices

- Extend diagonals and try to link them together, allowing for minimal mismatches/indels
- Linking diagonals reveals approximate matches over longer substrings

<table>
<thead>
<tr>
<th></th>
<th>G</th>
<th>A</th>
<th>T</th>
<th>T</th>
<th>C</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>T</th>
<th>A</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></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></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></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></td>
<td></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></td>
<td></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></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></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></td>
<td></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></td>
<td></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></td>
<td></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></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></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></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></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ l = 2 \]
Approximate Pattern Matching Problem

• **Goal:** Find all approximate occurrences of a pattern in a text

• **Input:**
  – pattern $p = p_1 \ldots p_n$
  – text $t = t_1 \ldots t_m$
  – the maximum number of mismatches $k$

• **Output:** All positions $1 \leq i \leq (m - n + 1)$ such that $t_{i} \ldots t_{i+n-1}$ and $p_1 \ldots p_n$ have at most $k$ mismatches
  – i.e., Hamming distance between $t_{i} \ldots t_{i+n-1}$ and $p \leq k$
Approximate Pattern Matching: A Brute-Force Algorithm

ApproximatePatternMatching(p, t, k)

1. \( n \Leftarrow \text{length of pattern } p \)
2. \( m \Leftarrow \text{length of text } t \)
3. \textbf{for } i \Leftarrow 1 \text{ to } m - n + 1 \textbf{ do}
4. \hspace{1em} \textbf{dist} \Leftarrow 0
5. \hspace{2em} \textbf{for } j \Leftarrow 1 \text{ to } n \textbf{ do}
6. \hspace{3em} \textbf{if } t_{i+j-1} \neq p_j \textbf{ then}
7. \hspace{4em} \textbf{dist} \Leftarrow \textbf{dist} + 1
8. \hspace{1em} \textbf{if } \textbf{dist} \leq k \textbf{ then}
9. \hspace{2em} \textbf{output } i
Approximate Pattern Matching: Running Time

• That algorithm runs in $O(nm)$.

• We can generalize the “Approximate Pattern Matching Problem” into a “Query Matching Problem”:
  – We want to match substrings in a query to substrings in a text with at most $k$ mismatches
  – Motivation: we want to see similarities to some gene, but we may not know which parts of the gene to look for
Query Matching Problem

- **Goal:** Find all substrings of the query that approximately match the text

- **Input:** Query $q = q_1 \ldots q_w$, text $t = t_1 \ldots t_m$, $n$ (length of matching substrings $n \leq w \leq m$), $k$ (maximum number of mismatches)

- **Output:** All pairs of positions $(i, j)$ such that the $n$-letter substring of $q$ starting at $i$ approximately matches the $n$-letter substring of $t$ starting at $j$, with at most $k$ mismatches
Approximate Pattern Matching vs Query Matching

(a) Approximate Pattern Matching

(b) Query Matching
Query Matching: Main Idea

- Approximately matching strings share some perfectly matching substrings.

- Instead of searching for approximately matching strings (difficult) search for perfectly matching substrings first (easy).
Filtration in Query Matching

- We want all \( n \)-matches between a query and a text with up to \( k \) mismatches

- “Filter” out positions that do not match between text and query

- **Potential match detection**: find all matches of \( \mathcal{L} \)-tuples in query and text for some small \( \mathcal{L} \)

- **Potential match verification**: Verify each potential match by extending it to the left and right, until \((k + 1)\) mismatches are found
Filtration: Match Detection

• If $x_1...x_n$ and $y_1...y_n$ match with at most $k$ mismatches, they must share an $\ell$-tuple that is perfectly matched, with $\ell = \lceil n/(k + 1) \rceil$

• Break string of length $n$ into $k+1$ parts, each of length $\lfloor n/(k + 1) \rfloor$
  – $k$ mismatches can affect at most $k$ of these $k+1$ parts
  – At least one of these $k+1$ parts is perfectly matched
Suppose $k = 3$. We would then have $l = n/(k+1) = n/4$:

There are at most $k$ mismatches in $n$, so at the very least there must be one out of the $k+1$ $\ell$-tuples without a mismatch.
Filtration: Match Verification

- For each $\mathcal{I}$-match we find, try to extend the match further to see if it is substantial.

Extend perfect match of length $\mathcal{I}$ until we find an approximate match of length $n$ with $k$ mismatches.
### Filtration: Example

<table>
<thead>
<tr>
<th>$\ell$-tuple length</th>
<th>$k = 0$</th>
<th>$k = 1$</th>
<th>$k = 2$</th>
<th>$k = 3$</th>
<th>$k = 4$</th>
<th>$k = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>$n/2$</td>
<td>$n/3$</td>
<td>$n/4$</td>
<td>$n/5$</td>
<td>$n/6$</td>
<td></td>
</tr>
</tbody>
</table>

- **Shorter perfect matches required**
- **Performance decreases**
Local alignment is too slow...

- Quadratic local alignment is too slow while looking for similarities between long strings (e.g. the entire GenBank database)

\[
s_{i,j} = \max\begin{cases} 
0 \\
 0 \\
 0 \\
 s_{i-1,j} + \delta(v_i, -) \\
 s_{i,j-1} + \delta(-, w_j) \\
 s_{i-1,j-1} + \delta(v_i, w_j) 
\end{cases}
\]
Local alignment is too slow…

- Quadratic local alignment is too slow while looking for similarities between long strings (e.g. the entire GenBank database)
- Guaranteed to find the optimal local alignment
- Sets the standard for sensitivity

\[
s_{i,j} = \max \begin{cases} 0 \\ s_{i-1,j} + \delta(v_i,-) \\ s_{i,j-1} + \delta(-,w_j) \\ s_{i-1,j-1} + \delta(v_i,w_j) \end{cases}
\]
Local alignment is too slow...

- Quadratic local alignment is too slow while looking for similarities between long strings (e.g. the entire GenBank database)

- **Basic Local Alignment Search Tool**
  - Altschul, S., Gish, W., Miller, W., Myers, E. & Lipman, D.J.

- Search sequence databases for local alignments to a query
BLAST

• Great improvement in speed, with a modest decrease in sensitivity
• Minimizes search space instead of exploring entire search space between two sequences
• Finds short exact matches (“seeds”), only explores locally around these “hits”
What Similarity Reveals

• BLASTing a new gene
  – Evolutionary relationship
  – Similarity between protein function

• BLASTing a genome
  – Potential genes
Measuring Similarity

- Measuring the extent of similarity between two sequences
  - Based on percent sequence **identity**
  - Based on **conservation**
Percent Sequence Identity

- The extent to which two nucleotide or amino acid sequences are invariant

\[
\text{ACCTGAG} \quad \text{ACGTCAG}
\]

70% identical

Mismatch
Indel
Conservation

• Amino acid changes that preserve the physico-chemical properties of the original residue
  – Polar to polar
    • aspartate $\rightarrow$ glutamate
  – Nonpolar to nonpolar
    • alanine $\rightarrow$ valine
  – Similarly behaving residues
    • leucine to isoleucine
Scoring matrices

• Amino acid substitution matrices
  – PAM
  – BLOSUM

• DNA substitution matrices
  – DNA: less conserved than protein sequences
  – Less effective to compare coding regions at nucleotide level
PAM

- **Point Accepted Mutation** (Dayhoff et al.)
- 1 PAM = PAM$_1$ = 1% average change of all amino acid positions
  - After 100 PAMs of evolution, not every residue will have changed
    - May have mutated several times
    - Returned to their original state
    - Not changed at all
## PAM\textsubscript{250} is most widely used:

|    | Ala | Arg | Asn | Asp | Cys | Gln | Glu | Gly | His | Ile | Leu | Lys | ...
|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-------
| Ala | A   | R   | N   | D   | C   | Q   | E   | G   | H   | I   | L   | K   | ...
| Ala A | 13  | 6   | 9   | 9   | 5   | 8   | 9   | 12  | 6   | 8   | 6   | 7   | ...
| Arg R | 3   | 17  | 4   | 3   | 2   | 5   | 3   | 2   | 6   | 3   | 2   | 9   | ...
| Asn N | 4   | 4   | 6   | 7   | 2   | 5   | 6   | 4   | 6   | 3   | 2   | 5   | ...
| Asp D | 5   | 4   | 8   | 11  | 1   | 7   | 10  | 5   | 6   | 3   | 2   | 5   | ...
| Cys C | 2   | 1   | 1   | 1   | 52  | 1   | 1   | 2   | 2   | 2   | 1   | 1   | ...
| Gln Q | 3   | 5   | 5   | 6   | 1   | 10  | 7   | 3   | 7   | 2   | 3   | 5   | ...
| ...  |     |     |     |     |     |     |     |     |     |     |     |     | ...
| Trp W | 0   | 2   | 0   | 0   | 0   | 0   | 0   | 0   | 1   | 0   | 1   | 0   | ...
| Tyr Y | 1   | 1   | 2   | 1   | 3   | 1   | 1   | 1   | 3   | 2   | 2   | 1   | ...
| Val V | 7   | 4   | 4   | 4   | 4   | 4   | 4   | 4   | 5   | 4   | 15  | 10  | ...
• **Blocks Substitution Matrix** (Henikoff and Henikoff)
• Scores derived from *observations* of the frequencies of substitutions
  – From blocks of local alignments in related proteins
• Matrix name indicates evolutionary distance
  – BLOSUM62 was created using sequences sharing no more than 62% identity
BLAST algorithm

- **Keyword search** of all words of length $w$ from the query of length $n$ in database of length $m$ with score above threshold
  - $w = 11$ for DNA queries, $w = 3$ for proteins
- **Local alignment extension** for each found keyword
  - Extend result until longest match above threshold is achieved
- **Running time** $O(nm)$
**BLAST algorithm**

Query: KRHRKVLRDNIQGITKPAIRRLARRGGV KRISGLIYEEETRGVLK IFLE NVIRD

---

**neighborhood score threshold**

(T = 13)

- GVK 18
- GAK 16
- GIK 16
- GGK 14
- GLK 13
- GNK 12
- GRK 11
- GEK 11
- GDK 11

---

**Neighborhood words**

---

**extension**

---

Query: 22 VLRDNIQGITKPAIRRLARRGGV KRISGLIYEEETRGVLK 60
+++DN +G + IR L G+K I+ L+ E+ RG++K

Sbjct: 226 IIKDNGRGSQIRNLNYGILKVIADLV-EKHRGIK 263

High-scoring Pair (HSP)
Original BLAST

- **Dictionary**
  - All words of length $w$

- **Alignment**
  - *Ungapped* extensions until score falls below some statistical threshold

- **Output**
  - All local alignments with score $>\text{threshold}$
Original BLAST: Example

- $w = 4$
- Exact keyword match of GGTC
- Extend diagonals with mismatches until score is under 50%
- Output result GTAAGGTCC GTTAGGTCC

From lectures by Serafim Batzoglou (Stanford)
Gapped BLAST: Example

- Original BLAST exact keyword search, THEN:
- Extend with gaps around ends of exact match until score < threshold
- Output result
  GTAAGGTCCAGT
  GTTAGGTC–AGT

From lectures by Serafim Batzoglou (Stanford)
Incarnations of BLAST

- blastn: Nucleotide-nucleotide
- blastp: Protein-protein
- blastx: Translated query vs. protein database
- tblastn: Protein query vs. translated database
- tblastx: Translated query vs. translated database (6 frames each)
Incarnations of BLAST (cont’d)

• PSI-BLAST
  – Find members of a protein family or build a custom position-specific score matrix

• Megablast:
  – Search longer sequences with fewer differences

• WU-BLAST: (Wash U BLAST)
  – Optimized, added features
Assessing sequence similarity

• Need to know how strong an alignment can be expected from chance alone
• “Chance” relates to comparison of sequences that are generated randomly based upon a certain sequence model
• Sequence models may take into account:
  – G+C content
  – Poly-A tails
  – “Junk” DNA
  – Codon bias
  – Etc.
BLAST: Segment Score

- BLAST uses scoring matrices ($\delta$) to improve on efficiency of match detection
  - Some proteins may have very different amino acid sequences, but are still similar
- For any two $\ell$-mers $x_1...x_\ell$ and $y_1...y_\ell$:
  - Segment pair: pair of $\ell$-mers, one from each sequence
  - Segment score: $\sum_{i=1}^{\ell} \delta(x_i, y_i)$
BLAST: Locally Maximal Segment Pairs

- A segment pair is maximal if it has the best score over all segment pairs.
- A segment pair is locally maximal if its score can’t be improved by extending or shortening.
- Statistically significant locally maximal segment pairs are of biological interest.
- BLAST finds all locally maximal segment pairs with scores above some threshold.
  - A significantly high threshold will filter out some statistically insignificant matches.
BLAST: Statistics

• Threshold: Altschul-Dembo-Karlin statistics
  – Identifies smallest segment score that is unlikely to happen by chance
• # matches above θ has mean $E(\theta) = Kmne^{-\lambda\theta}$; $K$ is a constant, $m$ and $n$ are the lengths of the two compared sequences
  – Parameter $\lambda$ is a positive root of:
    $$\Sigma_{x,y \in A}(p_x p_y e^{\delta(x,y)}) = 1,$$
    where $p_x$ and $p_y$ are frequencies of amino acids $x$ and $y$, $\delta$ is the scoring matrix, and $A$ is the twenty letter amino acid alphabet
P-values

• The probability of finding $b$ HSPs with a score $\geq \theta$ is given by:
  \[ \frac{(e^{-E}E^b)}{b!} \]

• For $b = 0$, that chance is:
  \[ e^{-E} \]

• Thus the probability of finding at least one HSP with a score $\geq \theta$ is:
  \[ P = 1 - e^{-E} \]
Sample BLAST output

Sequences producing significant alignments:

<table>
<thead>
<tr>
<th>Sequence ID</th>
<th>Description</th>
<th>Score (bits)</th>
<th>Expect</th>
</tr>
</thead>
<tbody>
<tr>
<td>gi</td>
<td>18858329</td>
<td>ref</td>
<td>NP_571095.1</td>
</tr>
<tr>
<td>gi</td>
<td>18858331</td>
<td>ref</td>
<td>NP_571096.1</td>
</tr>
<tr>
<td>gi</td>
<td>37606100</td>
<td>emb</td>
<td>CAE48992.1</td>
</tr>
<tr>
<td>gi</td>
<td>31419195</td>
<td>gb</td>
<td>AAH53176.1</td>
</tr>
</tbody>
</table>

ALIGNMENTS

>gi|18858329|ref|NP_571095.1| ba1 globin [Danio rerio]

Length = 148

Score = 171 bits (434), Expect = 3e-44
Identities = 76/148 (51%), Positives = 106/148 (71%), Gaps = 1/148 (0%)

Query: 1 MVHLTPEEKSAVTALWGBKVNDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPK 60
      MV  T  E++A+  LWGK+N+DE+G +AL R L+VYPWTQR+F +FG+LS+P A+MGNPK

Sbjct: 1 MVEWITDAERTAILGLWGBKLNIDEIGPQALSRCLIVYPWTQRFATFGNLSPPAAIMGNPK 60

Query: 61 VKAHGKKVLGAFSDGLAHLNKLKGFATLSELHCDKLHVDPENFRLNQVLAVLHFFG 120
      V AHG+ V+G ++DN+A LS +H +KLHVDP+NFRLL + + A FG

Sbjct: 61 VAAHGRTVMGGLERAIKNMDNVKNTYAALSVMHSEKLHVDPDNFRLLADICTVCAAMKFG 120

Query: 121 KE-FTPPVQAAYQKVAGVANALAHKYH 147
      + F VQ A+QK +A V +AL +YH

Sbjct: 121 QAGFNADVQEAWKFLAVVVSALCRQYH 148
**Sample BLAST output (cont’d)**

- **Blast of human beta globin DNA against human DNA**

Sequences producing significant alignments:

<table>
<thead>
<tr>
<th>Accession</th>
<th>Description</th>
<th>Score</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>gi</td>
<td>19849266</td>
<td>gb</td>
<td>AF487523.1</td>
</tr>
<tr>
<td>gi</td>
<td>183868</td>
<td>gb</td>
<td>M11427.1</td>
</tr>
<tr>
<td>gi</td>
<td>44887617</td>
<td>gb</td>
<td>AY534688.1</td>
</tr>
<tr>
<td>gi</td>
<td>31726</td>
<td>emb</td>
<td>V00512.1</td>
</tr>
<tr>
<td>gi</td>
<td>38683401</td>
<td>ref</td>
<td>NR_001589.1</td>
</tr>
<tr>
<td>gi</td>
<td>18462073</td>
<td>gb</td>
<td>AF339400.1</td>
</tr>
</tbody>
</table>

**ALIGNMENTS**

>`gi|28380636|ref|NG_000007.3| Homo sapiens beta globin region (HBB@) on chromosome 11`

  Length = 81706
  Score = 149 bits (75), Expect = 3e-33
  Identities = 183/219 (83%)
  Strand = Plus / Plus

Query: 267   ttgggagatgccacaaagcacctggatgatctcaagggcacctttgcccagctgagtgaa 326

  || || | || | || | || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || |
Timeline

• 1970: Needleman-Wunsch global alignment algorithm
• 1981: Smith-Waterman local alignment algorithm
• 1985: FASTA
• 1990: BLAST (basic local alignment search tool)
• 2000s: BLAST has become too slow in “genome vs. genome” comparisons - new faster algorithms evolve!
  – Pattern Hunter
  – BLAT
PatternHunter: faster and even more sensitive

• BLAST: matches short consecutive sequences (consecutive seed)
  • Length = \(k\)
  • Example (\(k = 11\)):
    \[
    11111111111
    \]
    Each 1 represents a “match”

• PatternHunter: matches short non-consecutive sequences (spaced seed)
  • Increases sensitivity by locating homologies that would otherwise be missed
  • Example (a spaced seed of length 18 w/ 11 “matches”):
    \[
    111010010100110111
    \]
    Each 0 represents a “don’t care”, so there can be a match or a mismatch
Spaced seeds

Example of a hit using a spaced seed:

GAGTACTCAACACCAAACATTTAGTGCAATGGAAAAAT...

How does this result in better sensitivity?
Why is PH better?

- BLAST: redundant hits

  - PatternHunter

This results in > 1 hit and creates clusters of redundant hits

This results in very few redundant hits
Why is PH better?

BLAST may also miss a hit

GAGTACTCAACACCAACATTTAGTGGGCAATGGAAAT

|| || |||| | ||| ||| |||||

GAATACTCAACAGCAACATCAATGGGCAAGCAAAT

9 matches

In this example, despite a clear homology, there is no sequence of continuous matches longer than length 9. BLAST uses a length 11 and because of this, BLAST does not recognize this as a hit!

Resolving this would require reducing the seed length to 9, which would have a damaging effect on speed
Advantage of Gapped Seeds

11 positions

11 positions

10 positions

$\text{similarity}$
Why is PH better?

• Higher hit probability
• Lower expected number of random hits
Use of Multiple Seeds

Basic Searching Algorithm

1. Select a group of spaced seed models
2. For each hit of each model, conduct extension to find a homology.
Another method: BLAT

• BLAT (BLAST-Like Alignment Tool)
• Same idea as BLAST - locate short sequence hits and extend
BLAT vs. BLAST: Differences

- BLAT builds an index of the database and scans linearly through the query sequence, whereas BLAST builds an index of the query sequence and then scans linearly through the database.
- Index is stored in RAM which is memory intensive, but results in faster searches.
BLAT: Fast cDNA Alignments

Steps:
1. Break cDNA into 500 base chunks.
2. Use an index to find regions in genome similar to each chunk of cDNA.
3. Do a detailed alignment between genomic regions and cDNA chunk.
4. Use dynamic programming to stitch together detailed alignments of chunks into detailed alignment of whole.

A sophisticated divide and conquer approach
However…

- BLAT was designed to find sequences of 95% and greater similarity of length >40; may miss more divergent or shorter sequence alignments
PatternHunter and BLAT vs. BLAST

• PatternHunter is 5-100 times faster than Blastn, depending on data size, at the same sensitivity

• BLAT is several times faster than BLAST, but best results are limited to closely related sequences