

# Sequence Alignment

Lecture 2, Thursday April 3, 2003

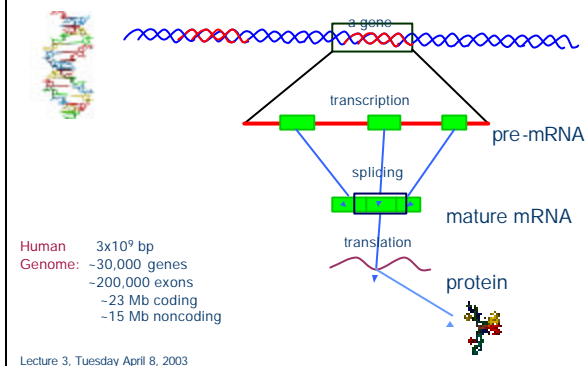


# Review of Last Lecture

Lecture 2, Thursday April 3, 2003



## Structure of a genome



Lecture 3, Tuesday April 8, 2003

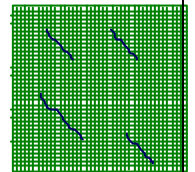
## The Smith-Waterman algorithm

**Idea:** Ignore badly aligning regions

Modifications to Needleman-Wunsch:

**Initialization:**  $F(0, j) = F(i, 0) = 0$

**Iteration:**  $F(i, j) = \max \begin{cases} F(i-1, j) - d \\ F(i, j-1) - d \\ F(i-1, j-1) + s(x_i, y_j) \end{cases}$

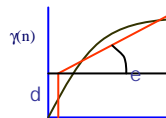


Lecture 3, Tuesday April 8, 2003

## Affine Gaps

$$\gamma(n) = d + (n-1) \times e$$

gap open                  gap extend



To compute optimal alignment,

At position  $i, j$ , need to "remember" best score if gap is open  
 best score if gap is not open

$F(i, j)$ : score of alignment  $x_1 \dots x_i$  to  $y_1 \dots y_j$   
 if  $x_i$  aligns to  $y_j$

$G(i, j)$ : score if  $x_i$  or  $y_j$  aligns to a gap

Lecture 3, Tuesday April 8, 2003

## Needleman-Wunsch with affine gaps

**Initialization:**  $F(i, 0) = d + (i-1) \times e$   
 $F(0, j) = d + (j-1) \times e$

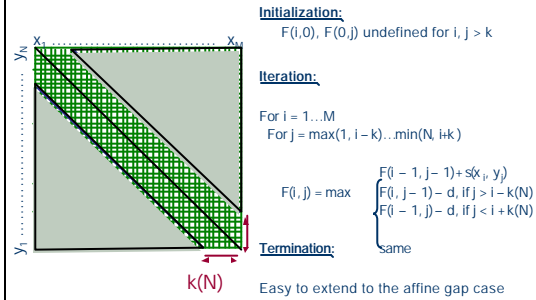
**Iteration:**  $F(i, j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_j) \\ G(i-1, j) + s(x_i, y_j) \end{cases}$

$G(i, j) = \max \begin{cases} F(i-1, j) - d \\ F(i, j-1) - d \\ G(i-1, j) - e \\ G(i, j-1) - e \end{cases}$

**Termination:** same

Lecture 3, Tuesday April 8, 2003

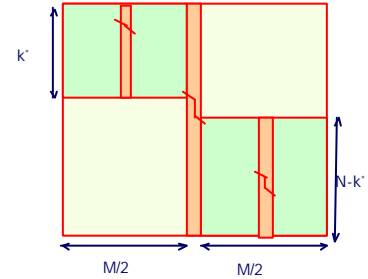
## Bounded Dynamic Programming



Lecture 3, Tuesday April 8, 2003

## Linear-space alignment

- Iterate this procedure to the left and right!



Lecture 3, Tuesday April 8, 2003

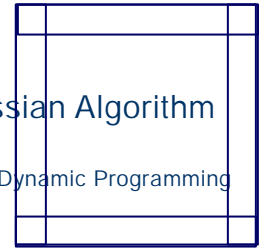
## Today

- The Four-Russian Speedup
- Heuristic Local Aligners
  - BLAST
  - BLAT
  - PatternHunter
- Time Warping

Lecture 3, Tuesday April 8, 2003

## The Four-Russian Algorithm

A useful speedup of Dynamic Programming



## Main Observation

Within a rectangle of the DP matrix, values of  $D$  depend only on the values of  $A, B, C$ , and substrings  $x_{l..r}, y_{l'..r'}$

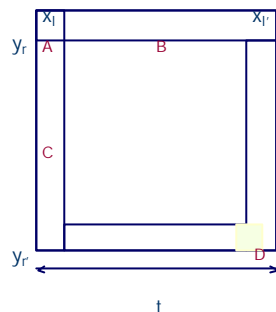
### Definition:

A t-block is a  $t \times t$  square of the DP matrix

### Idea:

Divide matrix in t-blocks,  
Precompute t-blocks

Speedup:  $O(t)$



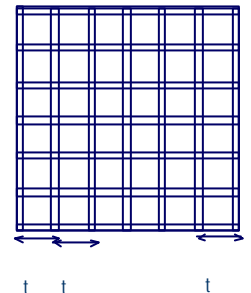
Lecture 3, Tuesday April 8, 2003

## The Four-Russian Algorithm

### Main structure of the algorithm:

- Divide  $N \times N$  DP matrix into  $K \times K \log_2 N$ -blocks that overlap by 1 column & 1 row
- For  $i = 1 \dots K$
- For  $j = 1 \dots K$
- Compute  $D_{ij}$  as a function of  $A_{ij}, B_{ij}, C_{ij}, x[l_1 \dots l_j], y[r_1 \dots r_j]$

**Time:**  $O(N^2 / \log^2 N)$   
 times the cost of step 4



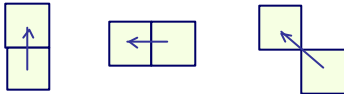
Lecture 3, Tuesday April 8, 2003

## The Four-Russian Algorithm

Another observation:  
( Assume  $m = 0$ ,  $s = 1$ ,  $d = 1$  )

**Lemma:** Two adjacent cells of  $F(\dots)$  differ by at most 1

Gusfield's book covers case where  $m = 0$ ,  
called the edit distance (p. 216):  
minimum # of substitutions + gaps to transform one string to another



Lecture 3, Tuesday April 8, 2003

## The Four-Russian Algorithm

**Proof of Lemma:**



1. Same row:  
a.  $F(i, j) - F(i - 1, j) \leq +1$

At worst, one more gap:

$$\begin{matrix} x_1 \dots x_{i-1} x_i \\ y_1 \dots y_j - \end{matrix}$$

- a.  $F(i, j) - F(i - 1, j) \geq -1$

$$\begin{array}{ccc} F(i, j) & F(i - 1, j - 1) & F(i, j) - F(i - 1, j - 1) \\ \begin{matrix} x_1 \dots x_{i-1} x_i \\ y_1 \dots y_{a-1} y_a y_{a+1} \dots y_j \end{matrix} & \begin{matrix} x_1 \dots x_{i-1} - \\ y_1 \dots y_{a-1} y_a y_{a+1} \dots y_j \end{matrix} & \geq -1 \\ \begin{matrix} x_1 \dots x_{i-1} x_i \\ y_1 \dots y_{a-1} - y_a \dots y_j \end{matrix} & \begin{matrix} x_1 \dots x_{i-1} x_i \\ y_1 \dots y_{a-1} y_a \dots y_j \end{matrix} & +1 \end{array}$$

2. Same column: similar argument

Lecture 3, Tuesday April 8, 2003

## The Four-Russian Algorithm

**Proof of Lemma:**

3. Same diagonal:

- a.  $F(i, j) - F(i - 1, j - 1) \leq +1$

At worst, one additional mismatch in  $F(i, j)$

- b.  $F(i, j) - F(i - 1, j - 1) \geq -1$

$$\begin{array}{ccc} F(i, j) & F(i - 1, j - 1) & F(i, j) - F(i - 1, j - 1) \\ \begin{matrix} x_1 \dots x_{i-1} x_i \\ y_1 \dots y_{i-1} y_i \end{matrix} & \begin{matrix} x_1 \dots x_{i-1} \\ y_1 \dots y_{i-1} \end{matrix} & \geq -1 \\ \begin{matrix} x_1 \dots x_{i-1} x_i \\ y_1 \dots y_{a-1} - y_a \dots y_j \end{matrix} & \begin{matrix} x_1 \dots x_{i-1} \\ y_1 \dots y_{a-1} y_a \dots y_j \end{matrix} & +1 \end{array}$$

Lecture 3, Tuesday April 8, 2003

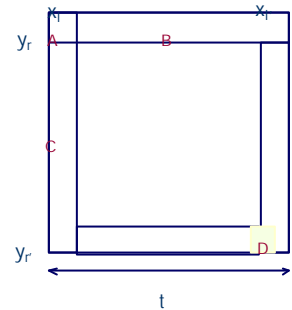
## The Four-Russian Algorithm

**Definition:**

The offset vector is a  $t$ -long vector of values  
from  $\{2, -1, 0, 1, 2\}$ ,  
where the first entry is 0

If we know the value at A,  
and the top row, left column  
offset vectors,  
and  $x_1, \dots, x_r, y_1, \dots, y_r$ ,

Then we can find D



Lecture 3, Tuesday April 8, 2003

## The Four-Russian Algorithm

**Example:**

$x = \text{AACT}$   
 $y = \text{CACT}$

	A	A	C	T	
C	0	1	-1	-1	0
A	0	5	6	5	1
C	-1	4	5	6	1
T	1	5	5	6	-1
	0	0	1	-1	

Lecture 3, Tuesday April 8, 2003

## The Four-Russian Algorithm

**Example:**

$x = \text{AACT}$   
 $y = \text{CACT}$

	A	A	C	T	
C	0	1	-1	-1	0
A	0	1	1	2	1
C	-1	0	1	1	1
T	1	1	2	1	-1
	0	0	1	-1	

Lecture 3, Tuesday April 8, 2003

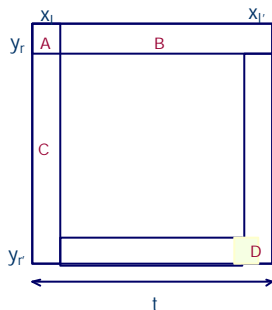
## The Four-Russian Algorithm

### Definition:

The offset function of a  $t$ -block is a function that for any

given offset vectors  
of top row, left column,  
and  $x_l, \dots, x_r, y_l, \dots, y_r$ ,

produces offset vectors  
of bottom row, right column



Lecture 3, Tuesday April 8, 2003

## The Four-Russian Algorithm

We can pre-compute the offset function:

$5^{2(l-1)}$  possible input offset vectors

$4^{2l}$  possible strings  $x_l, \dots, x_r, y_l, \dots, y_r$

Therefore  $5^{2(l-1)} \times 4^{2l}$  values to pre-compute

We can keep all these values in a table, and look up in linear time,  
or in  $O(1)$  time if we assume  
constant-lookup RAM for log-sized inputs

Lecture 3, Tuesday April 8, 2003

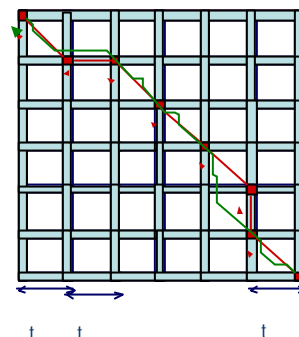
## The Four-Russian Algorithm

### Four-Russians Algorithm: (Arslanov, Dinic, Kronrod, Faradzev)

1. Cover the DP table with  $t$ -blocks
2. Initialize values  $F(\dots)$  in first row & col
3. Row-by-row, use offset values at leftmost col and top row of each block, to find offset values at rightmost col and bottom row
4. Let  $Q$  = total of offsets at row  $N$   
 $F(N, N) = Q + F(N, 0)$

Lecture 3, Tuesday April 8, 2003

## The Four-Russian Algorithm



Lecture 3, Tuesday April 8, 2003

## Heuristic Local Aligners

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

## State of biological databases

### Sequenced Genomes:

Human	$3 \times 10^9$	Yeast	$1.2 \times 10^7$
Mouse	$2.7 \times 10^9$		$\times 12$ different strains
Rat	$2.6 \times 10^9$	Neurospora	$4 \times 10^7$
			14 more fungi within next year
Fugu fish	$3.3 \times 10^8$		
Tetraodon	$3 \times 10^8$		~250 bacteria/viruses
Mosquito	$2.8 \times 10^8$		
Drosophila	$1.2 \times 10^8$	Next 2 years:	
Worm	$1.0 \times 10^8$		Dog, Chimpanzee, Chicken
2 sea squirts $\times$	$1.6 \times 10^8$		
Rice	$1.0 \times 10^9$		
Arabidopsis	$1.2 \times 10^8$		

### Current rate of sequencing:

4 big labs  $\times 3 \times 10^9$  bp / year/lab  
10s small labs

Lecture 3, Tuesday April 8, 2003

## State of biological databases

Number of genes in these genomes:

Vertebrate: ~30,000  
Insects: ~14,000  
Worm: ~17,000  
Fungi: ~6,000-10,000

Small organisms: 100s- 1,000s

Each known or predicted gene has an associated protein sequence

>1,000,000 known / predicted protein sequences

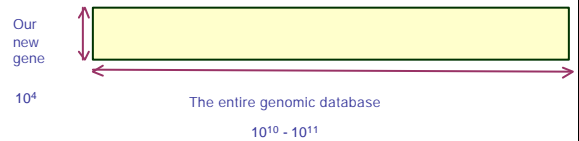
Lecture 3, Tuesday April 8, 2003

## Some useful applications of alignments

Given a newly discovered gene,

- Does it occur in other species?
- How fast does it evolve?

Assume we try Smith-Waterman:



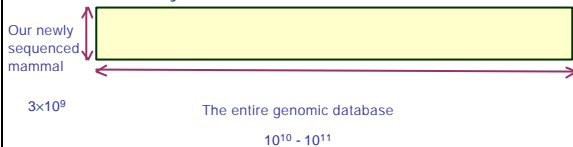
Lecture 3, Tuesday April 8, 2003

## Some useful applications of alignments

Given a newly sequenced organism,

- Which subregions align with other organisms?
- Potential genes
- Other biological characteristics

Assume we try Smith-Waterman:



Lecture 3, Tuesday April 8, 2003

## BLAST

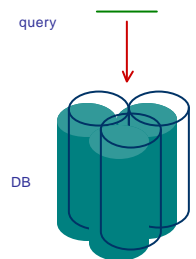
(Basic Local Alignment Search Tool)

### 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



Lecture 3, Tuesday April 8, 2003

## BLAST — Original Version

### Dictionary:

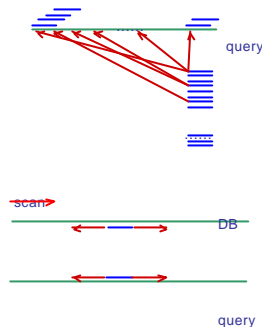
All words of length  $k$  (~11)  
Alignment initiated between words of alignment score  $\geq T$   
(typically  $T = k$ )

### Alignment:

Ungapped extensions until score below statistical threshold

### Output:

All local alignments with score > statistical threshold



Lecture 3, Tuesday April 8, 2003

## BLAST — Original Version

### Example:

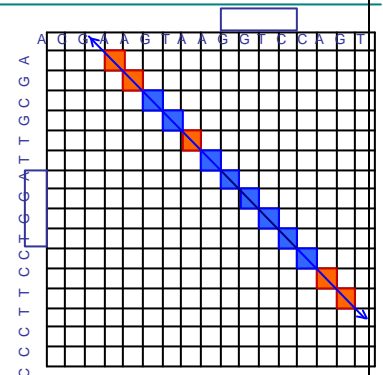
$k = 4$ ,  
 $T = 4$

The matching word GGTC initiates an alignment

Extension to the left and right with no gaps until alignment falls < 50%

### Output:

GTAAGGTCC  
GTTAGGTCC



Lecture 3, Tuesday April 8, 2003

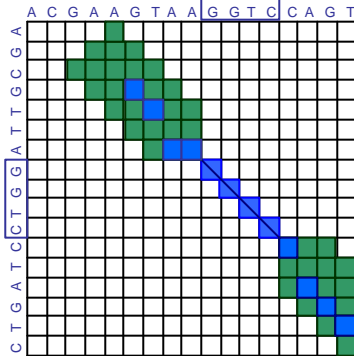
## Gapped BLAST

### Added features:

- Pairs of words can initiate alignment
- Extensions with gaps in a band around anchor

### Output:

GTAAGGTC CAGT  
GTTAGGTC-AGT



Lecture 3, Tuesday April 8, 2003

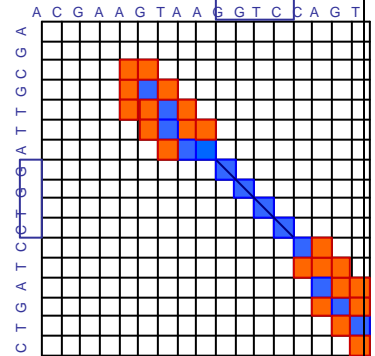
## Gapped BLAST

### Added features:

- Pairs of words can initiate alignment
- Nearby alignments are merged
- Extensions with gaps until score < T below best score so far

### Output:

GTAAGGTC CAGT  
GTTAGGTC-AGT



Lecture 3, Tuesday April 8, 2003

## Variants of BLAST

### MEGABLAST:

- Optimized to align very similar sequences
- Works best when  $k = 41 \geq 16$
- Linear gap penalty

### PSI-BLAST:

- BLAST produces many hits
- Those are aligned, and a pattern is extracted
- Pattern is used for next search; above steps iterated

### WU-BLAST: (Wash U BLAST)

- Optimized, added features

### BlastZ

- Combines BLAST/PatternHunter methodology

Lecture 3, Tuesday April 8, 2003

## Example

Query: gattacaccccgattacaccccgattaca (29 letters)

[2 mins]

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

>gi281701232.1|AC109904.0|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 tacaccccgattacaccccg 24  
Sbjct: 125138 tacaccccgattacaccccg 125158

Score = 34.2 bits (17),

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

Query: 4 tacaccccgattacaccccg 24  
Sbjct: 125104 tacaccccgattacaccccg 125124

>gi281730890.1|AC104321.7|Oryza sativa chromosome 3 BAC OSJNBa0052P07 genomic sequence, complete sequence Length = 139823 Score = 34.2 bits (17), Expect = 4.5 Identities = 20/21 (95%) Strand = Plus / Plus

Query: 4 tacaccccgattacaccccg 24  
Sbjct: 3891 tacaccccgattacaccccg 3911

Lecture 3, Tuesday April 8, 2003

## Example

Query: Human atoh enhancer, 179 letters

[1.5 min]

Result: 57 blast hits

```
1. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
2. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
3. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
4. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
5. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
6. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
7. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
8. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
9. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
10. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
11. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
12. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
13. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
14. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
15. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
16. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
17. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
18. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
19. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
20. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
21. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
22. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
23. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
24. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
25. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
26. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
27. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
28. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
29. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
30. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
31. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
32. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
33. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
34. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
35. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
36. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
37. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
38. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
39. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
40. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
41. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
42. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
43. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
44. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
45. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
46. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
47. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
48. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
49. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
50. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
51. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
52. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
53. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
54. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
55. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
56. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
57. 100% identity (179 letters) with sequence ATOH enhancer... 124-125
```

Lecture 3, Tuesday April 8, 2003

## BLAT: Blast-Like Alignment Tool

### Differences with BLAST:

1. Dictionary of the DB (BLAST: query)
2. Initiate alignment on any # of matching hits (BLAST: 1 or 2 hits)
3. More aggressive stitching-together of alignments
4. Special code to align mature mRNA to DNA

### Result: very efficient for:

- Aligning many seqs to a DB
- Aligning two genomes

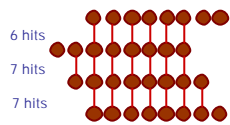
Lecture 3, Tuesday April 8, 2003

## PatternHunter

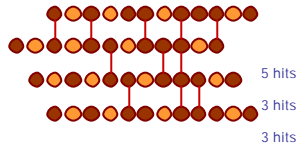
### Main features:

- Non-consecutive position words
- Highly optimized

### Consecutive Positions



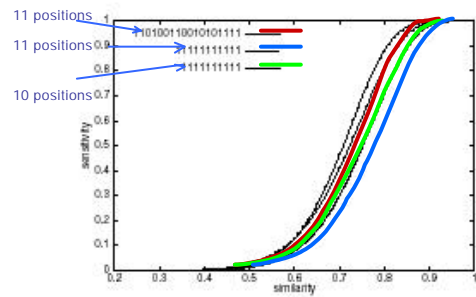
### Non-Consecutive Positions



On a 70% conserved region:

	<u>Consecutive</u>	<u>Non-consecutive</u>
Expected # hits:	1.07	0.97
Prob[at least one hit]:	0.30	0.47

## Advantage of Non-Consecutive Words



Lecture 3, Tuesday April 8, 2003