

MA3259 Lecture 4

# Rapid Heuristic Methods for Local Alignment

LX Zhang  
Department of Mathematics  
National University of Singapore  
matzlx@nus.edu.sg

# Motivation for Rapid Method

---

- Local alignment can be done in quadratic time rigorously
- In practice, quadratic time is too much for a typical application like finding the function of a new gene.
  - 1) there are a large number of homologous genes between two genomic sequences.
  - 2) Databases to be searched grow rapidly.

# Rapid Local Alignment Tools

---

- FASTA  
(Wilbur and Lipman, 1983; Lipman and Pearson, 1985)
- BLAST  
(Altschul et al., 1990; Altschul et al., 1997)
- BLAT  
(Kent, 2002)
- PatternHunter  
(Li et al., 2004)

All above tools are based on that a good alignment usually includes short identical or very similar fragments.

# Dilemma:

## Sensitivity vs Specificity (and Speed)

---

- **Sensitivity** is the percentage of the true local alignments found relative to the total number of local true alignment between given sequences.
- **Specificity** is the number of true local alignment found relative to the total number of alignments reported.

# 4.1 Work Distribution and Occurrences

---

- One important feature of biomolecular sequences is their composition.
- Random model for a DNA or protein sequence is the Bernoulli iid model

Each residue in a position is drawn randomly according the same probability distribution (like G: 35%, C: 32%, A: 17%, T: 15%) and different residues are drawn independently.

Given a random sequence

$$S = s_1 s_2 \cdots s_i s_{i+1} \cdots s_n$$

in which each letter  $x$  occurs with probability  $p(x)$  at a position.

For a  $k$ -character word  $W$  (called **k-mer**), we are interested in the following questions:

1. What is the probability that  $W$  occurs at a specific position in  $S$ ?
2. What is the average number of occurrences of  $W$  in  $S$ ?
3. What is the probability that  $W$  occurs in  $S$  (one or more times)?

Let  $W = w_1 w_2 \dots w_k$ .

We say **W occurs at position j in S (or W hit S at j)** if and only if

$$w_1 = s_{j-k+1}, \quad w_2 = s_{j-k+2}, \quad \dots, \quad w_k = s_j.$$

AGCTTTCGATTATAACGTATTATATAGGGCTTTAAAAAAA  
          TATA          TATA  
                          TATA

Thus, for  $j \geq k$ , W hits S at position j with the probability:

$$\begin{aligned} \Pr[\text{W hits S at position } j] &= \Pr[w_1 = s_{j-k+1}] \times \Pr[w_2 = s_{j-k+2}] \times \dots \times \Pr[w_k = s_j] \\ &= p(w_1)p(w_2)\dots p(w_k). \end{aligned}$$

For  $j < k$ ,  $\Pr[\text{W hits S at position } j] = 0$ .

**Example:** In a random DNA sequence generated with the following composition: G: 35%, C: 32%, A: 17%, T: 15%. Then, 6-mer GAATTC hits S at a position (in the middle) with the probability  $0.35 \times 0.17 \times 0.17 \times 0.15 \times 0.15 \times 0.32 = 0.000072828$

Define  $X_i$  as

$$X_j = \begin{cases} 1 & \text{if } W \text{ hits } S \text{ at position } j \\ 0 & \text{Otherwise} \end{cases}$$

Then,

$$N = X_k + X_{k+1} + \dots + X_n$$

is equal to the times the k-mer  $W$  hits  $S$

By the linearity property of the mean, the expected number of occurrences of  $W$  is

$$\begin{aligned} E(N) &= E(X_k) + E(X_{k+1}) + \dots + E(X_n) \\ &= (n-k+1) \Pr[W \text{ hits } S \text{ at some position}] \end{aligned}$$

Example: The base frequencies for E. Coli genome are

A: 24.6%, C: 25.4%, G: 25.4%, T: 24.6%.

The E. Coli genome has 4.693 million base pairs.

4-mer	Expected occurrences
TTTT	$17186 = 0.246 \times 0.246 \times 0.246 \times 0.246 \times (4693000 - 4 + 1)$
ATTC	17745
CCGC	19533

However, computing the probability that  $W$  hits  $S$  is not easy.  
Let  $A_j$  denote the event that  $W$  hits  $S$  at position  $j$ .

$$\begin{aligned} \text{Then, } & \Pr[A_1 \text{ or } A_2 \text{ or } \dots \text{ or } A_n \text{ occurs}] \\ & < \Pr[A_1] + \Pr[A_2] + \dots + \Pr[A_n] \\ & = (n-k+1)\Pr[W \text{ hits } S \text{ at position } k] \end{aligned}$$

because  $A_j$  and  $A_i$  are not independent if  $|i-j| \leq k$ .

S: XX  
GGGGATGG  
GGGGATGG

## 4.2 Finding Exact Word Matches

---

An exact word match is a run of identities between two sequences.

Formally, the exact word match problem is

**Input:** two sequences and  $k > 0$ ;

**Objective:** to find all occurrences of each  $k$ -mer that appears in both sequences.

There are several data structures can be used to solve it efficiently:  
We just introduce **hash tables**.

# Hash Table Technique

Hash table technique associates k-mers with integers.

- It stores all k-mers in an array (or table as the name indicates).
- It transforms a k-mer into the index of the cell where the k-mer is stored using a simple function, called the **hash function**.

For DNA sequences, the hash function maps a k-mer

$$W = w_1 w_2 \cdots w_k$$

into the following number

$$h(W) = f(w_k) + f(w_{k-1}) \times 4 + \cdots + f(w_1) \times 4^{k-1}$$

where  $f(w_i) = 0, 1, 2, 3$  for  $w_i = A, C, G, T$ .

## 3-mer

h(W)

W

0	AAA	16	CAA	32	GAA	48	TAA
1	AAC	17	CAC	33	GAC	49	TAC
2	AAG	18	CAG	34	GAG	50	TAG
3	AAT	19	CAT	35	GAT	51	TAT
4	ACA	20	CCA	36	GCA	52	TCA
5	ACC	21	CCC	37	GCC	53	TCC
6	ACG	22	CCG	38	GCG	54	TCG
7	ACT	23	CCT	39	GCT	55	TCT
8	AGA	24	CGA	40	GGA	56	TGA
9	AGC	25	CGC	41	GGC	57	TGC
10	AGG	26	CGG	42	GGG	58	TGG
11	AGT	27	CGT	43	GGT	59	TGT
12	ATA	28	CTA	44	GTA	60	TTA
13	ATC	29	CTC	45	GTC	61	TTC
14	ATG	30	CTG	46	GTG	62	TTG
15	ATT	31	CTT	47	GTT	63	TTT

# 1. Build a hash table and hash query sequence, say GATCCATCTT, into table

1 2 3 4 5 6 7 8 9 10  
 G A T C C A T C T T  
 G A T  
 A T C  
 T C C  
 C C A  
 C A T  
 A T C  
 T C T  
 C T T

000000 (0)	AAA	
...	...	
001101 (13)	ATC	→ 2 → 6
...	...	
010011 (19)	CAT	→ 5
010100 (20)	CCA	→ 4
...	...	
011111 (31)	CTT	→ 8
...	...	
100011 (35)	GAT	→ 1
...	...	
110101 (53)	TCC	→ 3
110110 (54)	TCG	
110111 (55)	TCT	→ 7
...	...	
111111 (63)	TTT	

## 2. Scan Target Sequence for Exact Word Matches

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

h → 19

000000 (0)	AAA		
...	...		
001101 (13)	ATC	→ 2	→ 6
...	...		
010011 (19)	CAT	→ 5	
010100 (20)	CCA	→ 4	
...	...		
011111 (31)	CTT	→ 8	
...	...		
100011 (35)	GAT	→ 1	
...	...		
110101 (53)	TCC	→ 3	
110110 (54)	TCG		
110111 (55)	TCT	→ 7	
...	...		
111111 (63)	TTT		

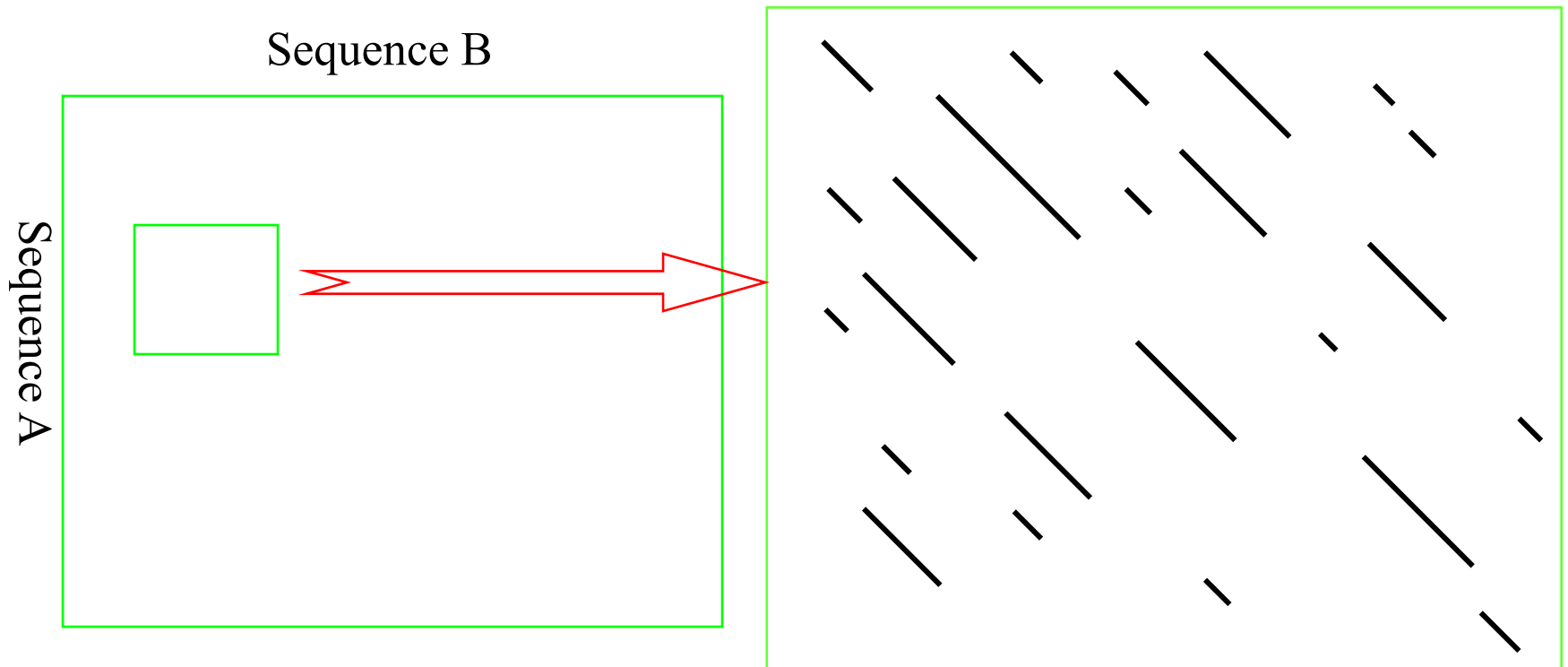
Scan target sequence by sliding window of size  $k$ . For each resulting  $k$ -mer  $W$ , compute  $h(W)$  and use it as index to find the matches in the query sequence.

## 4.3 FASTA

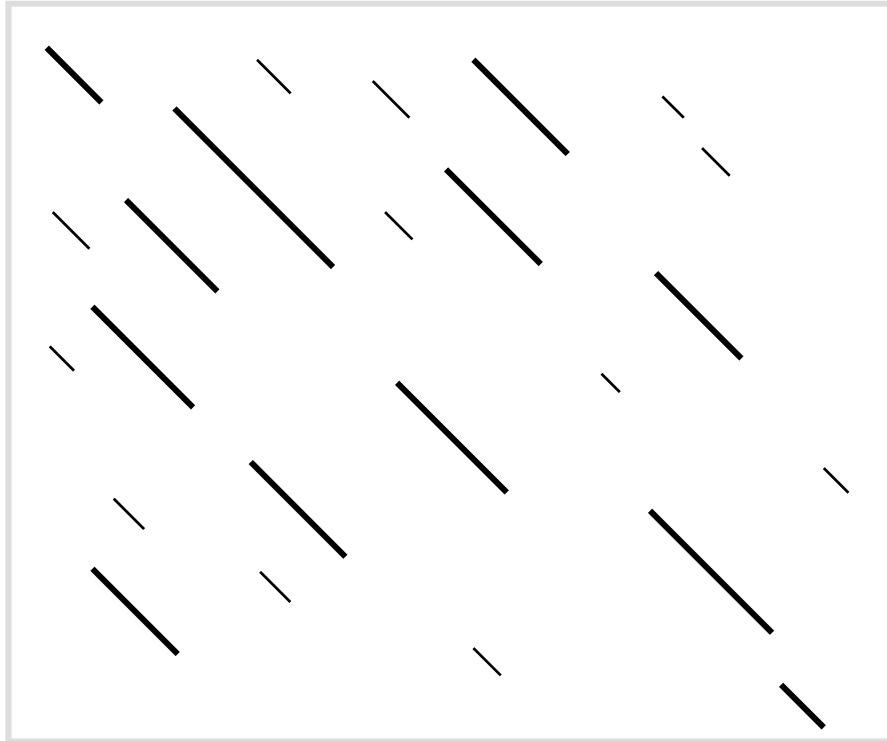
---

- 1) Find runs of  $t$  or more identities, and identify regions with the highest density of identities, where  $t$  is a parameter.
- 2) Re-score using PAM matrix, and keep top scoring exact matches.
- 3) Eliminate matches that are unlikely to be part of an optimal alignment to form a band where there are good alignments.
- 4) Optimize the alignment in the band.

Step 1: Find runes of  $t$  or more identities, and identify regions with the highest density of identities.



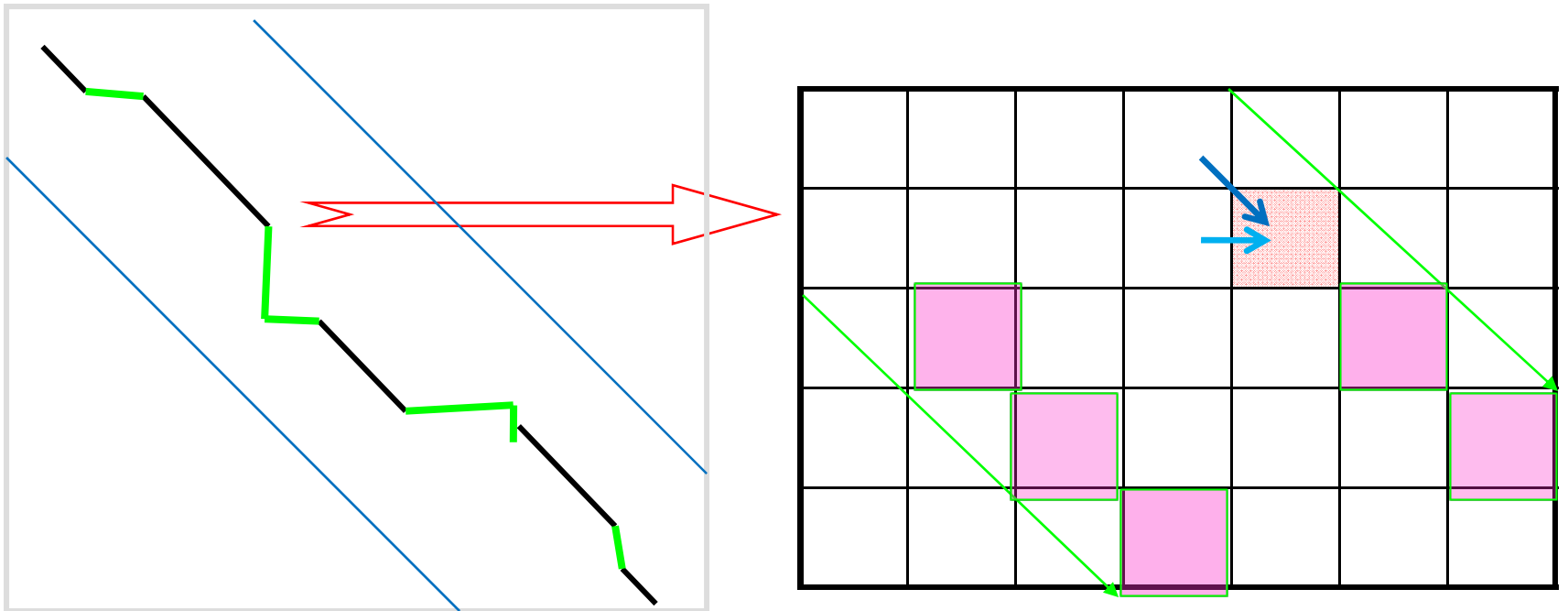
Step 2: Re-score using PAM matrix, and keep top scoring matches.



Step 3: Eliminate segments that are unlikely to be part of the alignment. Join top scoring segments together to form a chain and form a band of residue 32 centered around the chain.



Step 4: Optimize the alignment in a band of 32 residues centered around the chain.



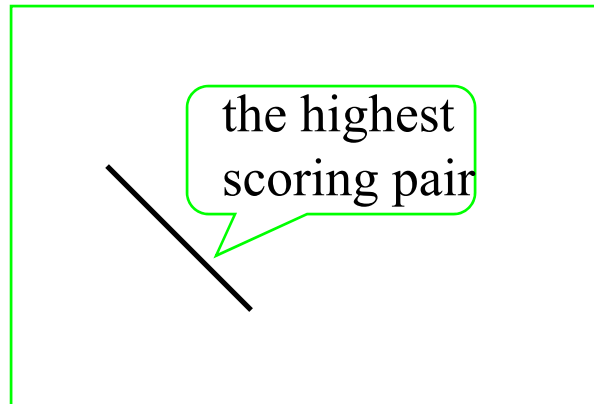
# 4.4 BLAST

---

- ✓ **B**asic **L**ocal **A**lignment **S**earch **T**ool  
(by Altschul, Gish, Miller, Myers and Lipman)
- ✓ The central idea of the BLAST algorithm is that a statistically significant alignment likely contain a *high-scoring pair* of aligned words.

# The maximal segment pair measure

- ✓ A maximal segment pair (MSP) is defined to be the highest scoring pair of identical length segments chosen from 2 sequences. (for DNA: Identities: +5; Mismatches: -4)



- The MSP score may be computed in time proportional to the product of their lengths. (How?) An exact procedure is too time consuming.
- BLAST heuristically attempts to calculate the MSP score.

MSP = Optimal ungapped local alignment

# A matrix of similarity scores

	C	T	A	T	C	A	T	T	C	T	G
G	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	5
A	-4	-4	5	-4	-4	5	-4	-4	-4	-4	-4
T	-4	5	-4	5	-4	-4	5	5	-4	5	-4
C	5	-4	-4	-4	5	-4	-4	-4	5	-4	-4
C	5	-4	-4	-4	5	-4	-4	-4	5	-4	-4
A	-4	-4	5	-4	-4	5	-4	-4	-4	-4	-4
T	-4	5	-4	5	-4	-4	5	5	-4	5	-4
C	5	-4	-4	-4	5	-4	-4	-4	5	-4	-4
T	-4	5	-4	5	-4	-4	5	5	-4	5	-4
T	-4	5	-4	5	-4	-4	5	5	-4	5	-4

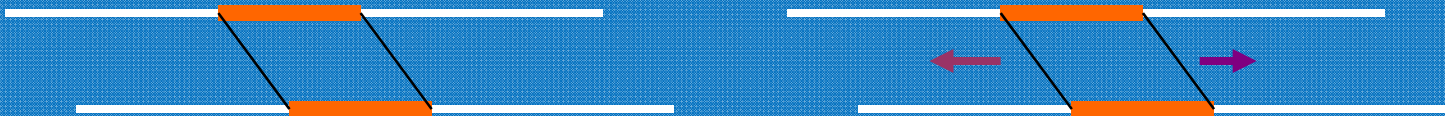
# A maximum-scoring segment

	C	T	A	T	C	A	T	T	C	T	G
G	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	5
A	-4	-4	5	-4	-4	5	-4	-4	-4	-4	-4
T	-4	5	-4	5	-4	-4	5	5	-4	5	-4
C	5	-4	-4	-4	5	-4	-4	-4	5	-4	-4
C	5	-4	-4	-4	5	-4	-4	-4	5	-4	-4
A	-4	-4	5	-4	-4	5	-4	-4	-4	-4	-4
T	-4	5	-4	5	-4	-4	5	5	-4	5	-4
C	5	-4	-4	-4	5	-4	-4	-4	5	-4	-4
T	-4	5	-4	5	-4	-4	5	5	-4	5	-4
T	-4	5	-4	5	-4	-4	5	5	-4	5	-4

# BLAST Program

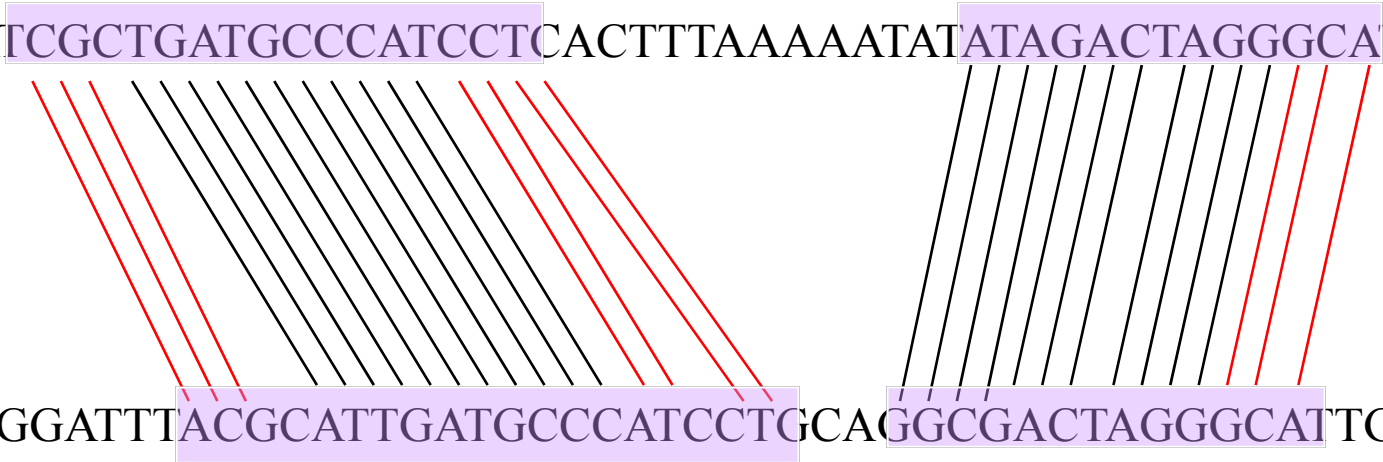
---

1. Filtering step: Look for short exact matches of size  $k$  ( $=11$ ) positions between query and target sequences.
  2. Alignment step: Extend each match found in Stage 1 in either side into for local alignment, and report it if significance.
- Its running time is linear time  $c(m+n)$ , where constant factor  $c$  depends on  $k$ .



ACTCATCGCTGATGCCCATCCTCACTTTAAAAATATATAGACTAGGGCATTGGGA

GCAAAGGATTACGCATTGATGCCCATCCTGCAGGCGACTAGGGCATTGG



# Filtering Step

## Step 1: Build the hash table for Sequence A. (3-tuple example)

For DNA sequences:

Seq. A = AGATCGAT  
          12345678

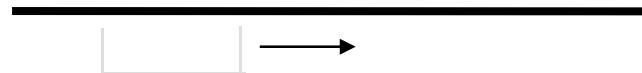
AAA	
AAC	
..	
AGA	→ 1
..	
ATC	→ 3
..	
CGA	→ 5
..	
GAT	→ 2 → 6
..	
TCG	→ 4
..	
TTT	

For protein sequences:

Seq. A = ELVIS

Add  $xyz$  to the hash table  
if  $Score(xyz, ELV) \geq T$ ;  
Add  $xyz$  to the hash table  
if  $Score(xyz, LVI) \geq T$ ;  
Add  $xyz$  to the hash table  
if  $Score(xyz, VIS) \geq T$ ;

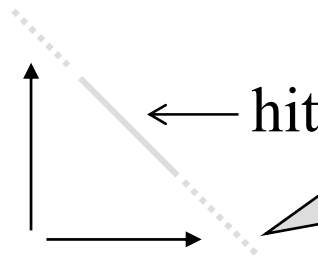
Step2: Scan sequence B for hits.



# Extension Step

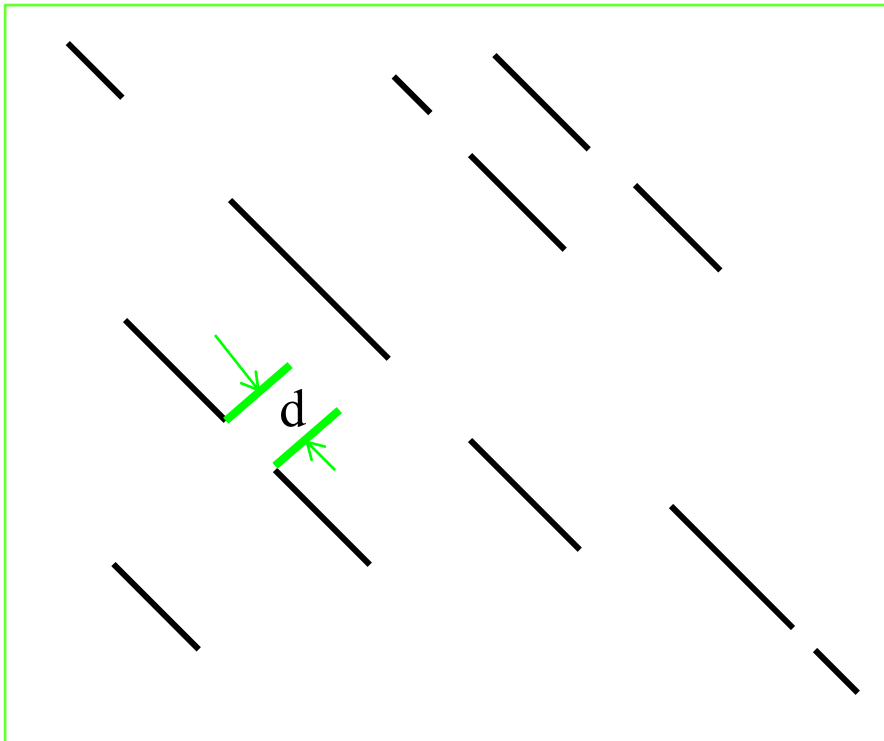
## Step 3: Extend hits.

Terminate if the score of the extension fades away. (That is, when we reach a segment pair whose score falls a certain distance below the best score found for shorter extensions.)



BLAST 2.0 saves the time spent in extension, and considers gapped alignments.

# Gapped BLAST



The two-hit  
method



## 4.5 PatternHunter: Spacing out matching positions

---

- Filtering step: Look for short matches in  $k(=11)$  non-contiguous positions, specified by a spaced seed (e.g. 1\*\*11\*1), between the given sequences.
- Alignment step: Extend matches found in the above step into an (ungapped or gapped) alignment.

```
GCAATTGCCGGATCTT
      | | | | |
GCGATTGCTGGCTCTA
```

```
GCAATTGCCGGATCTT
              | | | |
GCGATTGCTGGCTCTA
```

# The idea makes a big difference

---

- A good spaced seed not only increases specificity, but also reduces running time.
  - a) It is more likely that a good alignment has at least one hit to a good spaced seed. For instance, the PH spaced seed hits more than 45% of alignments of length 64 with 70% conservation. By contrast, the default BLASTN seed hits only 30% of alignments.
  - b) Few alignments have too many hits. Time reduction in Step 2 comes from that the average number of matches found in Step 1 decreases.

PH default seed: 111\*11\*\*1\*1\*\*11\*111

[Set subsequence](#) From:  To:

[Choose database](#) Mus musculus - WGS

[Return alignment endpoints only](#)

Now: **BLAST!** or **Reset query** **Reset all**

### Options for advanced blasting

[Hits computed](#) 250

[Choose filter](#)  Low complexity  Rodent Repeats  Mask for lookup table only  Mask lower case

[Expect](#) 10

[Word size](#) 11

[Percent Identity](#) None

[Discontiguous Word options](#) Template length 21 Template type Coding Require 2 word hits for extension

[Other advanced](#)

# Questions

---

- Why is a seed like the PH seed better than the BLAST seed  
(1111111111 for weight 11) of the same weight?

A very challenging mathematical problem.

- Are all spaced seeds better than BLAST seed of the same weight ?

1\*1\*1\*1\*1\*1\*1 is worse than 1111111

- Which spaced seeds are optimal?

A difficult problem. No polynomial-time algorithm is known for finding them

## 4.5 Running BLAST for Homology Search

- The BLAST program searches an entire database to find all sequences that are homologies of the query sequence.
- Sample BLAST output

```
>sp|P46420.2|GSTF4_MAIZE Glutathione S-transferase 4 (GST-IV) (GST-27) (GST
class-phi member 4)
Length=223
```

```
Score = 36.6 bits (83), Expect = 0.11, Method: Compositional matrix adjust.
Identities = 29/86 (33%), Positives = 44/86 (51%), Gaps = 9/86 (10%)
```

```
Query 1  MSLPIIKVH-WLDHSRAFRLWLWLLDHLNLEYEIVPYKR-DANFRAPPELKKIHPLGRSPL 58
          M+ P +KV+ W          R L L+  ++YE+VP R D + R P L + +P G+ P+
Sbjct 1  MATPAVKVYGWAI SPFVSRALLALEEAGVDYELVPMSRQDGDHRRPEHLAR-NPPGKVPV 59

Query 59  LEVQDRETGKKKILAESGFIFQYVLQ 84
          LE D          L ES I ++VL+
Sbjct 60  LEDGDL-----TLFESRAIARHVLR 79
```

# BLAST Terminology

- **Composition:**

The frequencies of letters in a given sequence.

- **Complexity**

A measure of the information content of a sequence.

AAAGGAAAGGG

AGVLTMSIKLVIJ

Low complexity



High complexity

- **Gapped and Ungapped Alignment**

```
LDHSRAFRLLWLLDHLNLEYEIVPYKR
  R L L+ ++YE+VP R
AISPFVSRALLALEEAGVDYELVPMSR
```

```
LEVQDRETGKKKILAESGFIFQYVLQ
LE D L ES I ++VL+
LEDGDL-----TLFESRAIARHVLR
```

# BLAST Program Parameters

- **Low complexity filter**
  - Replace low complexity sequence stretches with X's so that they will not be aligned in extension
- **Expect**
  - E-value cutoff for reporting HSPs
- **Word size**
  - Allow users to define word size for the look-up table.
  - Default is 11 for DNA sequences and 3 for proteins.
- **Matrix**
  - Choose scoring matrix to be used for scoring.
- **Gap costs**
  - Choose gap opening and extension penalties.

# BLAST Printout

- **Bit score**
  - Normalized alignment score reported by BLAST for each HSP.
- **Expect**
  - The estimate of the number of HSPs with score **S or more** that are expected by chance.
  - The Expect value are calculated using the following formula:

$$\text{Expect}(s) = \sum_T K(l_Q - \bar{l}(s))(l_T - \bar{l}(s))e^{-\lambda s}$$

$K$  and  $\lambda$  are correcting constants that are computed from the sequence composition and scoring matrices.

$\bar{l}(s)$  is the length adjustment, the mean length of alignment with score  $s$ .

Score = 36.6 bits (83), Expect = 0.11, Method: Compositional matrix adjust.  
 Identities = 29/86 (33%), Positives = 44/86 (51%), Gaps = 9/86 (10%)

```
Query 1  MSLPIIKVH-WLDHSRAFRLLWLLDHLNLEYEIVPYKR-DANFRAPPELKKIHPLGRSPL 58
          M+ P +KV+ W          R L L+  ++YE+VP R D + R P L + +P G+ P+
Sbjct 1  MATPAVKVYGWAI SPFVSRALLALEEAGVDYELVPMRQDGDHRRPEHLAR-NPFGKVPV 59

Query 59 LEVQDRETGKKKILAESGFIFQYVLQ 84
          LE D          L ES I ++VL+
Sbjct 60 LEDGDL-----TLFESRAIARHVLR 79
```

...

Database: Non-redundant SwissProt sequences  
 Posted date: May 23, 2008 5:56 PM  
 Number of letters in database: 124,438,792  
 Number of sequences in database: 332,988

```
Lambda      K      H
          0.320  0.137  0.401
Gapped
Lambda      K      H
          0.267  0.0410 0.140
```

Matrix: BLOSUM62  
 Gap Penalties: Existence  
 Number of Sequences: 332

...  
 Length of query: 234  
 Length of database: 1244  
 Length adjustment: 111  
 ...

$$\text{Expect}(s) = \sum_T K(l_Q - \bar{l}(s))(l_T - \bar{l}(s))e^{-\lambda s}$$

$$\lambda = 0.267, K = 0.041.$$

Because the raw is 83, the bit score is

$$\left[0.267 \times 83 - \ln(0.041)\right] / \ln(2) \approx 36.58$$

in agreement with the printout value. The value of Expect is

$$0.041 \times (234 - 111) \times (124438793 - 332988 \times 111) \times e^{-0.267 \times 83} \approx 0.105,$$