# CS481/CS583: Bioinformatics Algorithms

Can Alkan
EA509
calkan@cs.bilkent.edu.tr

**http://www.cs.bilkent.edu.tr/~calkan/teaching/cs481/**

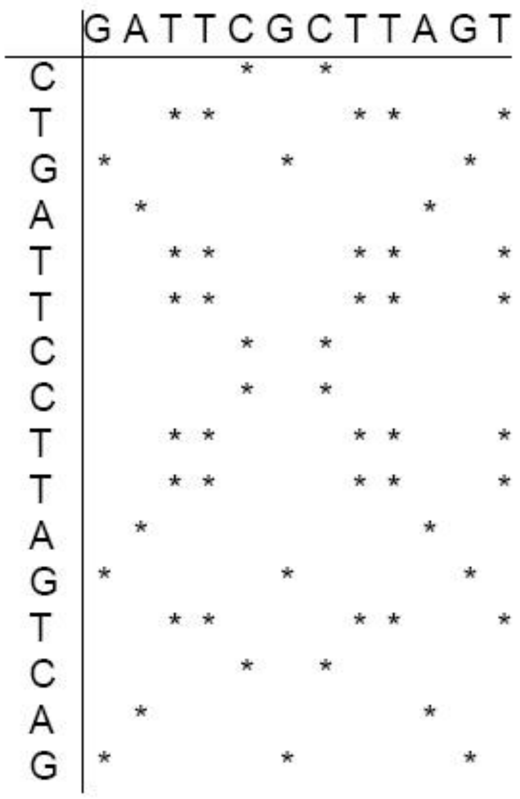
## SIMILARITY SEARCH

## Heuristic Similarity Searches

- Genomes are huge: Smith-Waterman quadratic alignment algorithms are too slow
- Alignment of two sequences usually has short identical or highly similar fragments
- Many heuristic methods (i.e., FASTA) are based on the same 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

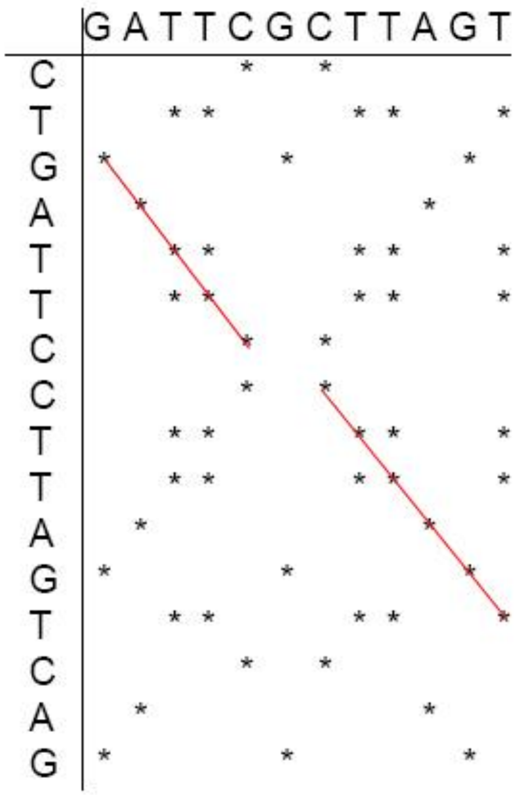
- Dot matrices show similarities between two sequences
- FASTA makes an implicit dot matrix from short exact matches, and tries to find long diagonals (allowing for some mismatches)



| | 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 | * | | | | | * | | | | | * | |

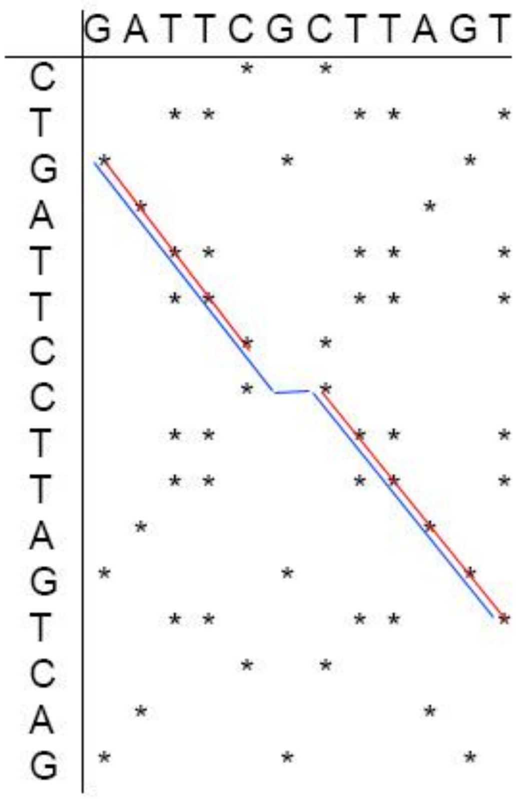
# Dot Matrices (cont'd)

- Identify diagonals above a threshold length
- Diagonals in the dot matrix indicate exact substring matching



# 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



# Approximate Pattern Matching Problem

- Goal: *Find all approximate occurrences of a pattern in a text*
- Input: A pattern $\mathbf{p} = p_1\ldots p_n$, text $\mathbf{t} = t_1\ldots t_m$, and $k$, the maximum number of mismatches
- 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 $\mathbf{p} \leq k$)

## Approximate Pattern Matching: A Brute-Force Algorithm

```
ApproximatePatternMatching(p, t, k)
n □ length of pattern p
m □ length of text t
for i □ 1 to $m - n + 1$
    dist □ 0
    for j □ 1 to n
        if $t_{i+j-1}$ != $p_j$
            dist □ dist + 1
    if dist $\leq$ k
        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),
  *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

## 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 (easy).

# Filtration in Query Matching

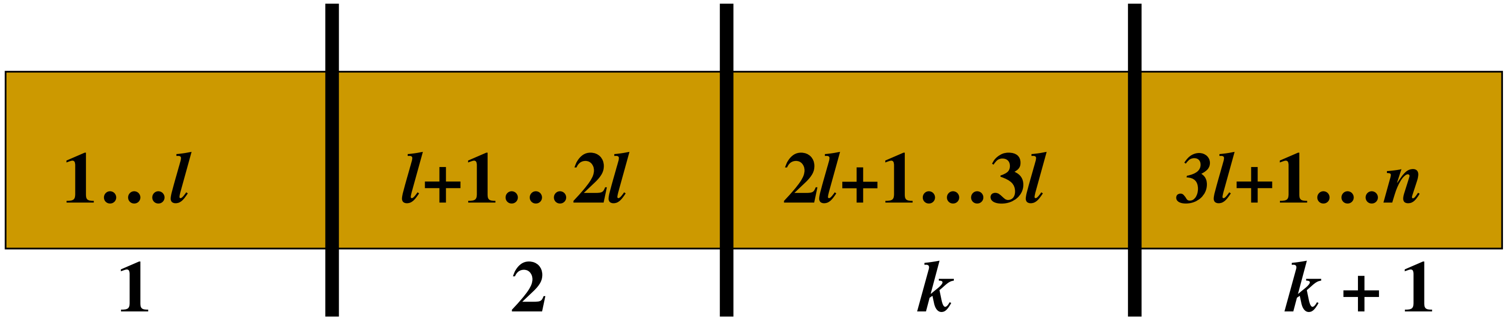
- We want all $n$-matches between a query and a text with up to $k$ mismatches
- "Filter" out positions we know do not match between text and query
- **Potential match detection**: find all matches of $\ell$-tuples in query and text for some small $\ell$
- **Potential match verification**: Verify each 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 $k$ mismatches, they must share an $\ell$-tuple that is perfectly matched, with $\ell = \lfloor n/(k+1) \rfloor$
- Break string of length $n$ into $k$+1 parts, each 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 $l=n/(k+1)=n/4$:

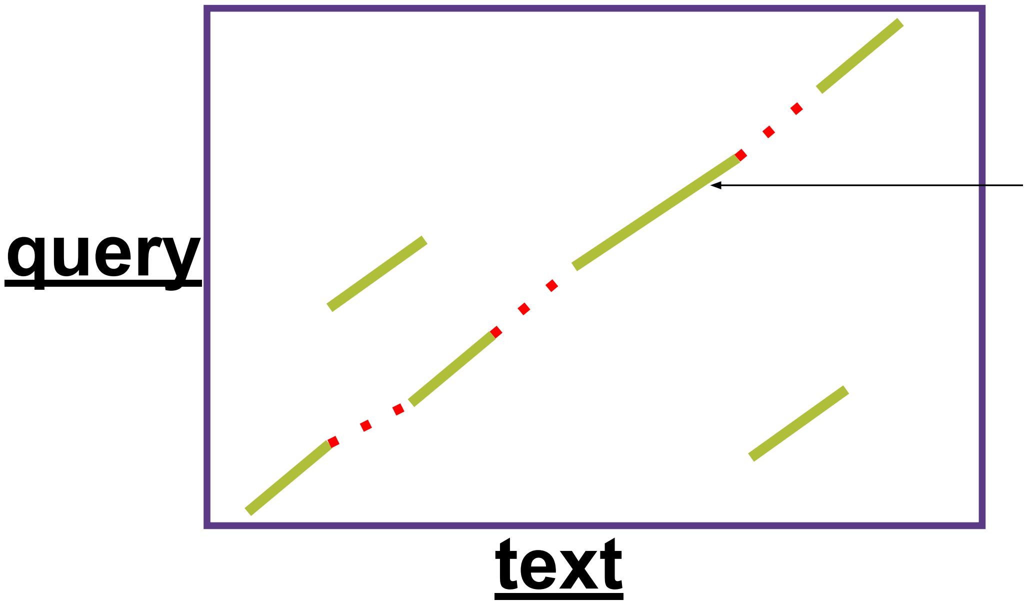


- There are at most $k$ mismatches in $n$, so at the very least there must be one out of the $k$+1 $\ell$–tuples without a mismatch

**What is this based on?**

# Filtration: Match Verification

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



Extend perfect match of length $\ell$ until we find an approximate match of length $n$ with $k$ mismatches

# Filtration: Example

| | ***k*** = 0 | ***k*** = 1 | ***k*** = 2 | ***k*** = 3 | ***k*** = 4 | ***k*** = 5 |
|---|---|---|---|---|---|---|
| $\ell$-tuple length | ***n*** | ***n***/2 | ***n***/3 | ***n***/4 | ***n***/5 | ***n***/6 |

**Shorter perfect matches required**

**Performance decreases**

Lipman & Pearson, 1985

## FASTP

# FASTP

- Three phase algorithm

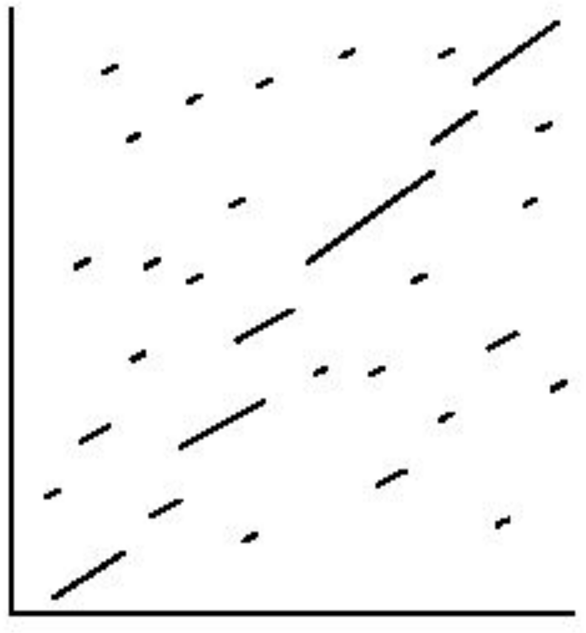
1. Find short good matches using k-mers
    1. k=1, k=2
2. Find start and end positions for good matches
3. Use DP to align good matches

# FASTP: Phase 1 (1)

```
position   1 2 3 4 5 6 7 8 9 10 11
protein 1 n c s p t a . . . . .
protein 2 . . . . . a c s p r k
                          position in            offset
amino acid             protein 1 protein 2    pos 1 – pos2
-----------------------------------------------------
a                         6          6              0
c                         2          7             -5
k                         -         11
n                         1          -
p                         4          9             -5
r                         -         10
s                         3          8             -5
t                         5          -
-----------------------------------------------------
Note the common offset for the 3 amino acids c,s and p
A possible alignment can be quickly found :
protein 1 n c s p t a
            | | |
protein 2 a c s p r k
```

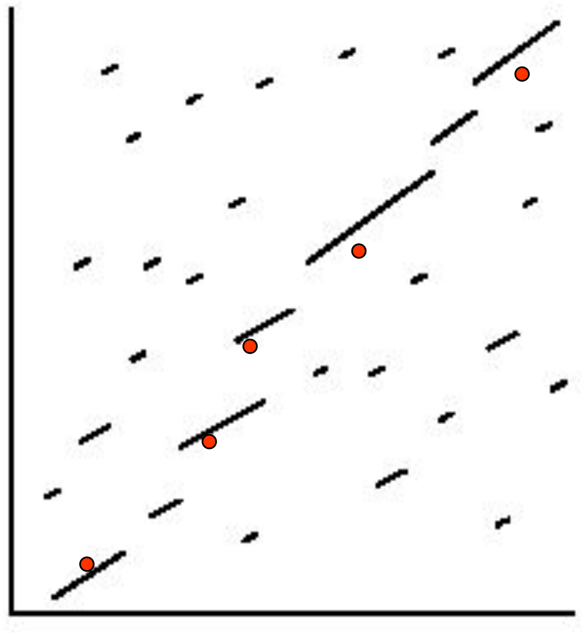
# FASTP: Phase 1 (2)

- Similar to dot plot
- Offsets range from 1-m to n-1
- Each offset is scored as
  - # matches - # mismatches
- Diagonals (offsets) with large score show local similarities



# FASTP: Phase 2

- 5 best diagonal runs are found
- Rescore these 5 regions using PAM250.
  - Initial score
- Indels are not considered yet



## FASTP: Phase 3

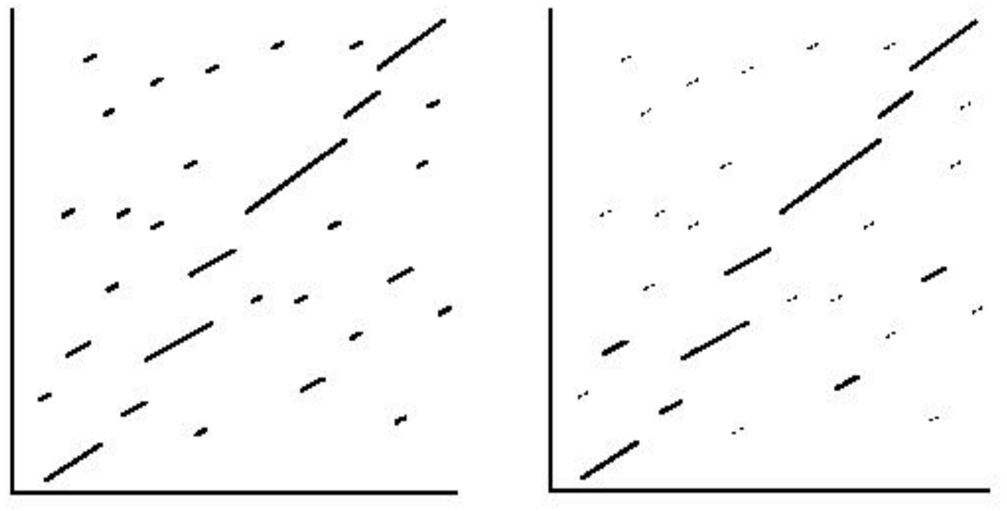
- Sort the aligned regions in descending score
- Optimize these alignments using Needleman-Wunsch
- Report the results

Pearson 1995

## FASTA – IMPROVEMENT OVER FASTP

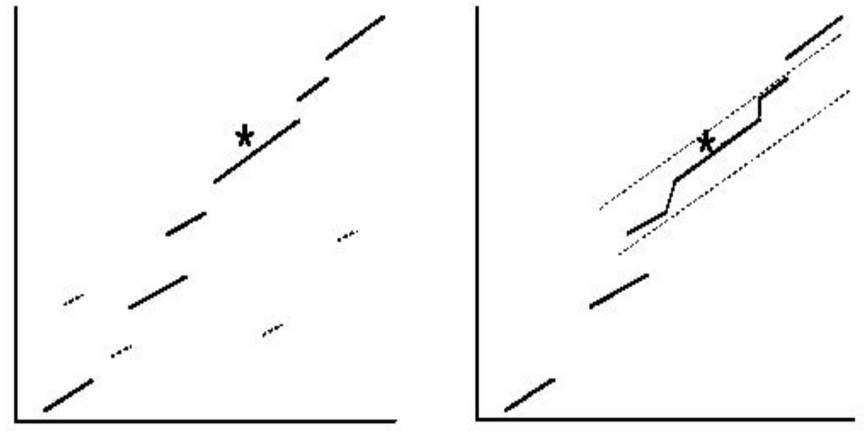
# FASTA (1)

- Phase 2: Choose 10 best diagonal runs instead of 5



# FASTA (2)

- Phase 2.5
  - Eliminate diagonals that score less than some given threshold.
  - Combine matches to find longer matches. It incurs join penalty similar to gap penalty



## FASTA Variations

- TFASTAX and TFASTAY: query protein against a DNA library in all reading frames
- FASTAX, FASTAY: DNA query in all reading frames against protein database

## BLAST

# Local alignment is too slow…

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

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

# 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)
- **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

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

## 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”
  - “Seed-and-extend”

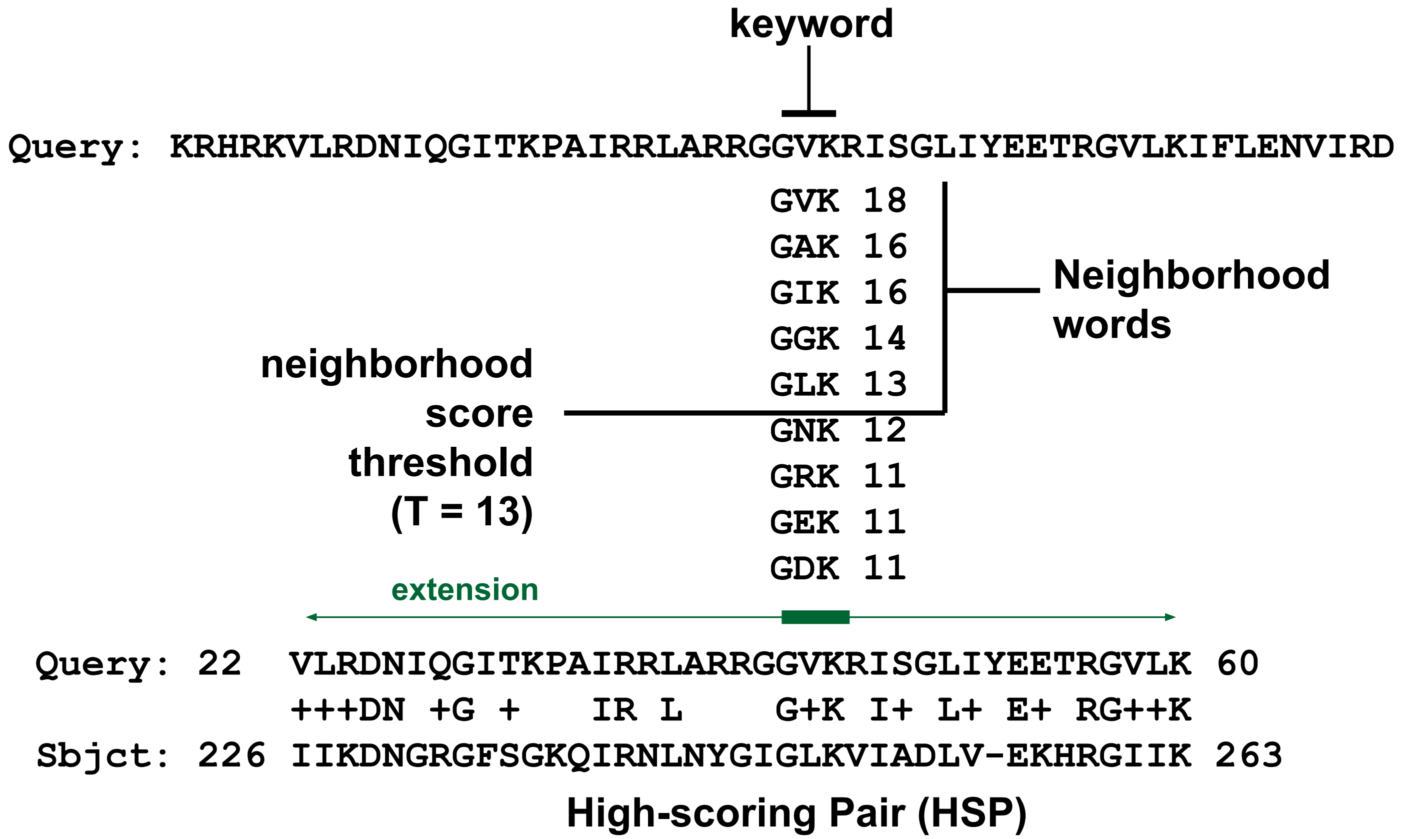
## What Similarity Reveals

- BLASTing a new gene
  - Evolutionary relationship ✓
  - Similarity between protein function ✓
- BLASTing a genome
  - Potential genes

# 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 → original version
  - For each k-mer *w* find all k-mer that aligns with score at least cutoff T
- Local alignment extension for each found keyword
  - Extend result until longest match above threshold is achieved
- Running time O(*nm*)

# BLAST algorithm (cont'd)



```
Query: KRHRKVLRDNIQGITKPAIRRLARRGGVKRISGLIYEETRGVLKIFLENVIRD
```

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

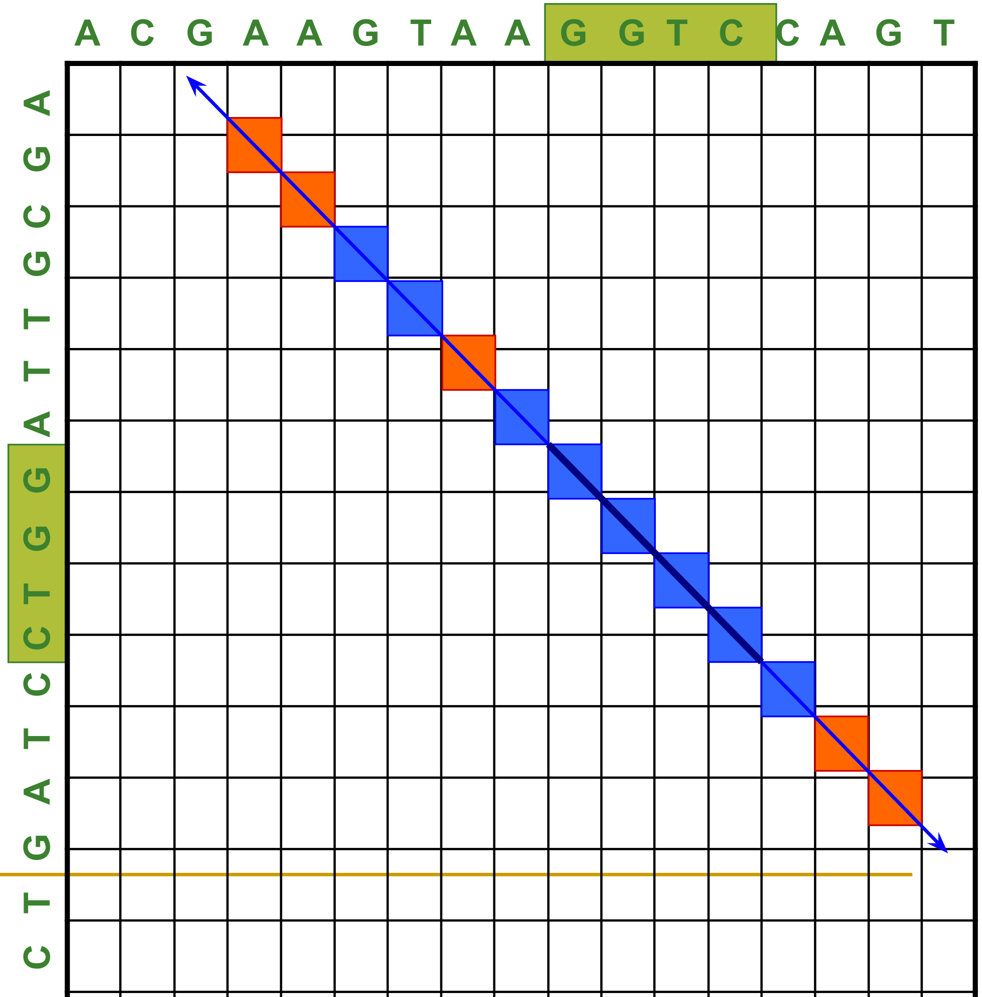
```
Query:  22  VLRDNIQGITKPAIRRLARRGGVKRISGLIYEETRGVLK 60
            +++DN +G +    IR L      G+K I+ L+ E+ RG++K
Sbjct: 226 IIKDNGRGFSGKQIRNLNYGIGLKVIADLV-EKHRGIIK 263
```

## 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 50%
- Output result
  GTAAGGTCC
  GTTAGGTCC



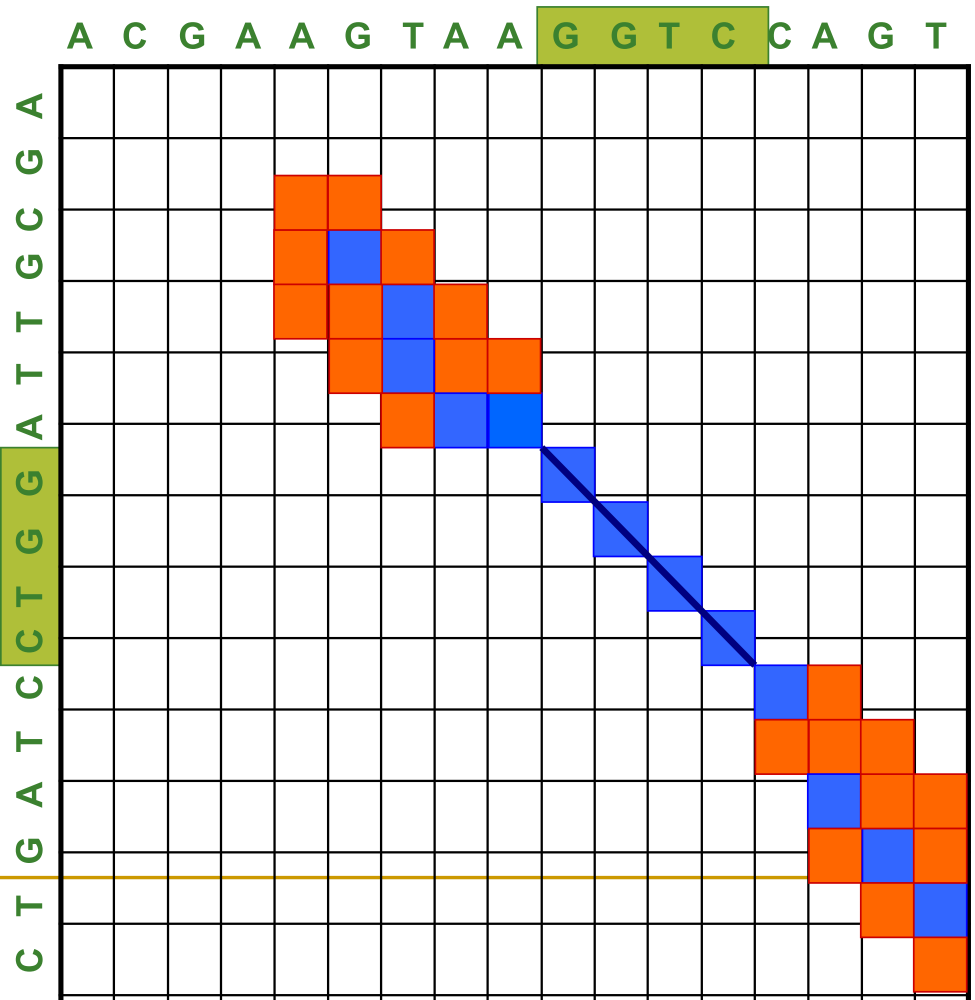
A C G A A G T A A G G T C C A G T
A G C G T T A G G T C C T A G T C

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: $\underbrace{\text{Translated query}}_{\text{DNA (translated)}}$ 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

## 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 $x_1 \ldots x_\ell$ and $y_1 \ldots y_\ell$:
  - Segment pair: pair of $\ell$-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 <u>maximal</u> if it has the best score over all segment pairs
- A segment pair is <u>locally maximal</u> 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 $\theta$ has mean $\underbrace{}$ expected value $\overbrace{E(\theta) = Kmne^{-\lambda\theta}}^{\text{expected value}}$; *K* is a constant, *m* and *n* are the lengths of the two compared sequences
  - Parameter $\lambda$ is positive root of:

    $\Sigma_{x,y \text{ in } A}(p_x p_y e^{\delta(x,y)}) = 1$, where $p_x$ and $p_y$ are frequenceies of amino acids *x* and *y*, and *A* is the twenty letter amino acid alphabet

## P-values

- The probability of finding *b* high-scoring segment pairs (HSPs) with a score ≥*S* is given by:
  - $(e^{-E}E^{b})/b\,!$
- For *b* = 0, that chance is:
  - $e^{-E}$
- Thus the probability of finding at least one HSP with a score ≥*S* is:
  - $P = 1 - e^{-E}$

# 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