Biological Database Searching :FASTA, BLAST

Database Searching

- Dynamic programming algorithms give “correct” solutions but is very slow unless the sequences are quite short.
- Current common protein sequence data bases contain more than 100 M residues, For a “query” sequence of 1000 residues, we need to evaluate 10^{11} matrix cells. Even if you compute 10 M cells /second, it will take 10^4 secs =~ 3 hour just for one query.
- Goal: search small fraction of the possible high scoring alignments.
- The vast literature on exact and approximate match algorithms can be used. But with scoring matrices, distant matches are hard to find.
- Need heuristic algorithms: FASTA and BLAST are two such classes of algorithms, BLAST is more popular but we still use the FASTA data format.

Database searching

- Core: pair-wise alignment algorithm
- Speed (fast sequence comparison)
- Relevance of the search results (statistical tests)
- Recovering all information of interest
  - The results depend of the search parameters like gap penalty, scoring matrix.
  - Sometimes searches with more than one matrix should be preformed

The Basic Idea

- "Filtration": find short sequences in the database that matches with segments of query sequence.
- Use these short stretches of matches (using a hash table or a suffix tree) as "seeds" from which extend out to get good longer alignments.
- **Approximate Pattern Matching Problem:**

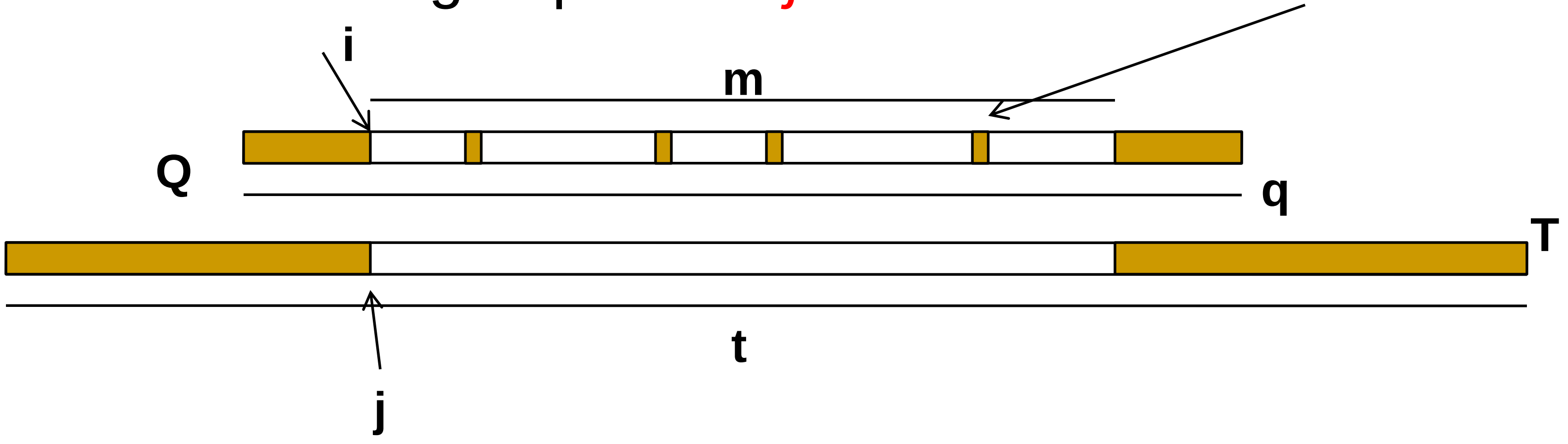
Given a pattern $P=p_1p_2 . \; . \; .p_m$ and a text $T=t_1t_2 . \; . \; .t_n$, $m<=n$, find all positions in the text such that P occurs in T beginning these positions having at most k mismatches.

Approximate Query Matching

Find all substrings of a query that approximately matches a text string. Given

1. two integers *m* and *k*
2. a query of length *q*, $Q=x_1x_2\ldots x_q$
3. a text string of length *n*, $T=t_1t_2\ldots t_n$

Find all pairs of positions (*i*,*j*), $1<= i <=q-m+1$ and $1<=j<=n-m+1$ such that the *m*-letter substring of *Q* starting at position *i* approximately matches the *m*-letter substring of *T* starting at position *j* with at most *k* mismatches.

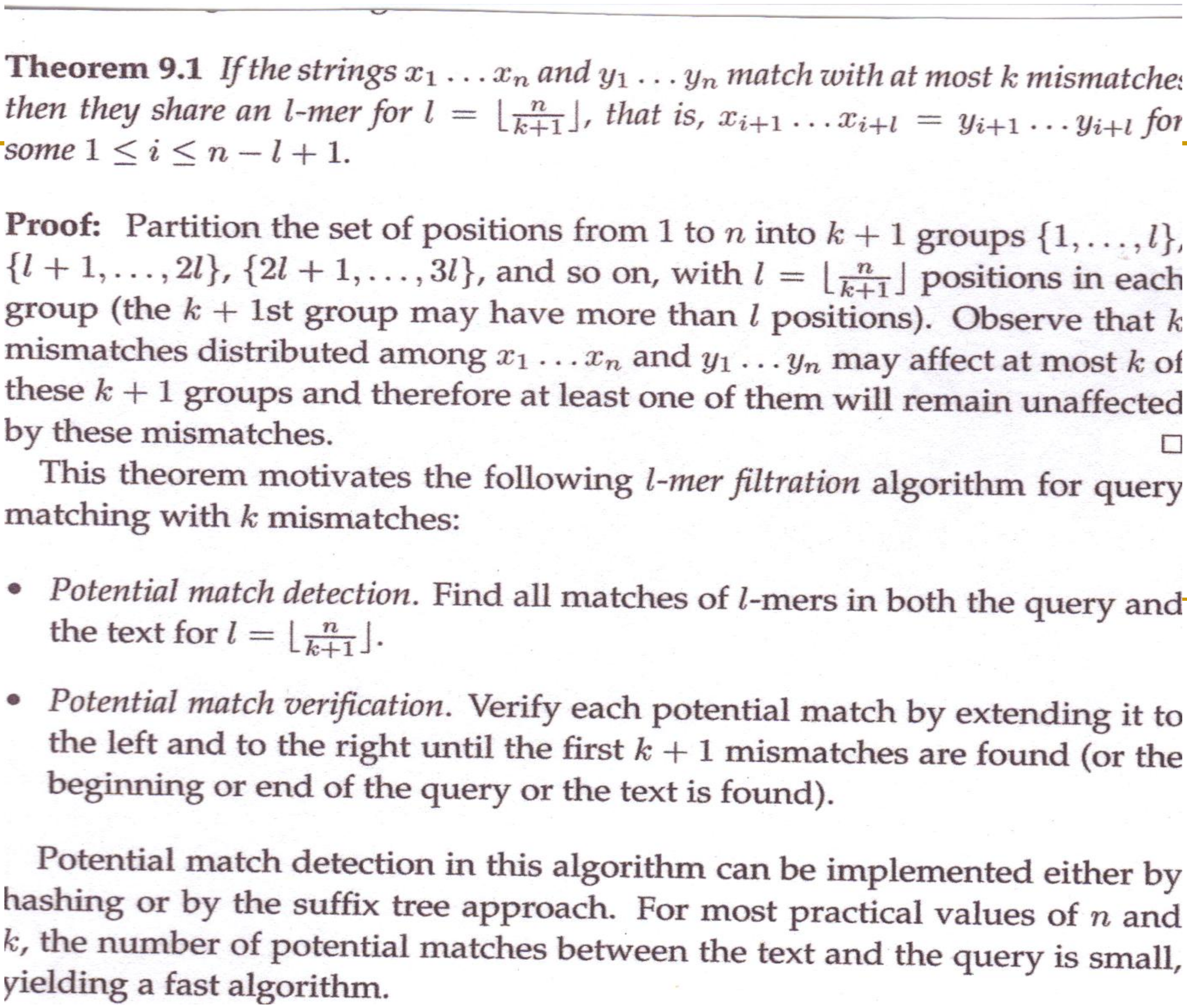


Filtration Algorithm

- Stage 1: Preselect a set of positions in the text that are potentially similar to the query.
- Stage 2: Verify each potential position and reject it if the number of mismatches exceeds *k.*

The underlying idea is that if an *m*-letter substring of *Q* approximately matches a *m*-letter substring of *T,* then the two substrings share at least one *l*-mer for appropriately large value of *l* given in the following theorem. All *l*-mers shared shared by the query and the text can be found by hashing. If the number of shared *l*-mers is relatively small, then all matches with a maximum of *k* mismatches can be found rapidly. [In the text, the symbol *n* has been used for *m*. Make a note of this.]

An Introduction to Bioinformatics



Theorem 9.1 *If the strings $x_1 \ldots x_n$ and $y_1 \ldots y_n$ match with at most k mismatche then they share an l-mer for $l = \lfloor \frac{n}{k+1} \rfloor$, that is, $x_{i+1} \ldots x_{i+l} = y_{i+1} \ldots y_{i+l}$ for some $1 \leq i \leq n - l + 1$.*

Proof: Partition the set of positions from 1 to n into $k+1$ groups $\{1, \ldots, l\}$, $\{l+1, \ldots, 2l\}$, $\{2l+1, \ldots, 3l\}$, and so on, with $l = \lfloor \frac{n}{k+1} \rfloor$ positions in each group (the $k+1$st group may have more than l positions). Observe that k mismatches distributed among $x_1 \ldots x_n$ and $y_1 \ldots y_n$ may affect at most k of these $k+1$ groups and therefore at least one of them will remain unaffected by these mismatches. □

This theorem motivates the following *l-mer filtration* algorithm for query matching with k mismatches:

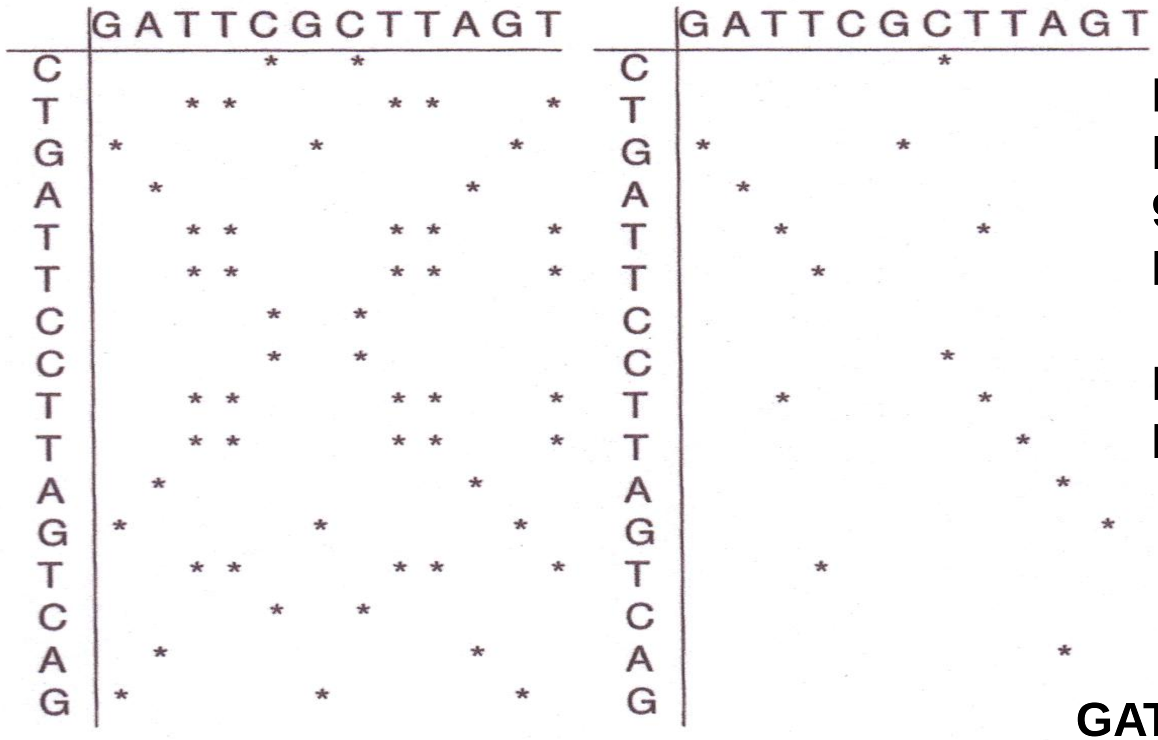
- *Potential match detection.* Find all matches of l-mers in both the query and the text for $l = \lfloor \frac{n}{k+1} \rfloor$.
- *Potential match verification.* Verify each potential match by extending it to the left and to the right until the first $k+1$ mismatches are found (or the beginning or end of the query or the text is found).

Potential match detection in this algorithm can be implemented either by hashing or by the suffix tree approach. For most practical values of n and k, the number of potential matches between the text and the query is small, yielding a fast algorithm.

FASTA

FASTA uses “dot matrix” to identify short segments of exact matches. A dot matrix is a matrix of 0’s and 1’s in which a 1 in the (i,j) position indicates some similarity. The similarity measures may vary. We could use seed length t=2 and allow k=1 or 2 mismatches.

An Intro



Note, nearly Diagonal run of 9 dots that Indicate an approximate Match with one Insertion.

GATTC—CTTAGT
GATTC G CTTAGT

Figure 9.7 Two dot matrices for the strings CTGATTCCTTAGTCAG GATTCGCTTAGT. A dot in position (i, j) of the first matrix indicates that th nucleotide of the first sequence matches the jth nucleotide of the second. A d position (i, j) of the second matrix indicates that the ith dinucleotide in the firs quence and jth dinucleotide in the second sequence are the same, i.e., the ith $(i + 1)$-th symbols of the first sequence match the jth and $(j + 1)$-th symbols c second sequence. The second matrix reveals a nearly diagonal run of 9 dots points to an approximate match (with one insertion) between substring GATT TAGT in the first sequence and substring GATTCGCTTAG of the second one. same diagonal run is present in the first matrix but disguised by many spurious

Database searching

- Goal: find similar sequences of a query sequence in a sequence of database
- Input: query sequence & database
- Output: hits (pairwise alignments)

What program to use for searching?

1) **BLAST** is fastest and easily accessed on the Web
   - limited sets of databases
   - nice translation tools (BLASTX, TBLASTN)

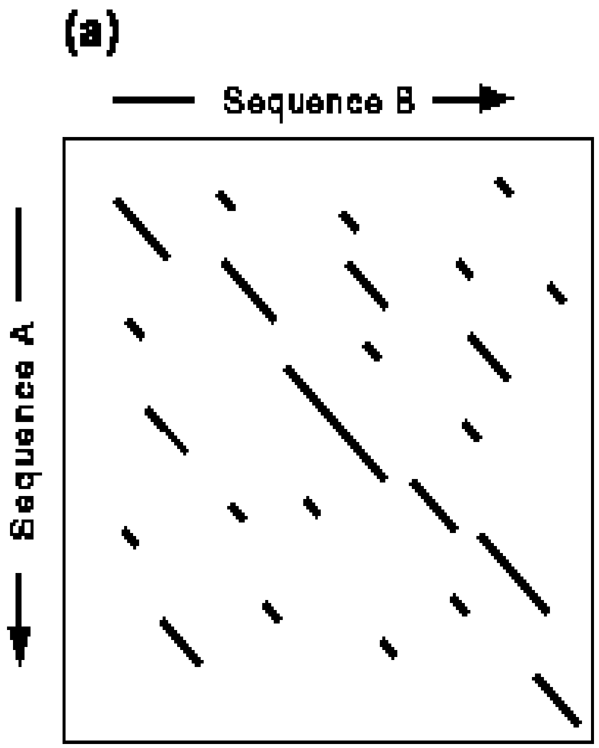
2) **FASTA**
   - precise choice of databases
   - more sensitive for DNA-DNA comparisons
   - **FASTX** and **TFASTX** can find similarities in sequences with frameshifts

3) Smith-Waterman is slower, but more sensitive
   - known as a “rigorous” or “exhaustive” search
   - **SSEARCH** in **GCG** and standalone **FASTA**

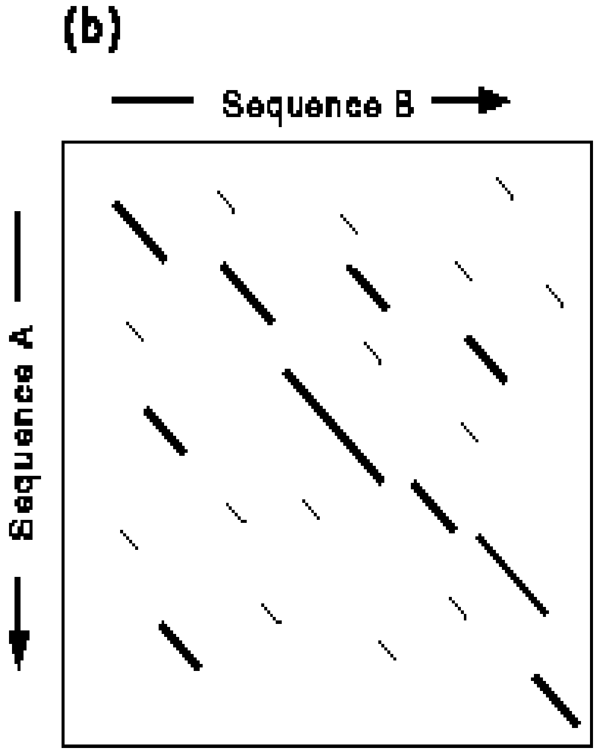
FASTA

1) Derived from logic of the dot plot
   - compute best diagonals from all frames of alignment
2) Word method looks for exact matches between words in query and test sequence
   - hash tables (fast computer technique)
   - DNA words are usually 6 bases
   - protein words are 1 or 2 amino acids
   - only searches for diagonals in region of word matches, leads to faster searching

FASTA Algorithm (Using score matrix)



Find runs of identitical words



Re-score using PAM matrix

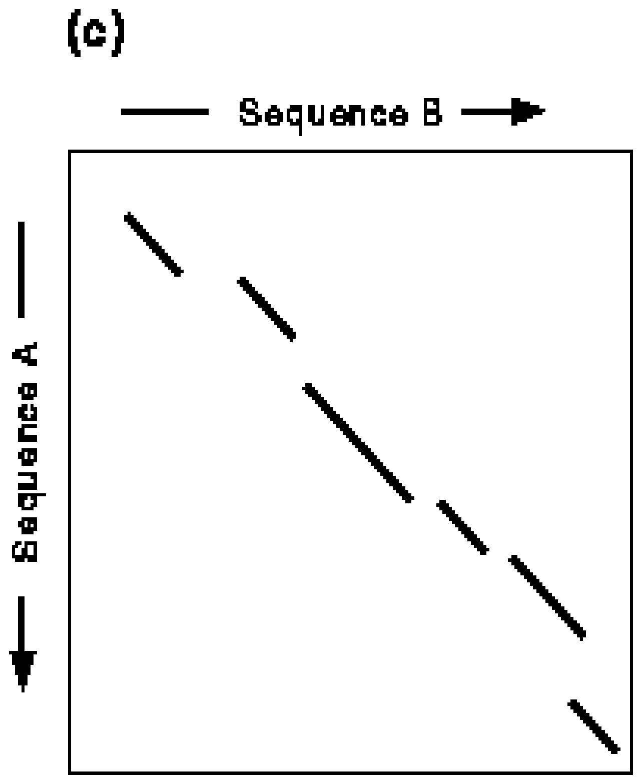
Keep top scoring segments

Makes Longest Diagonal

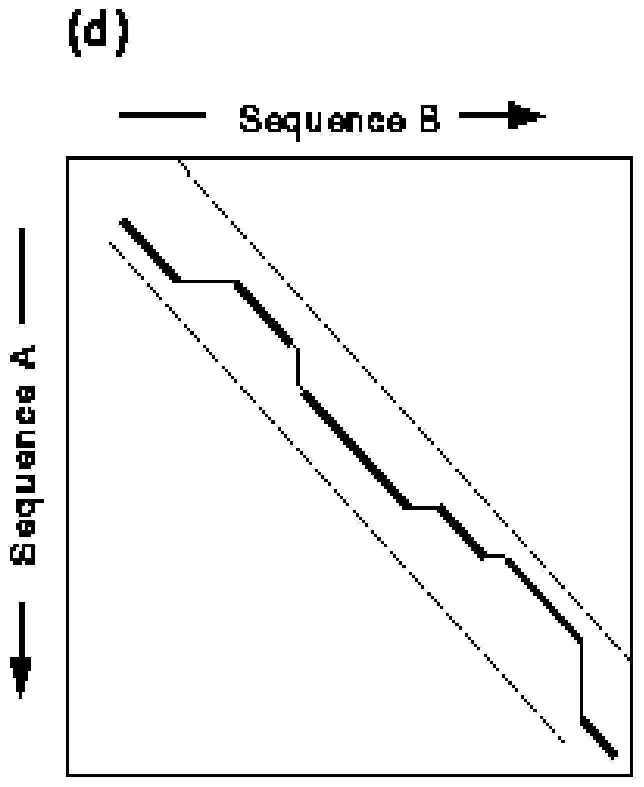
3) after all diagonals found, tries to join diagonals by adding gaps

4) computes alignments in regions of best diagonals

FASTA Alignments



Join segments using gaps, eliminate other segments



Use dynamic programming to create an optimal alignment

BLAST Searches GenBank

[**BLAST**= **B**asic **L**ocal **A**lignment **S**earch **T**ool]

The NCBI **BLAST** web server lets you compare your query sequence to various sections of GenBank:

- **nr** = non-redundant (main sections)
- **month** = new sequences from the past few weeks
- **ESTs**
- human, drososphila, yeast, or E.coli genomes
- proteins (by automatic translation)

- This is a VERY fast and powerful computer.

BLAST

- Uses word matching like FASTA
- Similarity matching of words
  - does not require identical words.
- If no words are similar, then no alignment
  - won’t find matches for very short sequences
- Does not handle gaps well
- New “gapped BLAST” (BLAST 2) is better

1. **FASTA cannot handle the situation when two proteins have different amino acid sequences but are biologically similar. The redundant codons for the same protein may evolve but continue to produce same function for the protein. This is handled in BLAST using scoring matrices for totally "dissimilar" amino acid sequences.**

2 . BLAST allows models for estimating statistical significance of the matches found.

A segment pair is a pair of *l*-mers ($x_1,x_2,\ldots,x_l$)and ($y_1,y_2,..,y_l$) one from each sequence The segment score is given by $\sum_{i=1}^{l}\delta(x_i,y_i)$ where $\delta(x,y)$ is the similarity score between amino acids *x* and *y.*

If the segment score is best over all segment pairs in the two sequences then The segment is called maximal segment pair.

A segment pair is locally maximal if the score cannot be improved extending or by shortening both segments.

A researcher is interested in all statistically significant locally maximal segment pairs. Typically, a threshold score is associated with each search Query. By making threshold very high, BLAST can do multiple pattern matching.

Indexing-based local alignment

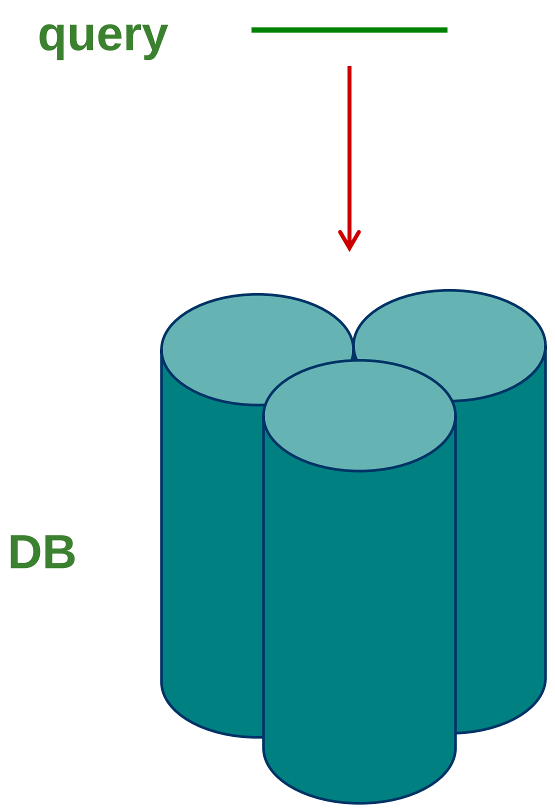
(BLAST- **B**asic **L**ocal **A**lignment **S**earch **T**ool)

Main idea:

1. Construct a dictionary of all the words in the query
2. Initiate a local alignment for each word match between query and DB

Running Time: O(mn)

However, orders of magnitude faster than Smith-Waterman



Indexing-based local alignment

Dictionary:

All words of length k (~10)

Alignment initiated between

words of alignment score ≥ Th

(typically Th = k)

Alignment:

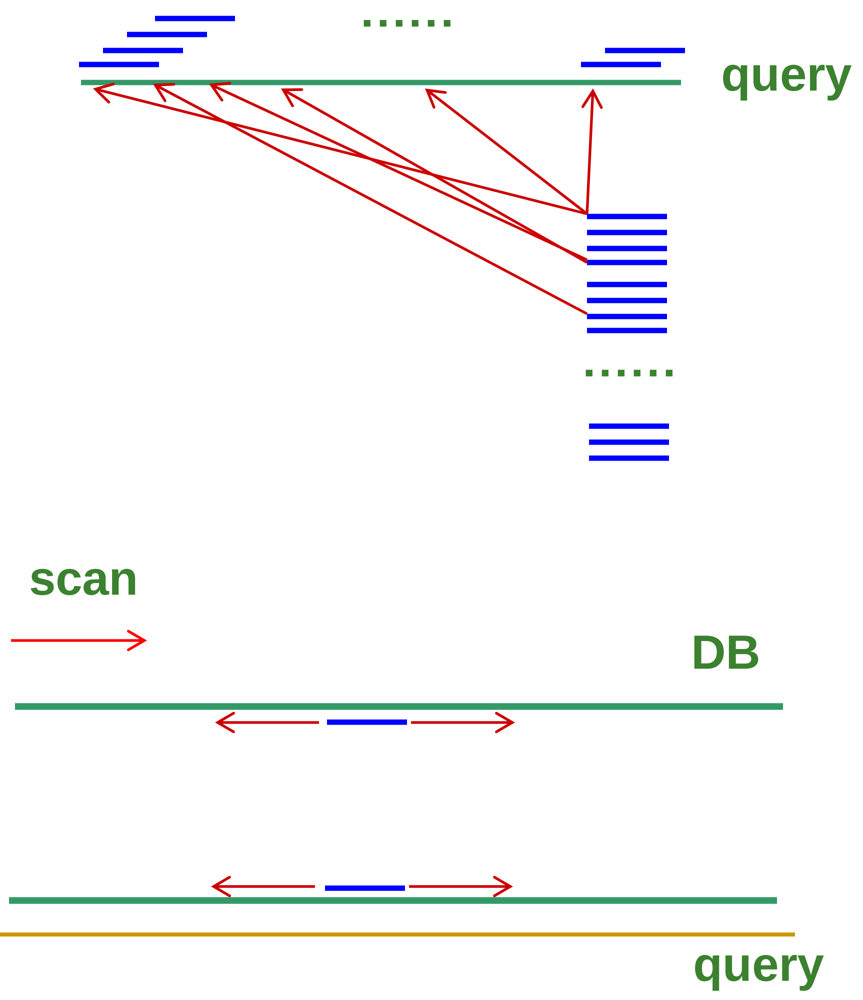
Ungapped extensions until score

below statistical threshold

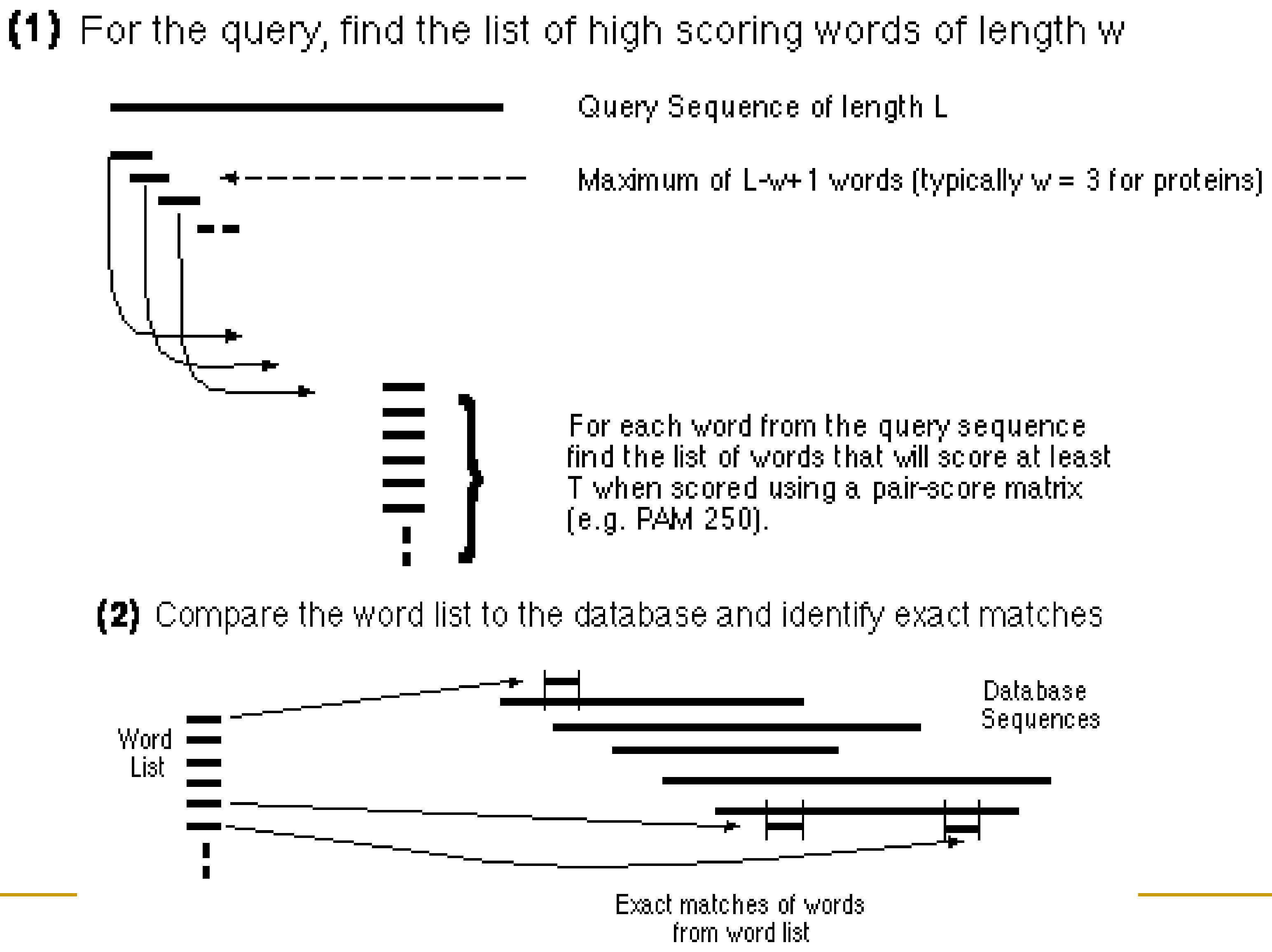
Output:

All local alignments with score

> statistical threshold



BLAST Algorithm



(1) For the query, find the list of high scoring words of length w

Query Sequence of length L
Maximum of L-w+1 words (typically w = 3 for proteins)

For each word from the query sequence find the list of words that will score at least T when scored using a pair-score matrix (e.g. PAM 250).

(2) Compare the word list to the database and identify exact matches

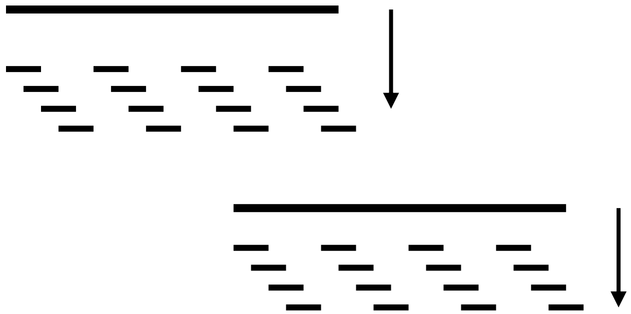
Word List
Database Sequences
Exact matches of words from word list

BLAST Word Matching

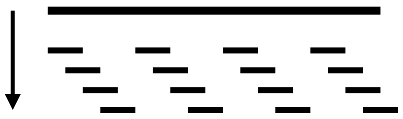
```
MEAAVKEEISVEDEAVDKNI

MEA
 EAA
  AAV
   AVK
    VKE
     KEE
      EEI
       EIS
        ISV
         ...
```

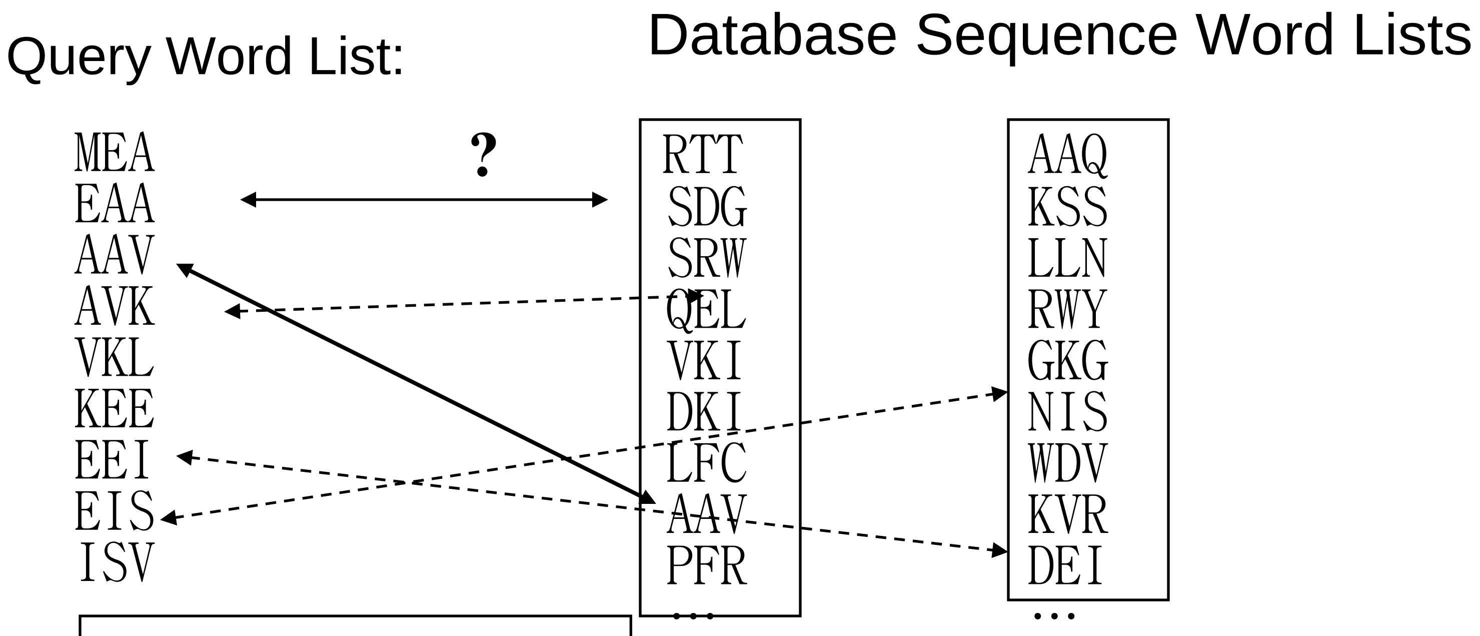
Break query into words:



Break database sequences into words:

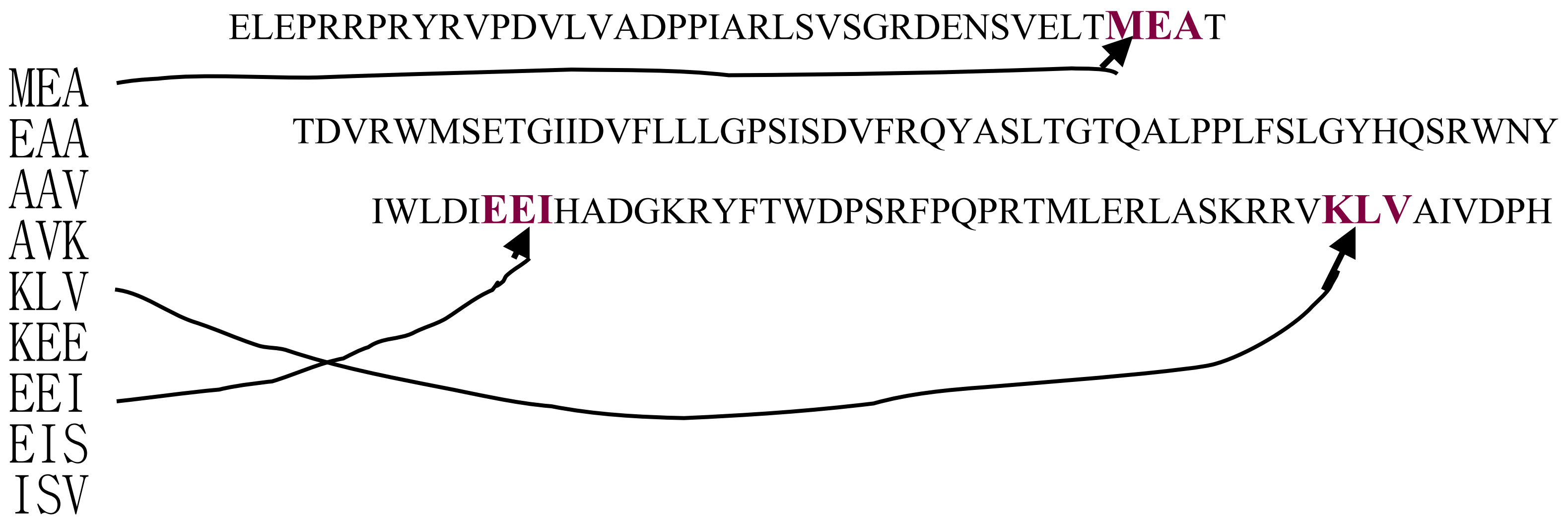


Compare Word Lists



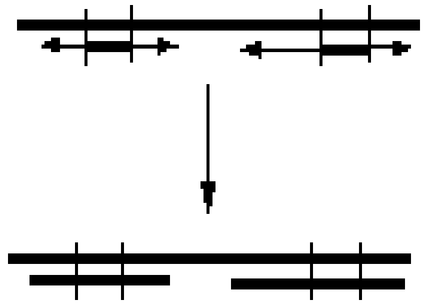
Compare word lists by Hashing
(allow near matches)

Find locations of matching words in database sequences


ELEPRRPRYRVPDVLVADPPIARLSVSGRDENSVELTMEAT
TDVRWMSETGIIDVFLLLGPSISDVFRQYASLTGTQALPPLFSLGYHQSRWNY
IWLDIEEIHADGKRYFTWDPSRFPQPRTMLERLASKRRVKLVAIVDPH
MEA
EAA
AAV
AVK
KLV
KEE
EEI
EIS
ISV

Extend hits one base at a time

(3) For each word match, extend the alignment in both directions to find alignments that score greater than a threshold of value S



Maximal Segment Pairs (MSPs)

Figure from Barton, G.J. Protein Sequence Alignment and Database Scanning (University of Oxford, Laboratory of Molecular Biophysics)

Indexing-based local alignment—Extensions

Example:

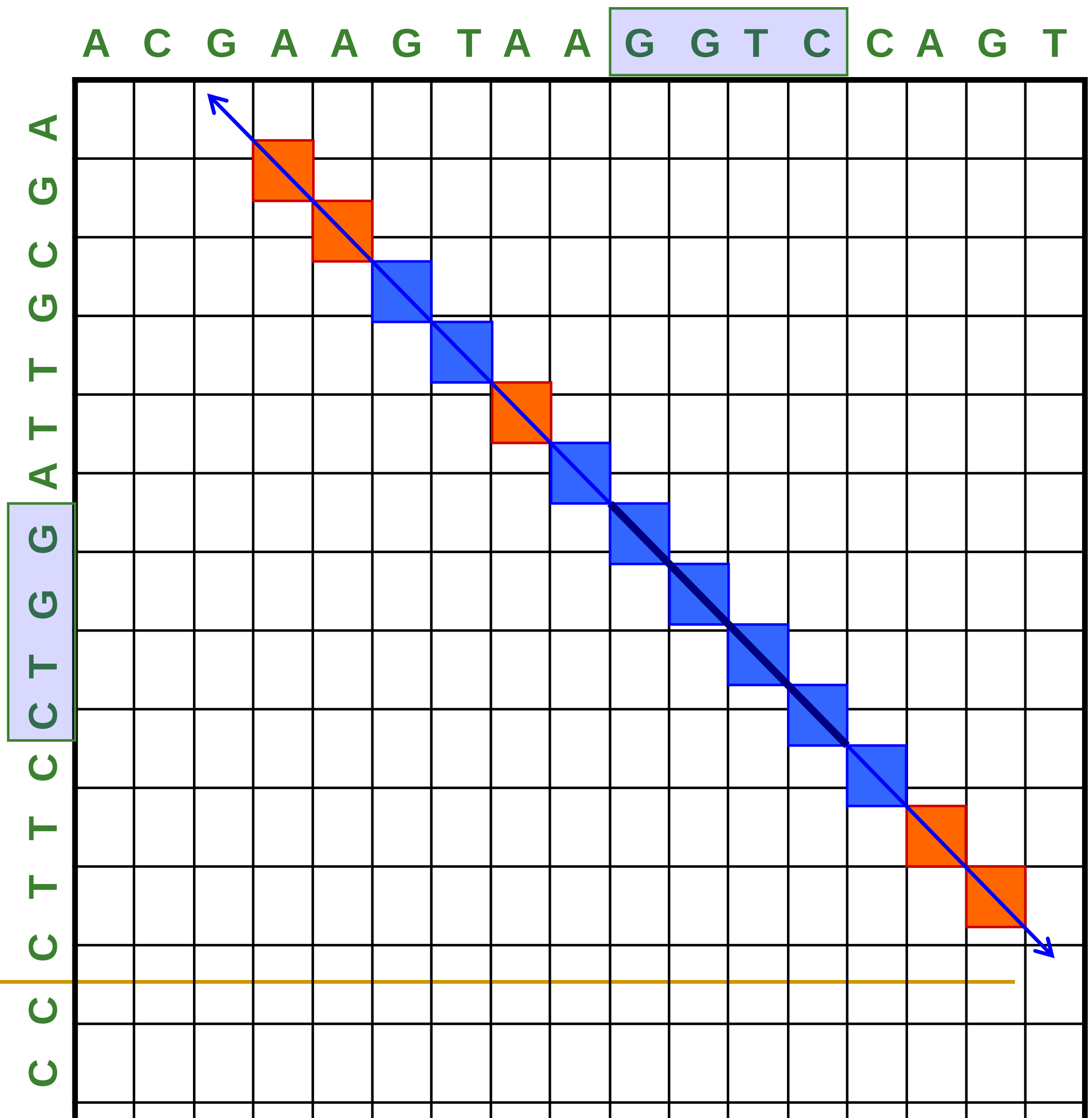
k = 4

The matching word GGTC initiates an alignment

Extension to the left and right with no gaps until alignment falls < Th below best so far

Output:

```
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 T C C C

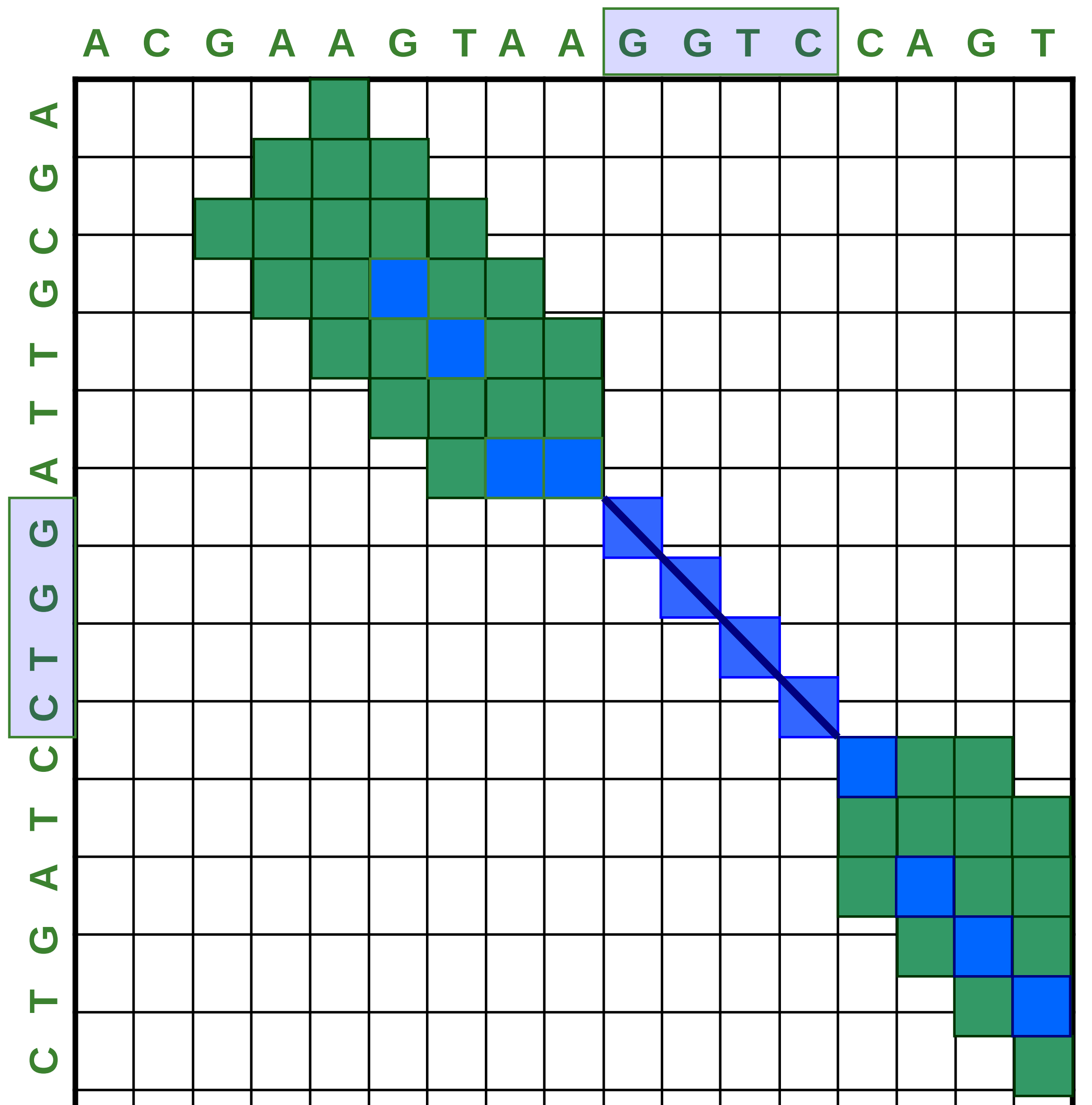
Indexing-based local alignment—Extensions

Gapped extensions

- Extensions with gaps in a band around anchor

Output:

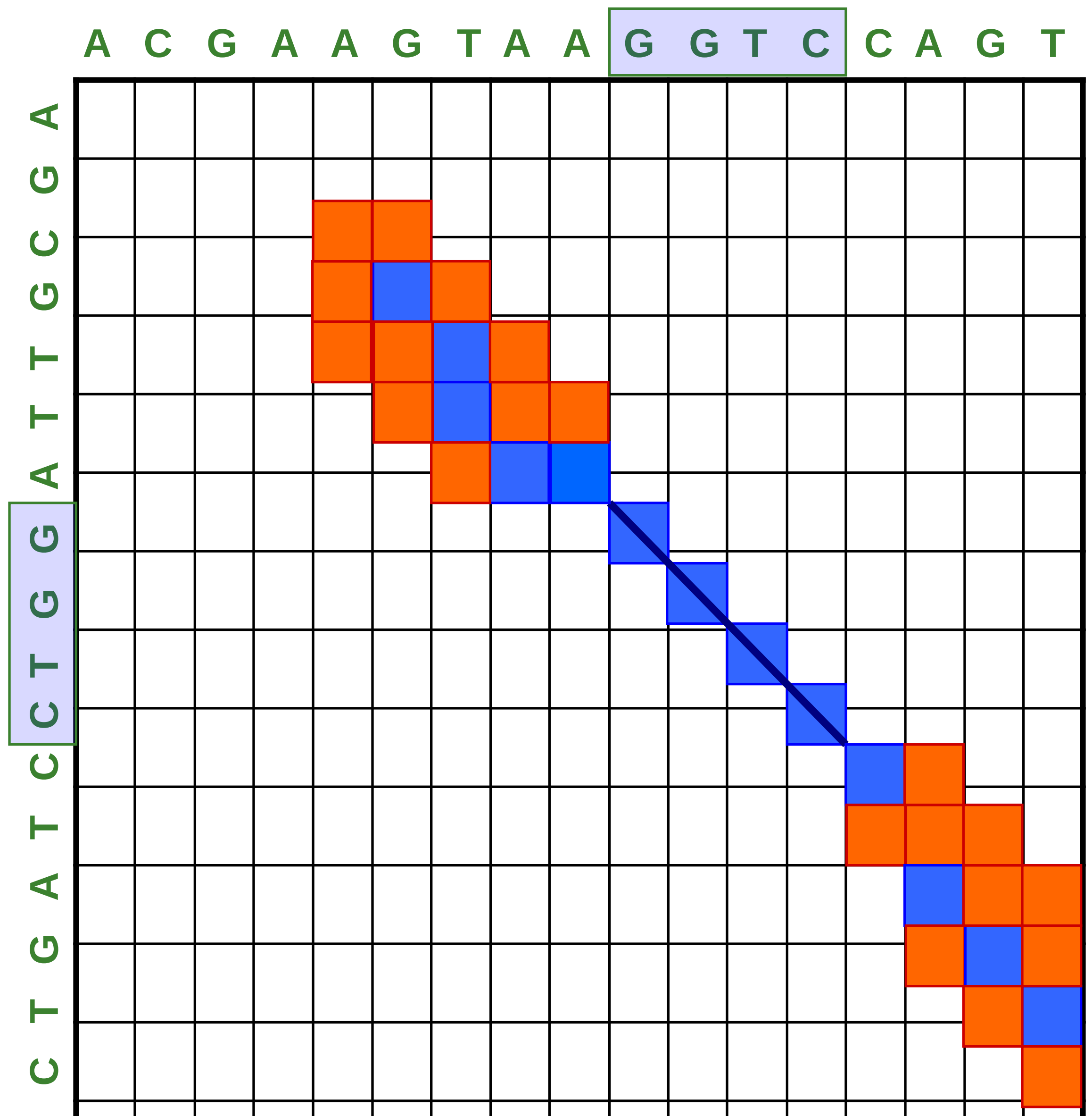
```
GTAAGGTCCAGT
GTTAGGTC-AGT
```


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

Indexing-based local alignment—Extensions

<u>Gapped extensions until threshold</u>

- Extensions with gaps until score < Th below best score so far


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

<u>Output:</u>

```
GTAAGGTCCAGT
GTTAGGTC-AGT
```

```
Seq_XYZ:  HVTGRSAF_FSYYGYGCYCGLGTGKGLPVDATDRCCWA
                 | |    |||   |  || | ||   | ||
Query:        QSVFDYIYYGCYCGWGLG_GK__PRDA
```

E-val=10^{-13}

- Use **<u>two</u>** word matches as anchors to build an alignment between the query and a database sequence.
- Then score the alignment.

HSPs are Aligned Regions

A local alignment without gaps consists simply of a pair of equal length segments, one from each of the two sequences being compared. A modification of the Smith-Waterman [7] or Sellers [8] algorithms will find all segment pairs whose scores can not be improved by extension or trimming. These are called high-scoring segment pairs or HSPs.

- The results of the word matching and attempts to extend the alignment are segments

  - called HSPs (High-scoring Segment Pairs)
- **BLAST** often produces several short HSPs rather than a single aligned region

BLAST 2 algorithm

- The NCBI's BLAST website now both use BLAST 2 (also known as "gapped BLAST")
- This algorithm is more complex than the original BLAST
- It requires two word matches close to each other on a pair of sequences (i.e. with a gap) before it creates an alignment

BLAST **Statistics**

- E value is equivalent to standard P value (based on Karlin-Altschul theorem)
- Significant if E < 0.05 (smaller numbers are more significant)
  - The E-value represents the likelihood that the observed alignment is due to chance alone. A value of 1 indicates that an alignment this good would happen by chance with any random sequence searched against this database.

Altschul-Dembo-Karlin Statistics
(Read on- line tutorial . Optional reading: Posted paper)

The number of matches with scores above threshold θ is approximately Poisson distribution given by $E(\theta) = Kmne^{-\theta\lambda}$ where *K* is a constant and λ is a positive root of

$$\sum_{x,y \varepsilon A} p_x p_y e^{\lambda\delta(x,y)} = 1$$

where p_x and p_y are frequencies of amino acids *x* and *y* from the *20* letter alphabet *A and* δ *is the scoring matrix.*

The probability that there is a match of score greater than θ *between the two random sequences of length n and m is computed as* $1 - e^{E(\theta)}$ *. This is used as a guide for the choice of BLAST parameters and evaluate the statistical significance of the matches.*

BLAST variants for different searches

(after S. Brenner, Trends Guide to Bioinformatics, 1998)

Program	Query	Database	Comparison	Common use
blastn	DNA	DNA	DNA level	Seek identical DNA sequences and splicing patterns
blastp	Protein	Protein	Protein level	Find homologous proteins
blastx	DNA	Protein	Protein level	Analyze new DNA to find genes and seek homologous proteins
tblastn	Protein	DNA	Protein level	Search for genes in unannotated DNA
tblastx	DNA	DNA	Protein level	Discover gene structure

BLAST is Approximate

- BLAST makes similarity searches very quickly because it takes shortcuts.
  - looks for short, nearly identical “words” (11 bases)
- It also makes errors
  - misses some important similarities
  - makes many incorrect matches
    - easily fooled by repeats or skewed composition

Interpretation of output

- very low E values (e-100) are homologs or identical genes
- moderate E values are related genes
- long list of gradually declining of E values indicates a large gene family
- long regions of moderate similarity are more significant than short regions of high identity

Biological Relevance

- It is up to you, the biologist to scrutinize these alignments and determine if they are significant.
- Were you looking for a short region of nearly identical sequence or a larger region of general similarity?
- Are the mismatches conservative ones?
- Are the matching regions important structural components of the genes or just introns and flanking regions?

Borderline similarity

- What to do with matches with E() values in the 0.5 -1.0 range?
- this is the **"Twilight Zone"**
- retest these sequences and look for related hits (not just your original query sequence)
- similarity is transitive:

  if **A~B** and **B~C**, then **A~C**

Indexing-based Local Aligners

BLAST, WU-BLAST,
BlastZ,
MegaBLAST,
BLAT,
PatternHunter, ……

Variants of BLAST

- NCBI BLAST: search the universe http://www.ncbi.nlm.nih.gov/BLAST/
- MEGABLAST:
  - Optimized to align very similar sequences
    - Works best when $k = 4i \geq 16$
    - Linear gap penalty
- WU-BLAST: (Wash U BLAST)
  - Very good optimizations
  - Good set of features & command line arguments
- BLAT
  - Faster, less sensitive than BLAST
  - Good for aligning huge numbers of queries
- CHAOS
  - Uses inexact k-mers, sensitive
- PatternHunter
  - Uses patterns instead of k-mers
- BlastZ
  - Uses patterns, good for finding genes

Example

Query: gattacaccccgattacaccccgattaca (29 letters) **[2 mins]**

Database: All GenBank+EMBL+DDBJ+PDB sequences (but no EST, STS, GSS, or phase 0, 1 or 2 HTGS sequences) 1,726,556 sequences; **8,074,398,388 total letters**

>gi|28570323|gb|AC108906.9| Oryza sativa chromosome 3 BAC OSJNBa0087C10 genomic sequence, complete sequence Length = 144487 **Score = 34.2 bits** (17), Expect = 4.5 **Identities = 20/21 (95%)** Strand = Plus / Plus

```
Query:          4   tacaccccgattacaccccga 24
                    ||||||| |||||||||||||
Sbjct:     125138  tacacccagattacaccccga 125158
```

Score = 34.2 bits (17),

Expect = 4.5 Identities = 20/21 (95%) Strand = Plus / Plus

```
Query:          4 tacaccccgattacaccccga 24
                  ||||||| |||||||||||||
Sbjct:     125104 tacacccagattacaccccga 125124
```

>gi|28173089|gb|AC104321.7| Oryza sativa chromosome 3 BAC OSJNBa0052F07 genomic sequence, complete sequence Length = 139823 **Score = 34.2 bits** (17), Expect = 4.5 **Identities = 20/21 (95%)** Strand = Plus / Plus

```
Query:          4 tacaccccgattacaccccga 24
                  ||||||| |||||||||||||
Sbjct:       3891 tacacccagattacaccccga 3911
```

Example

Query: Human atoh enhancer, 179 letters [1.5 min]

Result: 57 blast hits

```
1.      gi|7677270|gb|AF218259.1|AF218259 Homo sapiens ATOH1 enhanc... 355 1e-95
2.      gi|22779500|gb|AC091158.11| Mus musculus Strain C57BL6/J ch... 264 4e-68
3.      gi|7677269|gb|AF218258.1|AF218258 Mus musculus Atoh1 enhanc... 256 9e-66
4.      gi|28875397|gb|AF467292.1| Gallus gallus CATH1 (CATH1) gene... 78 5e-12
5.      gi|27550980|emb|AL807792.6| Zebrafish DNA sequence from clo... 54 7e-05
6.      gi|22002129|gb|AC092389.4| Oryza sativa chromosome 10 BAC O... 44 0.068
7.      gi|22094122|ref|NM 013676.1| Mus musculus suppressor of Ty ... 42 0.27
8.      gi|13938031|gb|BC007132.1| Mus musculus, Similar to suppres... 42 0.27
```

```
gi|7677269|gb|AF218258.1|AF218258 Mus musculus Atoh1 enhancer sequence Length = 1517
     Score = 256 bits (129), Expect = 9e-66 Identities = 167/177 (94%),

     Gaps = 2/177 (1%) Strand = Plus / Plus

Query:    3 tgacaatagagggtctggcagaggctcctggccgcggtgcggagcgtctggagcggagca 62
            ||||||||||||| ||||||||||||||||||| ||||||||||||||||||||||||||
Sbjct: 1144 tgacaatagaggggctggcagaggctcctggccccggtgcggagcgtctggagcggagca 1203

Query:   63 cgcgctgtcagctggtgagcgcactctcctttcaggcagctccccggggagctgtgcggc 122
            ||||||||||||||||||||||||||  ||||||||| |||||||||||||||| |||||
Sbjct: 1204 cgcgctgtcagctggtgagcgcactc-gctttcaggccgctccccggggagctgagcggc 1262

Query:  123 cacatttaacaccatcatcacccctccccggcctcctcaacctcggcctcctcctcg 179
            ||||||||||||| || ||| |||||||||||||||||||| |||||||||||||||
Sbjct: 1263 cacatttaacaccgtcgtca-ccctccccggcctcctcaacatcggcctcctcctcg 1318
```

NCBI BLAST Homepage:

http://www.ncbi.nlm.nih.gov/BLAST/

BLAST Tutorial:

http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/information3.html

BLAST Statistical Measures

http://www.ncbi.nlm.nih.gov/BLAST/tutorial/Altschul-1.html