Lecture 18: Approximate Pattern Matching

Study Chapter 9.6 – 9.8

Approximate vs. Exact Pattern Matching

- Previously we have 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 Matrix

- A dot matrix or dot plot 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)
- Nucleotide matches

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

$l = 1$

Dot Matrix

- A dot matrix or dot plot 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)
- Dinucleotide matches

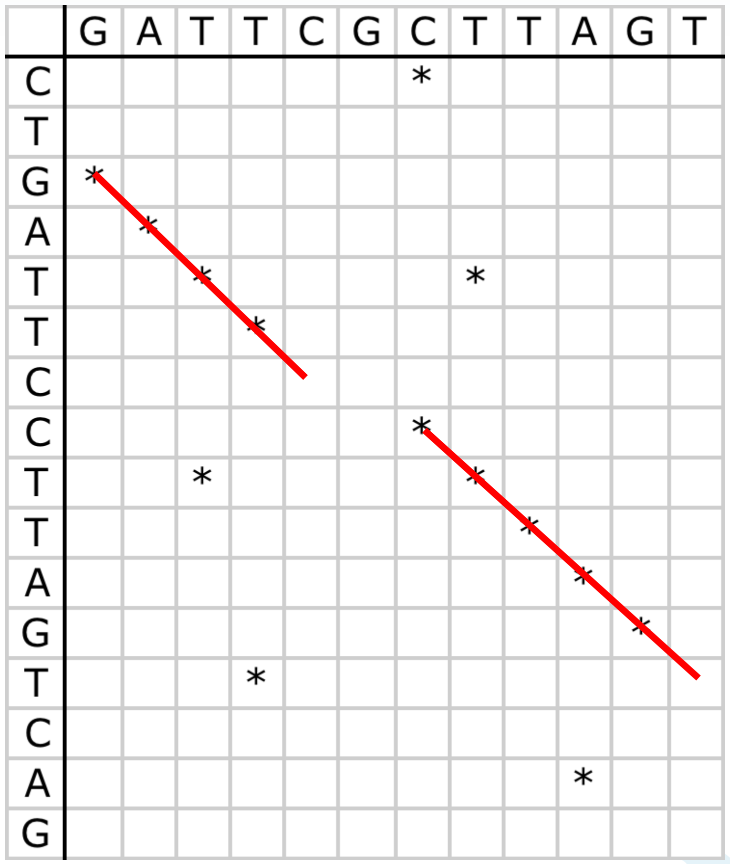
	G	A	T	T	C	G	C	T	T	A	G	T
C							*					
T												
G	*											
A		*										
T			*					*				
T				*								
C												
C							*					
T			*					*				
T									*			
A										*		
G											*	
T				*								
C												
A										*		
G												

$l = 2$

Dot Matrix

- Identify diagonals above a threshold length

- Diagonals in the dot matrix indicate exact substring matching

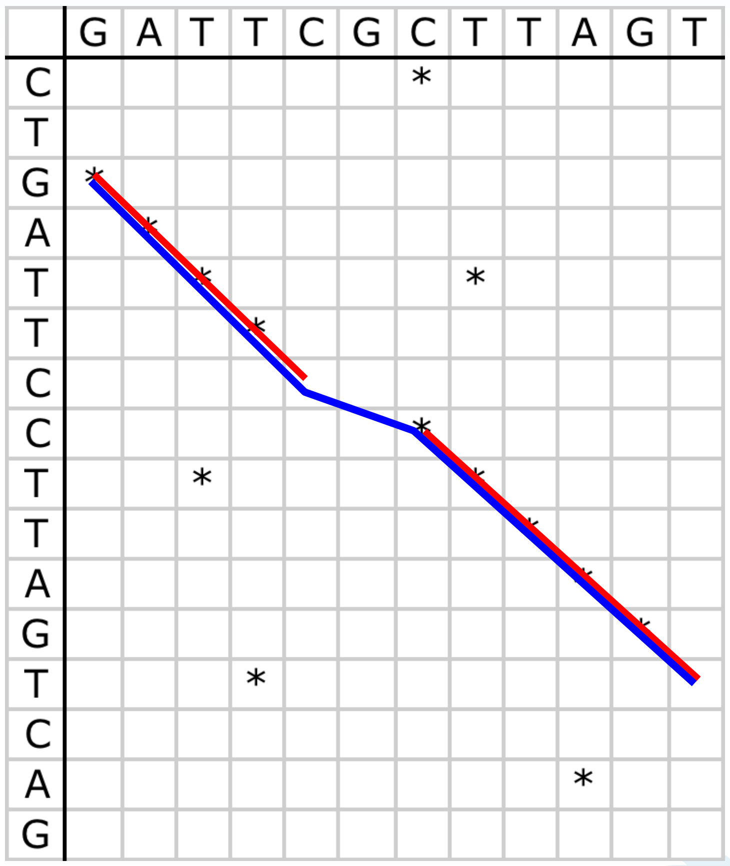


	G	A	T	T	C	G	C	T	T	A	G	T
C							*					
T												
G	*											
A		*										
T			*					*				
T				*								
C												
C							*					
T			*					*				
T									*			
A										*		
G											*	
T				*								
C												
A										*		
G												

$l = 2$

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



$l = 2$

Approximate Pattern Matching (APM)

- 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** $\leq k$

APM: A Brute-Force Algorithm

```
ApproximatePatternMatching(p, t, k)
1  $n \leftarrow$ length of pattern p
2  $m \leftarrow$ length of text t
3  for $i \leftarrow 1$ to $m - n + 1$
4     $dist \leftarrow 0$
5     for $j \leftarrow 1$ to $n$
6        if $t_{i+j-1} \,!= p_j$
7           $dist \leftarrow dist + 1$
8     if $dist \leq k$
9        output $i$
```

APM: Running Time

- That algorithm runs in O(nm).

- Extend "Approximate Pattern Matching" to a more general "Query Matching Problem":
  - Match *n-length substring of the query* (not the full pattern) to a substring in a text with at most $\boldsymbol{k}$ mismatches
  - **Motivation**: we may seek similarities to some gene, but not know which parts of the gene to consider

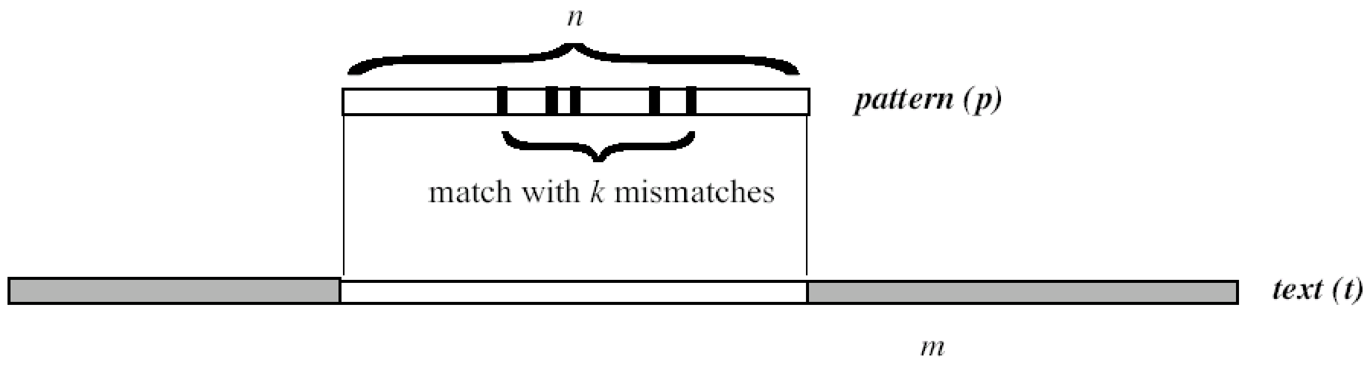
Query Matching Problem

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

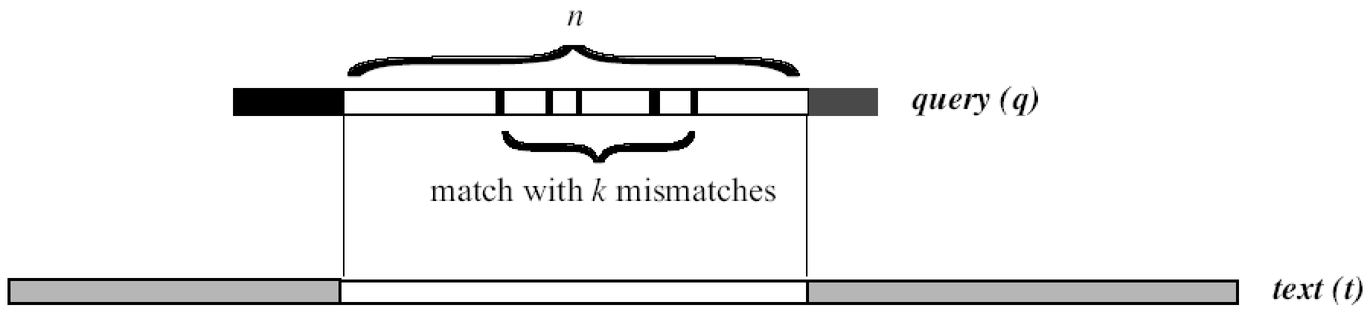
- Input: Query **q** = $q_1 \ldots q_w$,
  text **t** = $t_1 \ldots t_m$,
  n (length of matching substrings $n \le w \le m$),
  k (maximum number of mismatches)

- 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

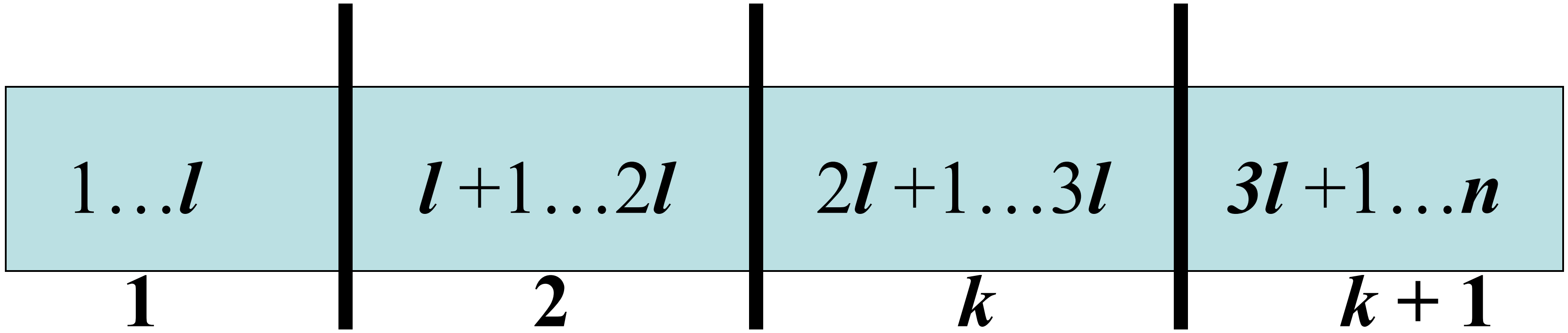
- **Potential match detection**: find all matches of ℓ-tuples in query and text for some small ℓ
- **Potential match verification**: Verify each potential match by extending it to the left and right, until (k + 1) mismatches are found

Filtration: Match Detection

- If $x_1 \ldots x_n$ and $y_1 \ldots y_n$ match with at most $k << n$ mismatches they must share ℓ–mers that are perfect matches, with $\ell = \lfloor n/(k+1) \rfloor$
- 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

Filtration: Match Detection (cont'd)

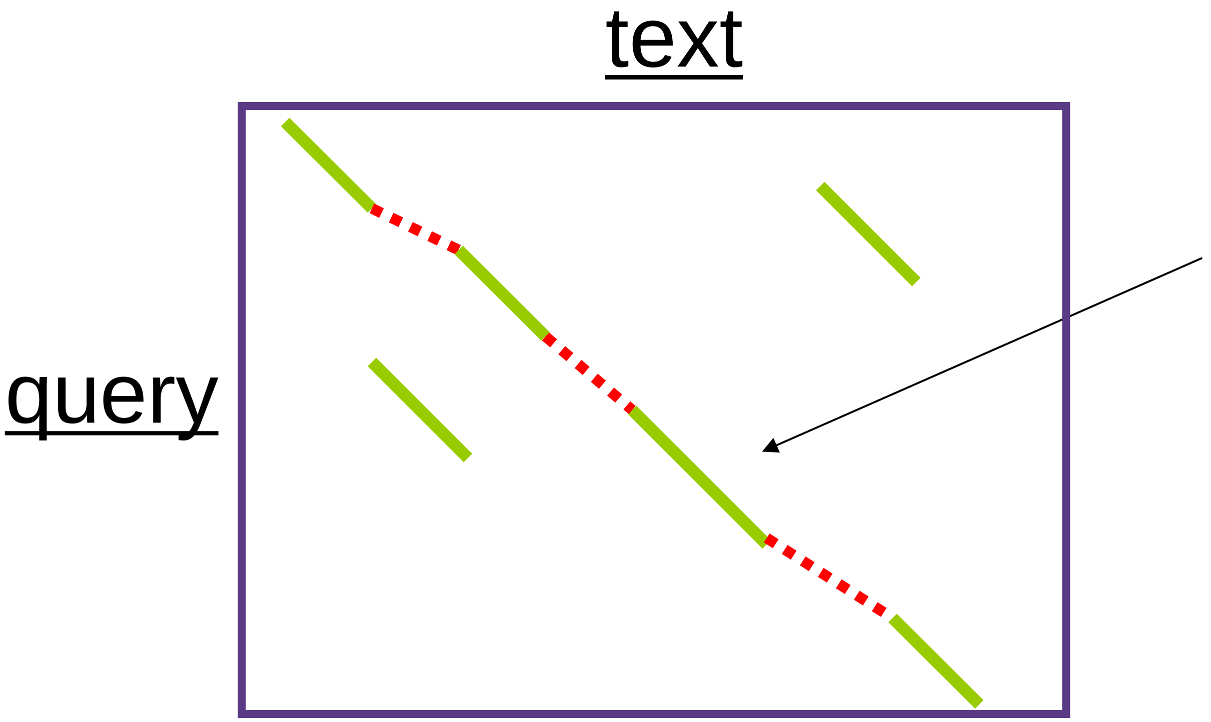
- Suppose $k = 3$. We would then have $l=n/(k+1)=n/4$:



- There are at most k mismatches in n, so at the very least there must be one out of the k+1 l–tuples without a mismatch

Filtration: Match Verification

- For each ℓ-match we find, try to extend the match further to see if it is substantial



Extend perfect match of length ℓ until we find an approximate match of length n with no more than k mismatches

Filtration: Example

	$k = 0$	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
ℓ-tuple length	n	$n/2$	$n/3$	$n/4$	$n/5$	$n/6$

Shorter **perfect matches** required

Performance **decreases**

Local alignment is too slow…

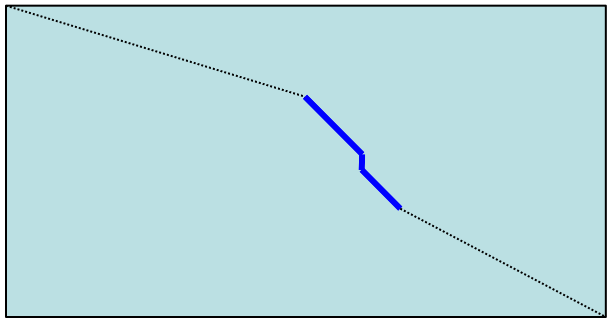
- Quadratic local alignment is too slow when looking for similarities between long strings (e.g. the entire GenBank database)
- Guaranteed to find the optimal local alignment
- Sets the standard for sensitivity

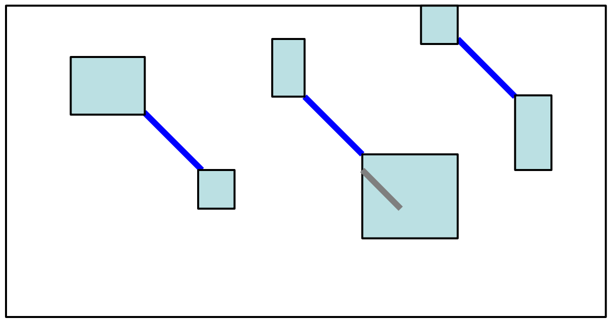
$$s_{i,j} = \max \begin{cases} 0 \\ s_{i-1,j} + \delta(v_i, -) \\ s_{i,j-1} + \delta(-, w_j) \\ s_{i-1,j-1} + \delta(v_i, w_j) \end{cases}$$

- **B**asic **L**ocal **A**lignment **S**earch **T**ool
  - Altschul, S., Gish, W., Miller, W., Myers, E. & Lipman, D.J. Journal of Mol. Biol., 1990
- Search sequence databases for local alignments to a query

BLAST

- Great improvement in speed, with only a modest decrease in sensitivity
- Opts to minimizes search space instead of exploring entire search space between two sequences
- Finds short exact matches ("seeds"), explore locally around these "hits"


Search space of Local Alignment

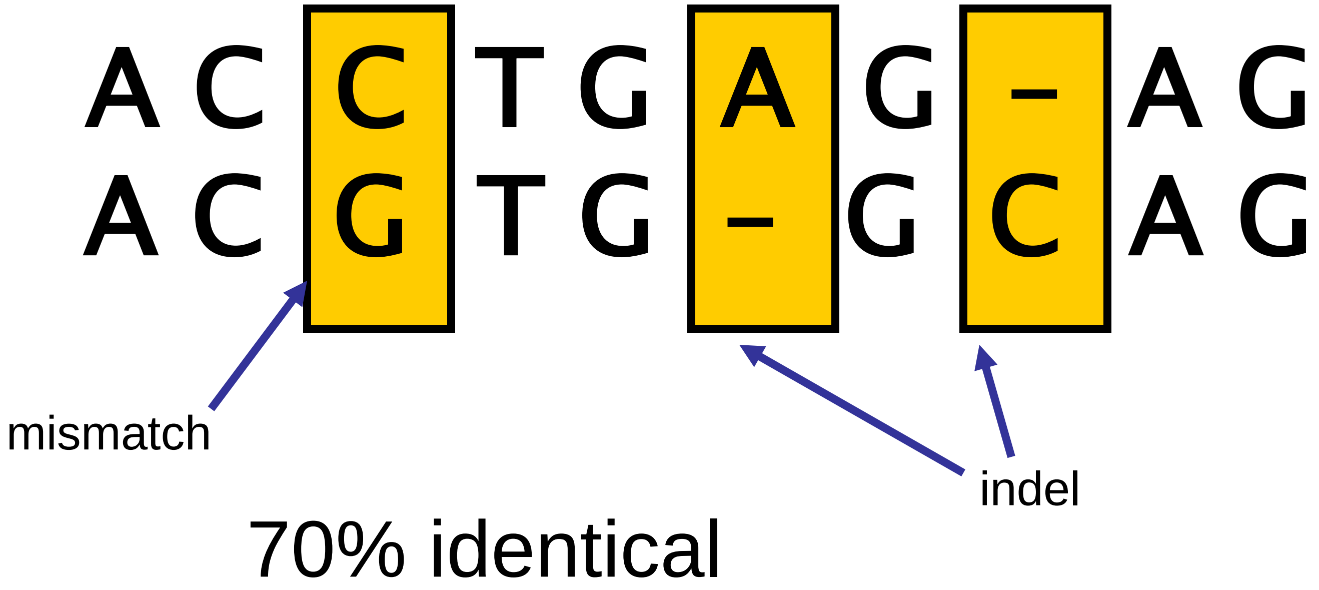

Search space of BLAST

Similarity

- BLAST only continues it's search as long as regions are sufficiently *similar*
- 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



Conservation

- Amino acid changes that preserve the physico-chemical properties of the original residue
  - Polar to polar
    - aspartate → glutamate
  - Nonpolar to nonpolar
    - alanine → valine
  - Similarly behaving residues
    - leucine to isoleucine
- Nucleotide changes that preserve molecular shape
  - Transitions (A-G, C-T) are more similar than Transversions (A-C, A-T, C-G, G-T)

Assessing Sequence Similarity

- How good of a local alignment score 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:
  - nucleotide frequency
  - dinucelotide frequency
    (e.g. C+G content in mammals)
  - common repeats
  - etc.

BLAST: Segment Score

- BLAST uses scoring matrices (δ) to improve on efficiency of match detection (we did this earlier for pairwise alignments)
  – Some proteins may have very different amino acid sequences, but are still similar (PAM, Blosum)
- For any two ℓ-mers $x_1 \ldots x_\ell$ and $y_1 \ldots y_\ell$:
  – Segment pair: pair of ℓ-mers, one from each sequence
  – Segment score: $\Sigma^{\ell}_{i=1}\, \delta(x_i, y_i)$

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 (MSPs) 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 (Poisson-distributed):

$$E(\theta) = Kmne^{-\lambda\theta}$$

K is a constant, *m* and *n* are the lengths of the two compared sequences, λ 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*, δ is the scoring matrix, and *A* is the twenty letter amino acid alphabet

P-values

- The probability of finding exactly k MSPs with a score ≥ θ is given by:

$$(E(\theta)^k e^{-E(\theta)})/k!$$

- For $k = 0$, that chance is:

$$e^{-E(\theta)}$$

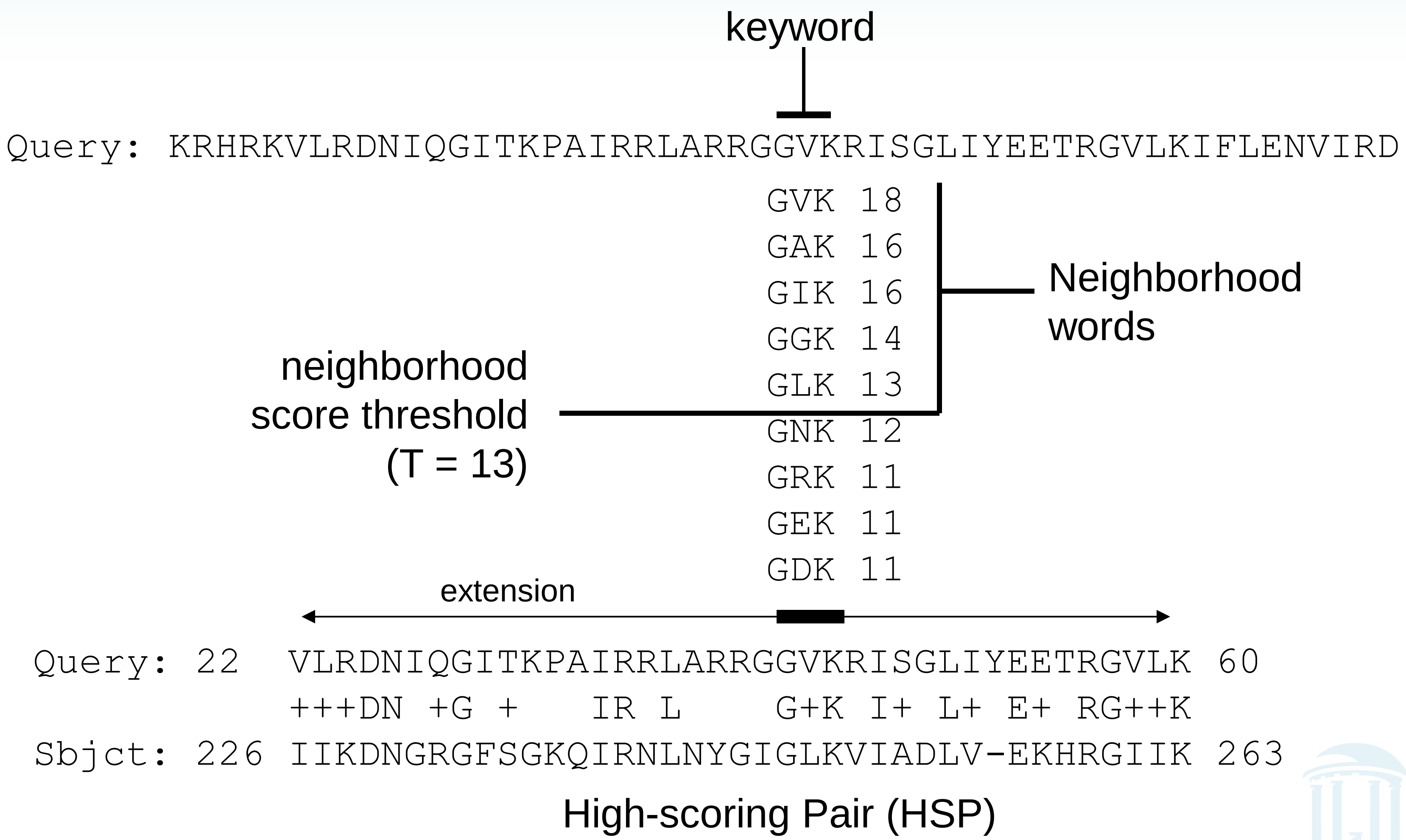
- Thus the probability of finding at least one MSP with a score ≥ θ is:

$$p(MSP > 0) = 1 - e^{-E(\theta)}$$

BLAST algorithm

- Keyword search of all substrings 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: KRHRKVLRDNIQGITKPAIRRLARRGGVKRISGLIYEETRGVLKIFLENVIRD
```

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

```
Query: 22  VLRDNIQGITKPAIRRLARRGGVKRISGLIYEETRGVLK 60
           +++DN +G +   IR L    G+K I+ L+ E+ RG++K
Sbjct: 226 IIKDNGRGFSGKQIRNLNYGIGLKVIADLV-EKHRGIIK 263
```

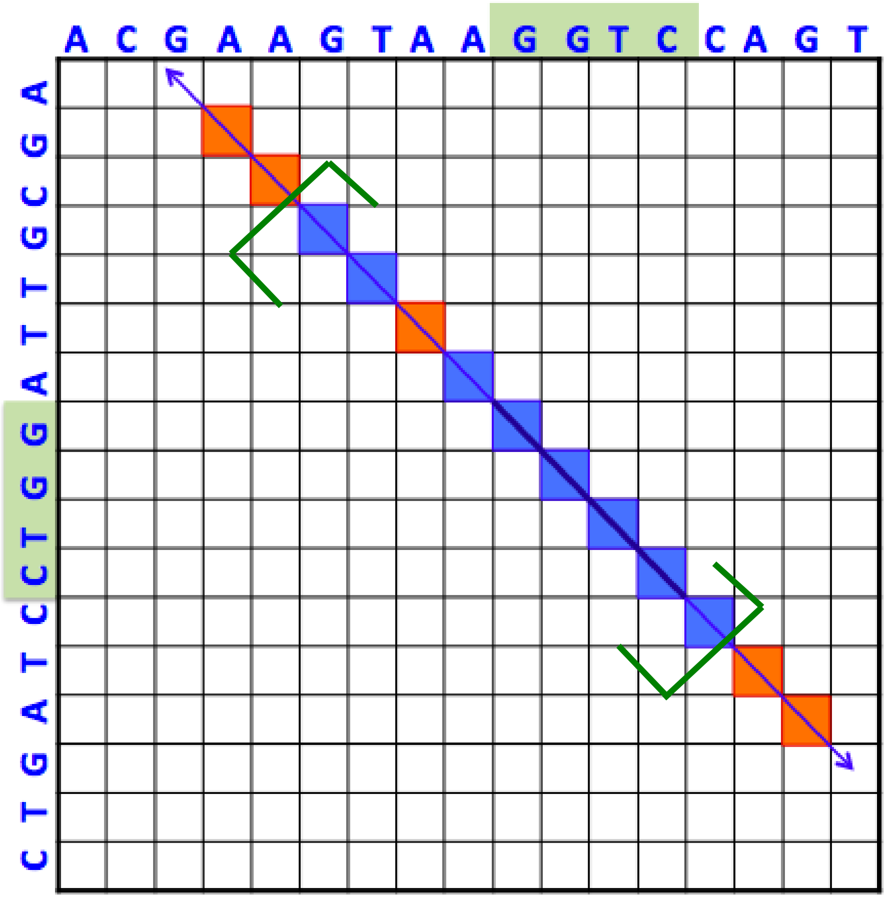
Original BLAST

- **Dictionary**
  - All words of length w
- **Alignment**
  - *Ungapped* extensions until score falls below some statistical threshold
- **Output**
  - All local alignments with score > threshold

Original BLAST: Example

- $w = 4$
- Exact keyword match of **GGTC**
- Extend diagonals with mismatches until score is under some threshold (65%)
- Trim until all mismatches are interior
- Output result:

```
GTAAGGTCC
|| ||||||
GTTAGGTCC
```

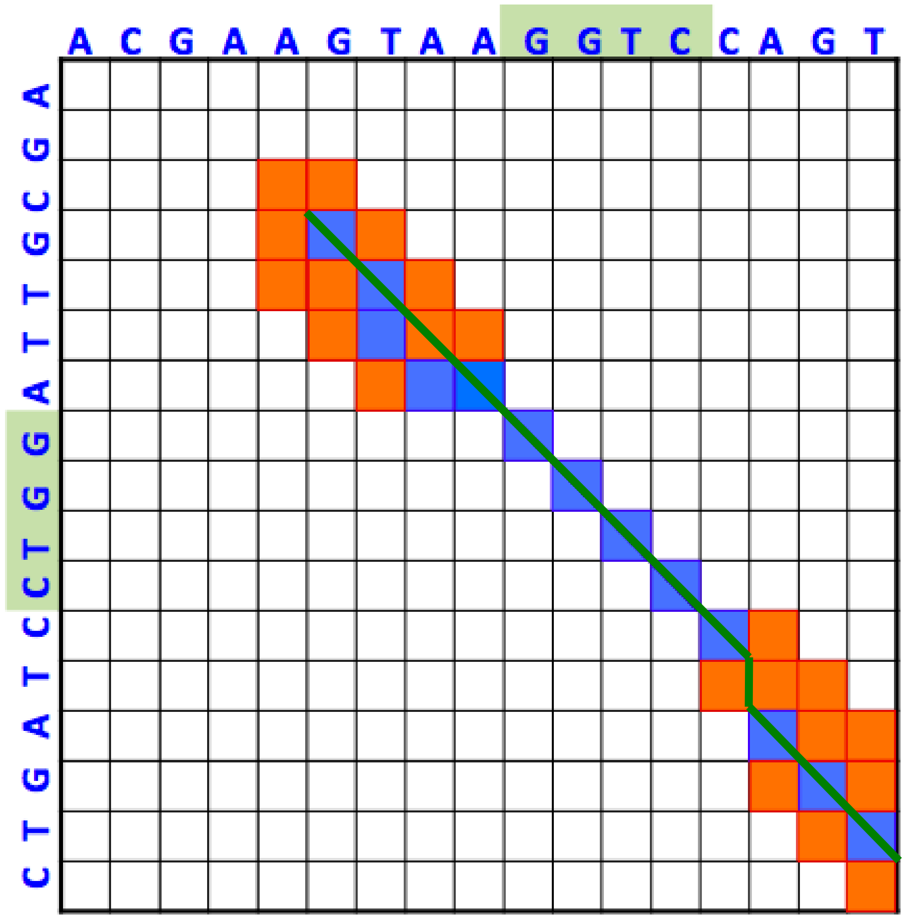

A C G A A G T A A G G T C C A G T
A G C G T T A G G T C C T A T C G T C
C T G A T C C T G G A T T G C G A

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
```


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

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

Sample BLAST output

- Blast of human beta globin protein against zebra fish

```
                                                                  Score    E
Sequences producing significant alignments:                      (bits) Value

gi|18858329|ref|NP_571095.1| ba1 globin [Danio rerio] >gi|147757...   171   3e-44
gi|18858331|ref|NP_571096.1| ba2 globin; SI:dZ118J2.3 [Danio rer...   170   7e-44
gi|37606100|emb|CAE48992.1| SI:bY187G17.6 (novel beta globin) [D...   170   7e-44
gi|31419195|gb|AAH53176.1| Ba1 protein [Danio rerio]                  168   3e-43

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

Query: 61  VKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVLVCVLAHHFG 120
           V AHG+ V+G     + ++DN+K T+A LS +H +KLHVDP+NFRLL + +    A  FG
Sbjct: 61  VAAHGRTVMGGLERAIKNMDNVKNTYAALSVMHSEKLHVDPDNFRLLADCITVCAAMKFG 120

Query: 121 KE-FTPPVQAAYQKVVAGVANALAHKYH 147
           +  F   VQ A+QK +A V +AL  +YH
Sbjct: 121 QAGFNADVQEAWQKFLAVVVSALCRQYH 148
```

Sample BLAST output (cont'd)

- Blast of human beta globin DNA against human DNA

```
                                                                             Score    E
Sequences producing significant alignments:                      (bits) Value

gi|19849266|gb|AF487523.1| Homo sapiens gamma A hemoglobin (HBG1...   289   1e-75
gi|183868|gb|M11427.1|HUMHBG3E Human gamma-globin mRNA, 3' end        289   1e-75
gi|44887617|gb|AY534688.1| Homo sapiens A-gamma globin (HBG1) ge...   280   1e-72
gi|31726|emb|V00512.1|HSGGL1 Human messenger RNA for gamma-globin     260   1e-66
gi|38683401|ref|NR_001589.1| Homo sapiens hemoglobin, beta pseud...   151   7e-34
gi|18462073|gb|AF339400.1| Homo sapiens haplotype PB26 beta-glob...   149   3e-33

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
             || ||| | ||    | || |  |||||| ||||| |||||||||||    ||||||||
Sbjct: 54409 ttcggaaaagctgttatgctcacggatgacctcaaaggcacctttgctacactgagtgac 54468


Query: 327   ctgcactgtgacaagctgcatgtggatcctgagaacttc 365
             ||||||||| |||||||||| ||||| ||||||||||||
Sbjct: 54469 ctgcactgtaacaagctgcacgtggaccctgagaacttc 54507
```

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:

```
GAGTACTCAACACCAACATTAGTGGCAATGGAAAAT…
|| ||||||||| ||||| || |||||   ||||||
GAATACTCAACAGCAACACTAATGGCAGCAGAAAAT…
       111010010100110111
```

How does this result in better sensitivity?

Why is PH better?

- BLAST
  redundant hits

```
TTGACCTCACC?
|||||||||||?
TTGACCTCACC?
11111111111
 11111111111
```

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

- PatternHunter

```
CAA?A??A?C??TA?TGG?
|||?|??|?|??||?|||?
CAA?A??A?C??TA?TGG?
111010010100110111
 111010010100110111
```

This results in very few redundant hits

Why is PH better?

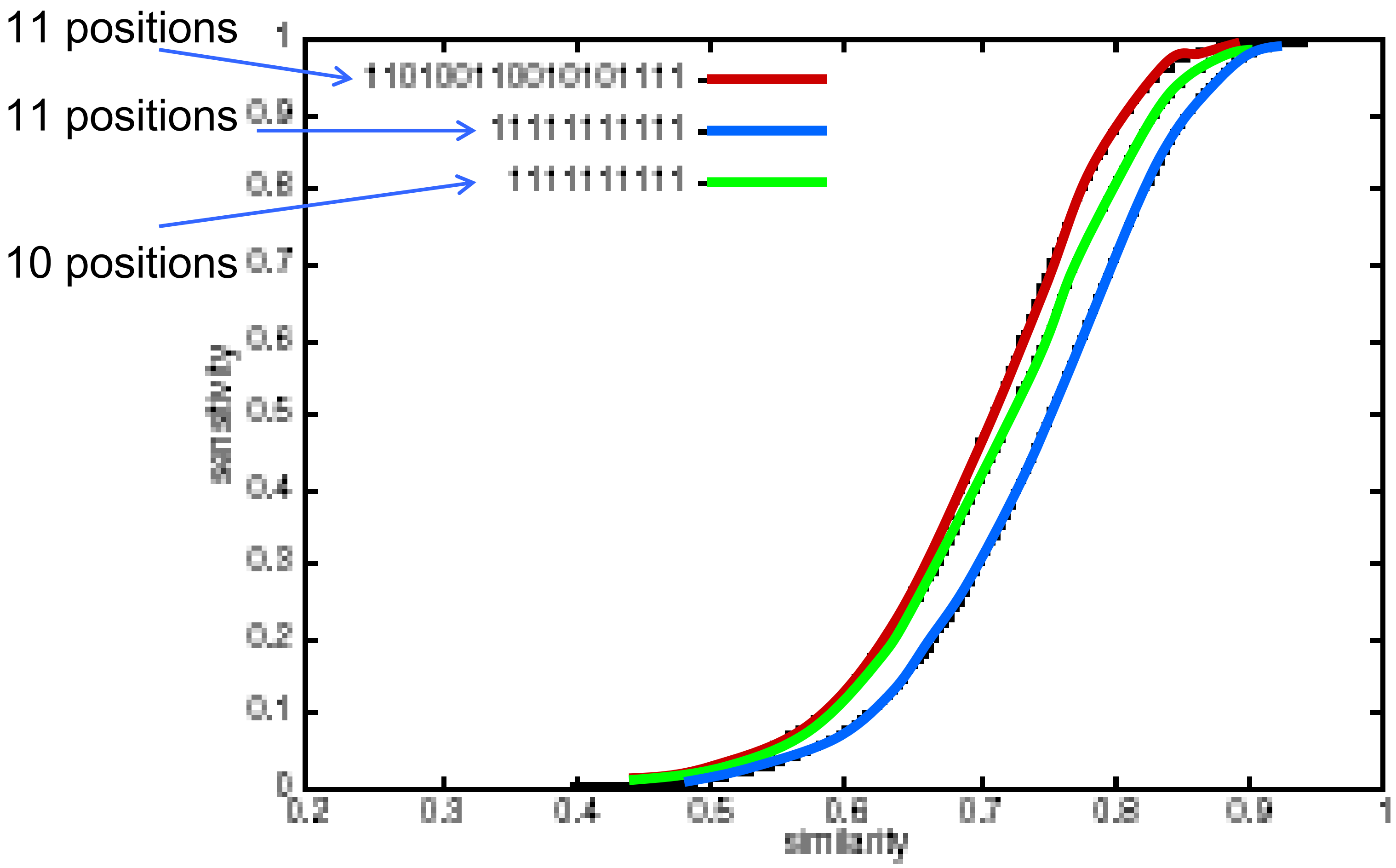
BLAST may also miss a hit

```
GAGTACTCAACACCAACATTAGTGGGCAATGGAAAAT
|| ||||||||| |||||| | ||||||   ||||||
GAATACTCAACAGCAACATCAATGGGCAGCAGAAAAT
   <------->
   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



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