

Dynamic Programming and Applications

Michael Schatz

Bioinformatics Lecture 2
Quantitative Biology 2012



Exact Matching Review

Where is GATTACA in the human genome?
 $E=183,105$

Brute Force
(3 GB)

BANANA
BAN
ANA
NAN
ANA

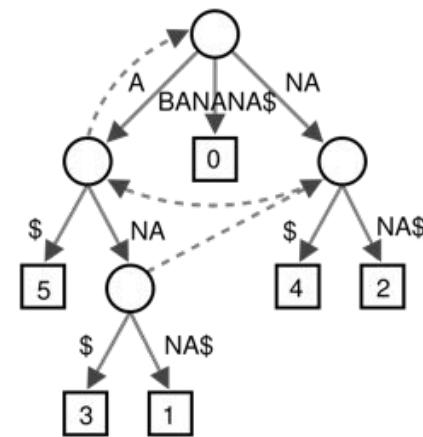
Naive
Slow & Easy

Suffix Array
(>15 GB)

6	\$
5	A\$
3	ANA\$
1	ANANA\$
0	BANANA\$
4	NA\$
2	NANA\$

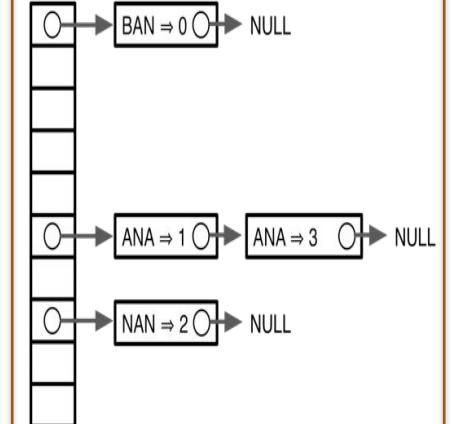
Vmatch, PacBio Aligner
Binary Search

Suffix Tree



MUMmer, MUMmerGPU
Tree Searching

Hash Table

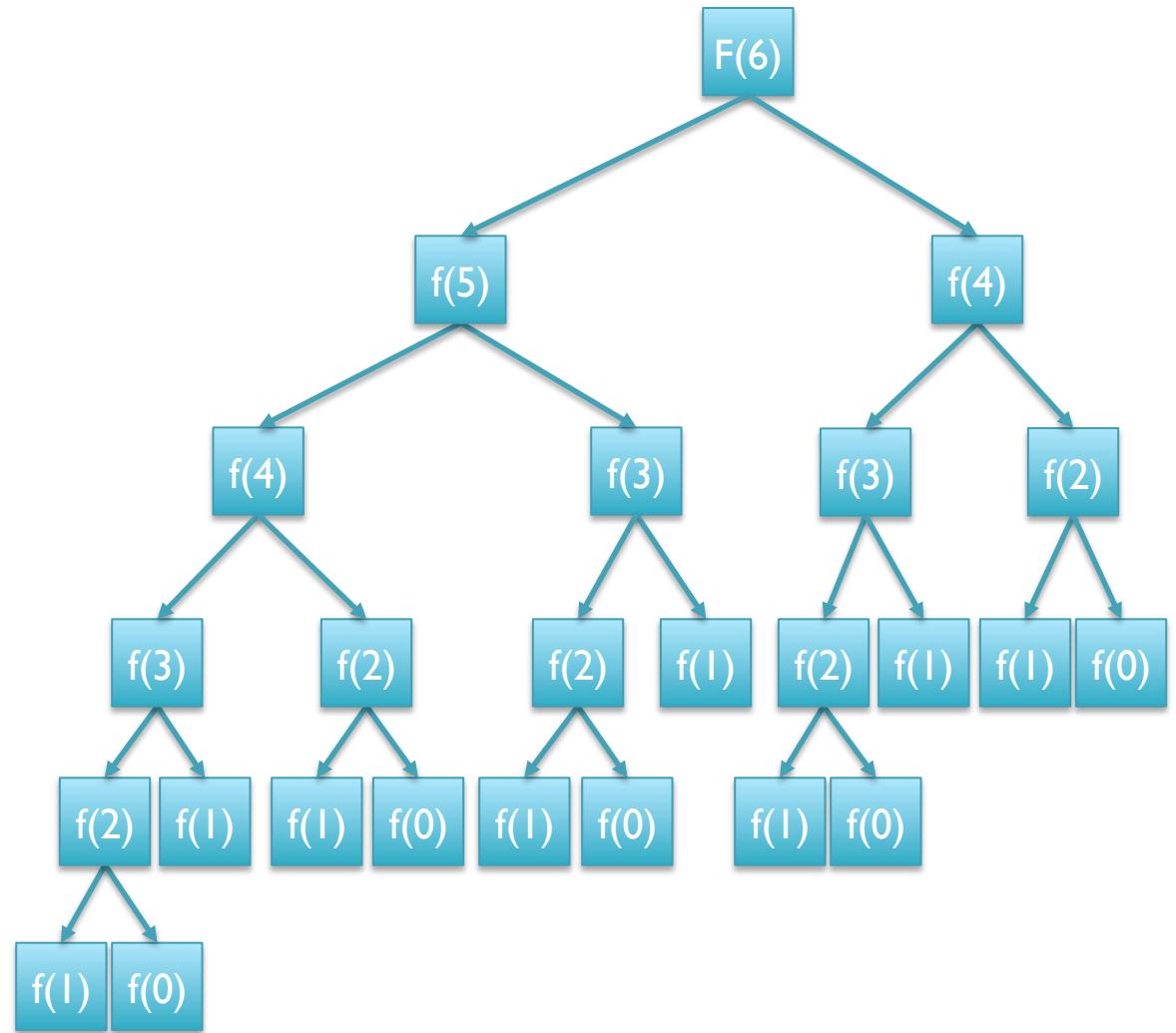
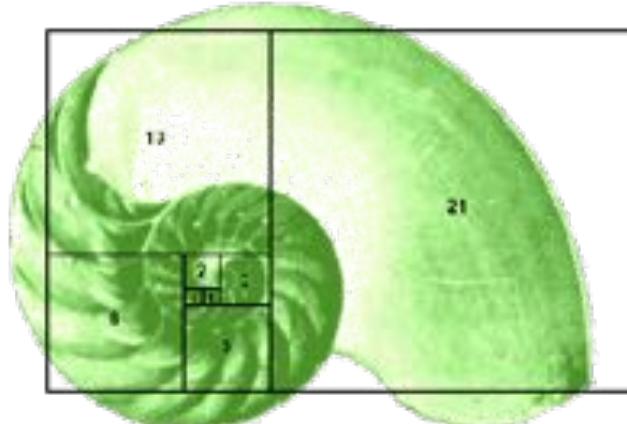


BLAST, MAQ, ZOOM,
RMAP, CloudBurst
Seed-and-extend

First:
A quick warm-up exercise

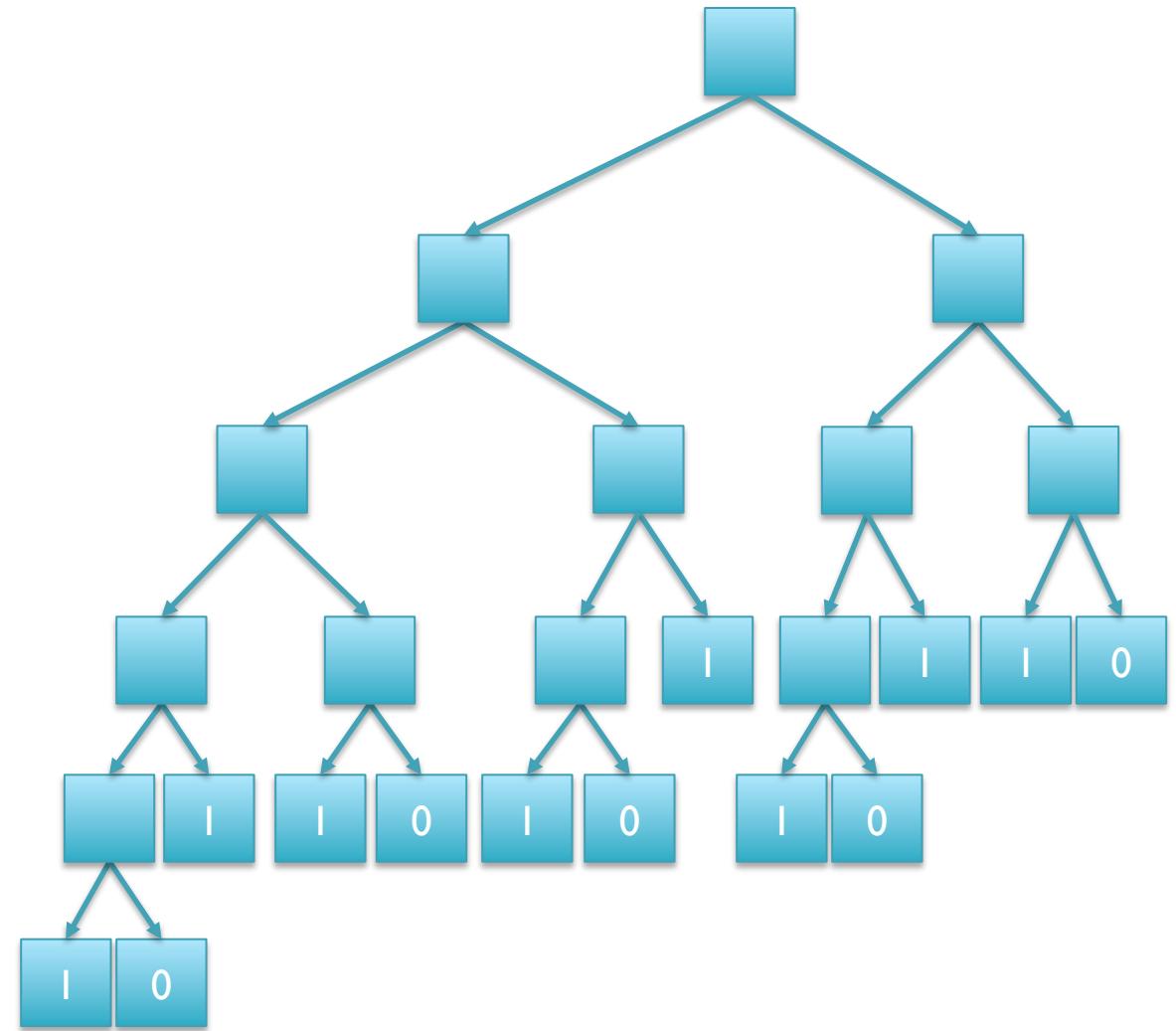
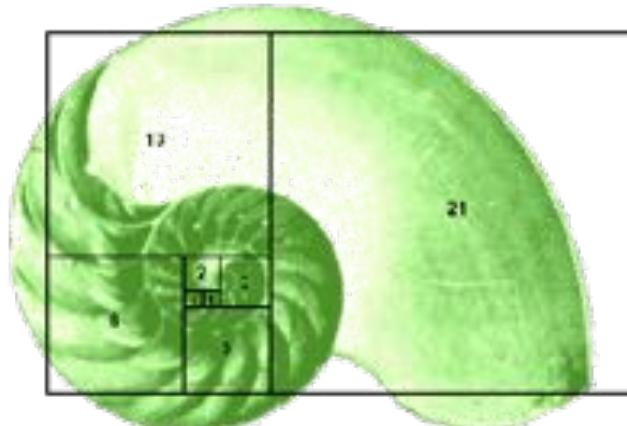
Fibonacci Sequence

```
int fib(n)
{
    if n == 0 return 0
    if n == 1 return 1
    return fib(n-1) + fib(n-2)
}
```



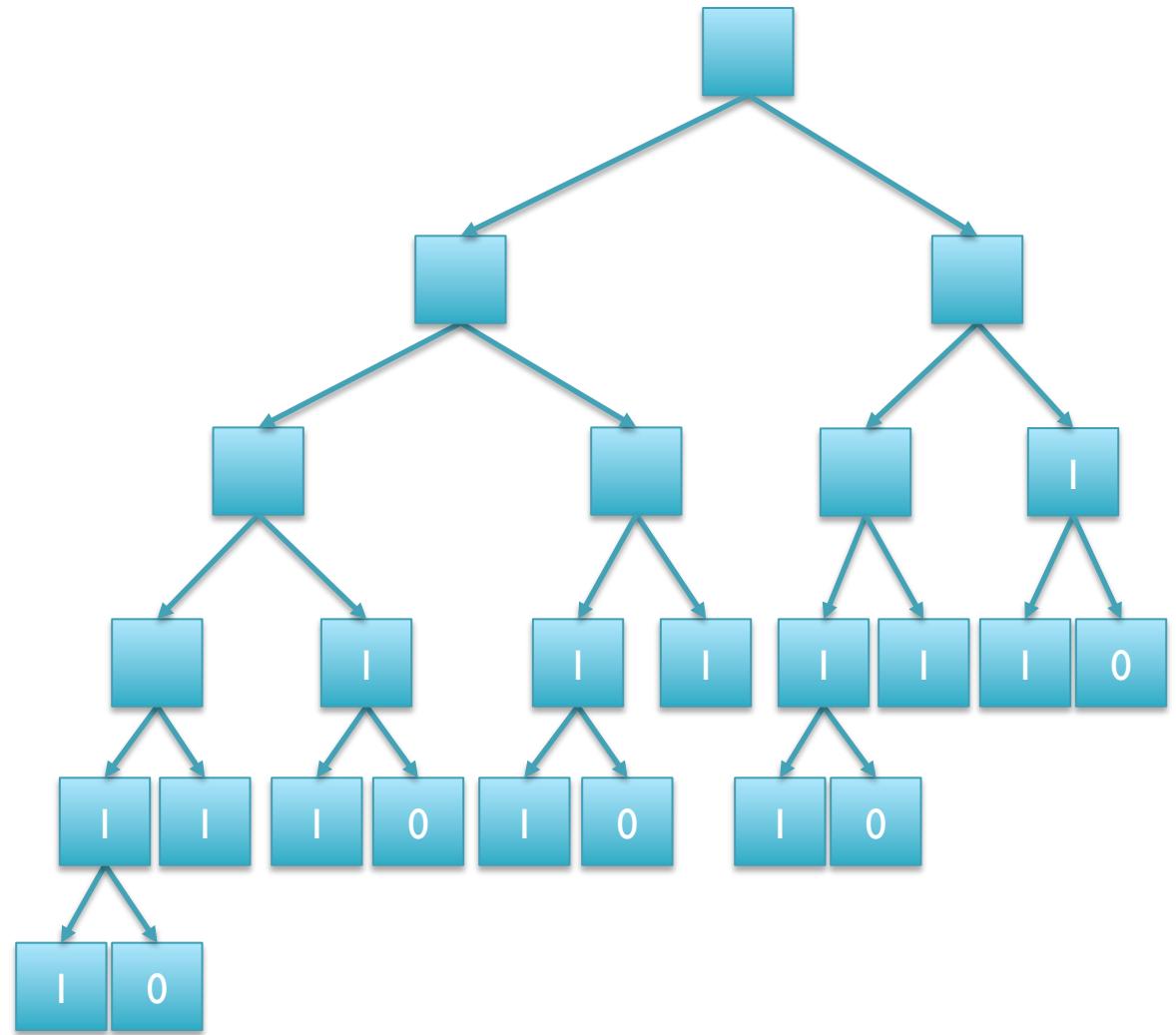
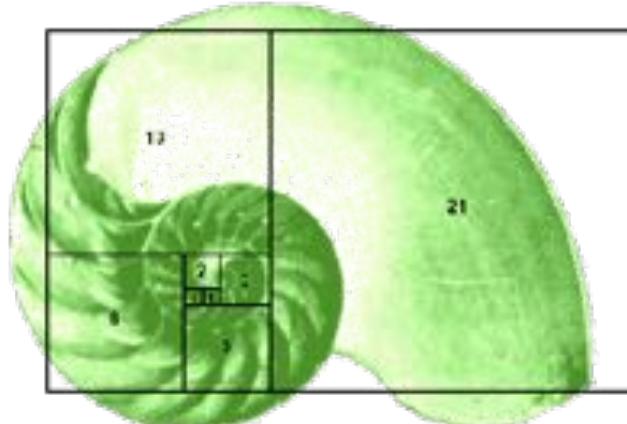
Fibonacci Sequence

```
int fib(n)
{
    if n == 0 return 0
    if n == 1 return 1
    return fib(n-1) + fib(n-2)
}
```



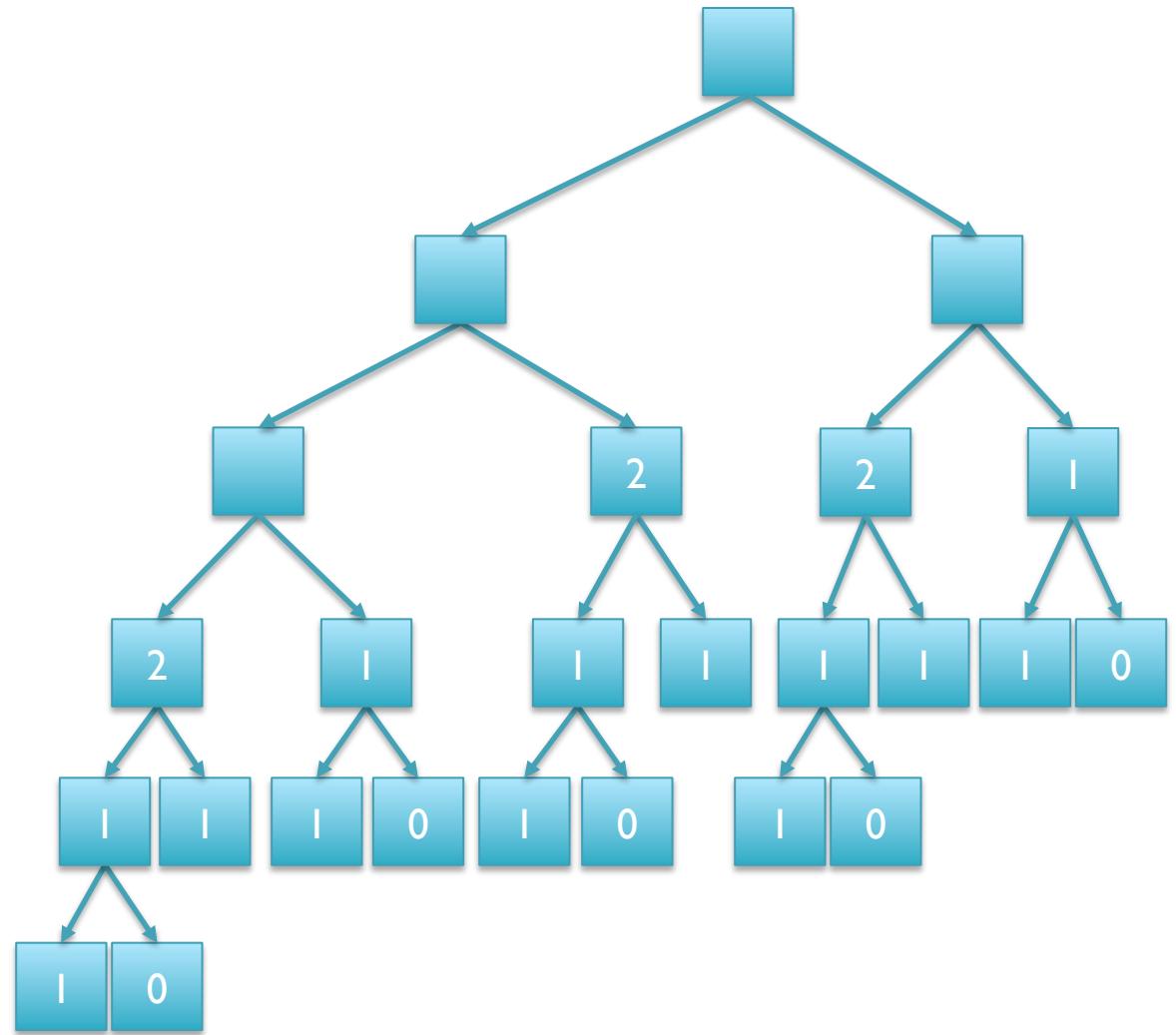
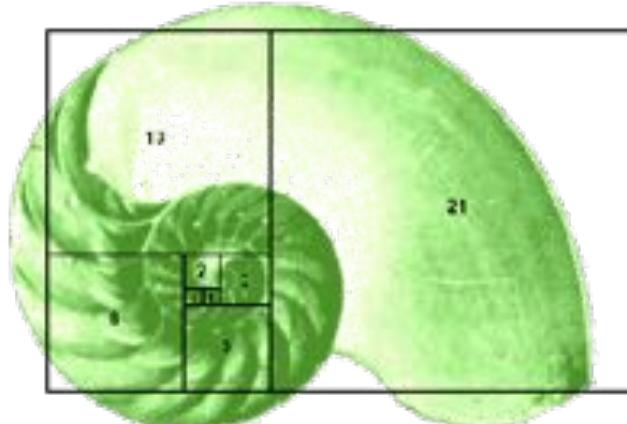
Fibonacci Sequence

```
int fib(n)
{
    if n == 0 return 0
    if n == 1 return 1
    return fib(n-1) + fib(n-2)
}
```



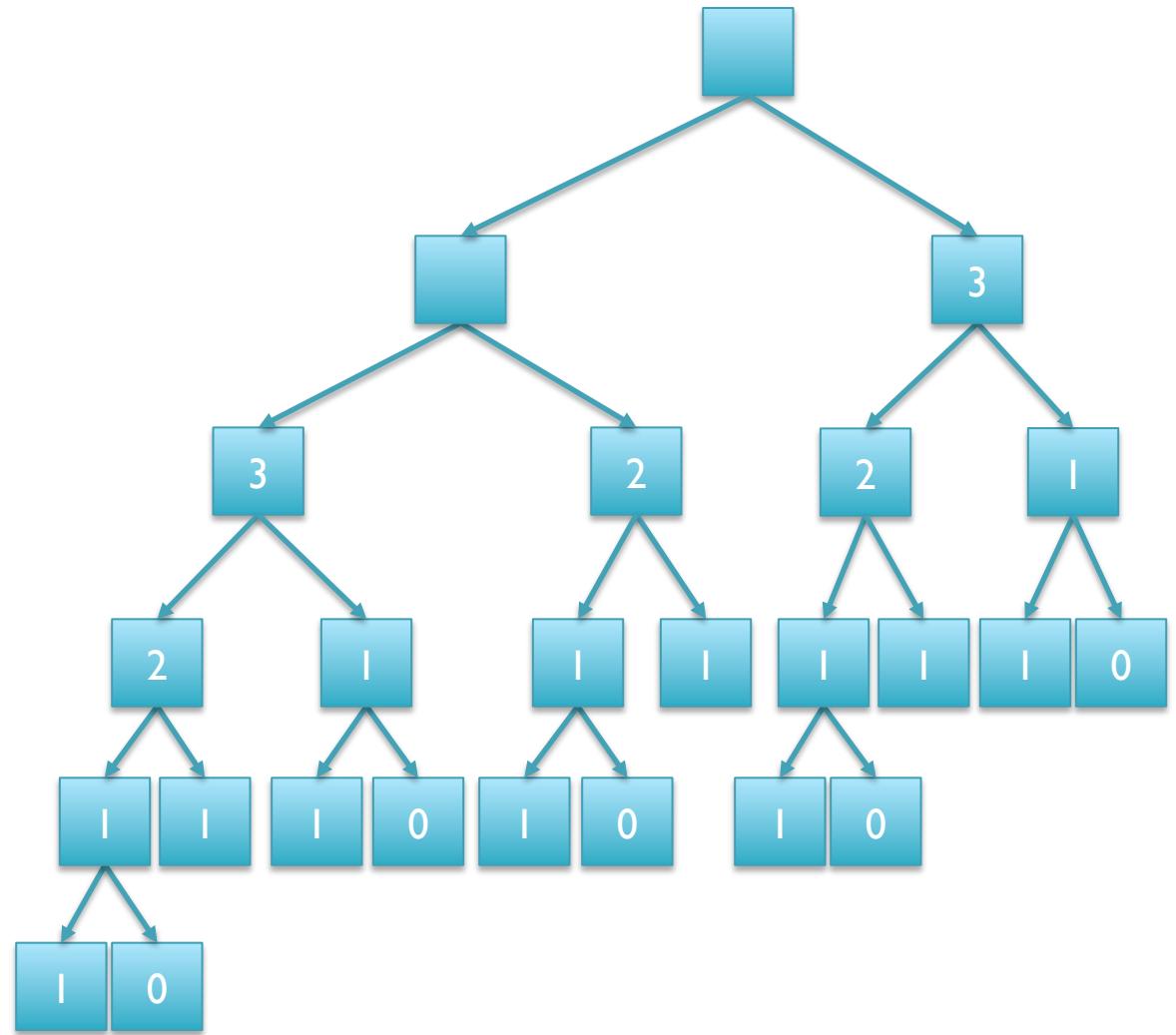
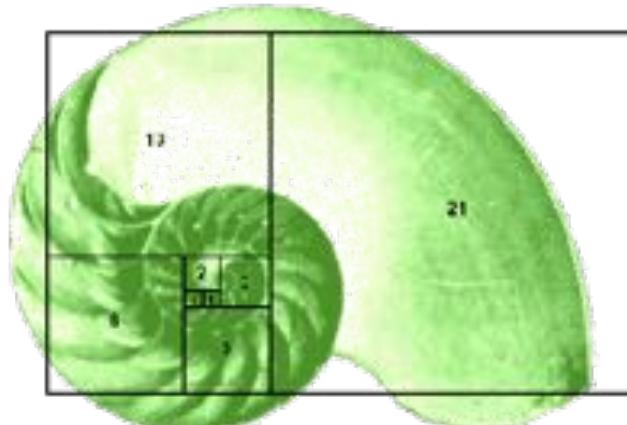
Fibonacci Sequence

```
int fib(n)
{
    if n == 0 return 0
    if n == 1 return 1
    return fib(n-1) + fib(n-2)
}
```



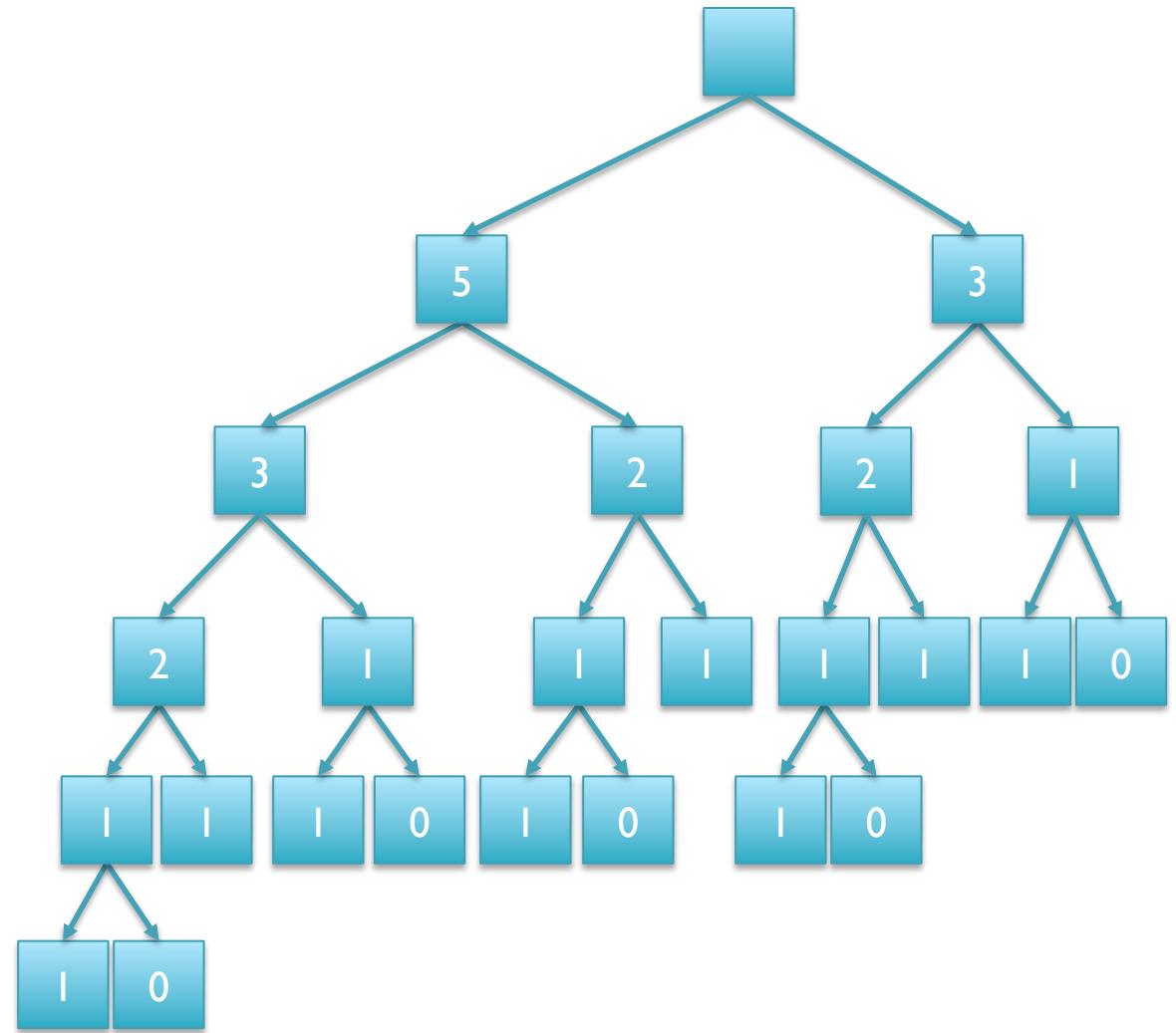
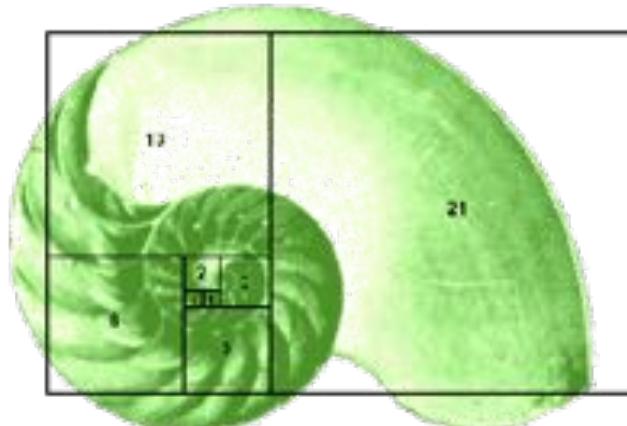
Fibonacci Sequence

```
int fib(n)
{
    if n == 0 return 0
    if n == 1 return 1
    return fib(n-1) + fib(n-2)
}
```



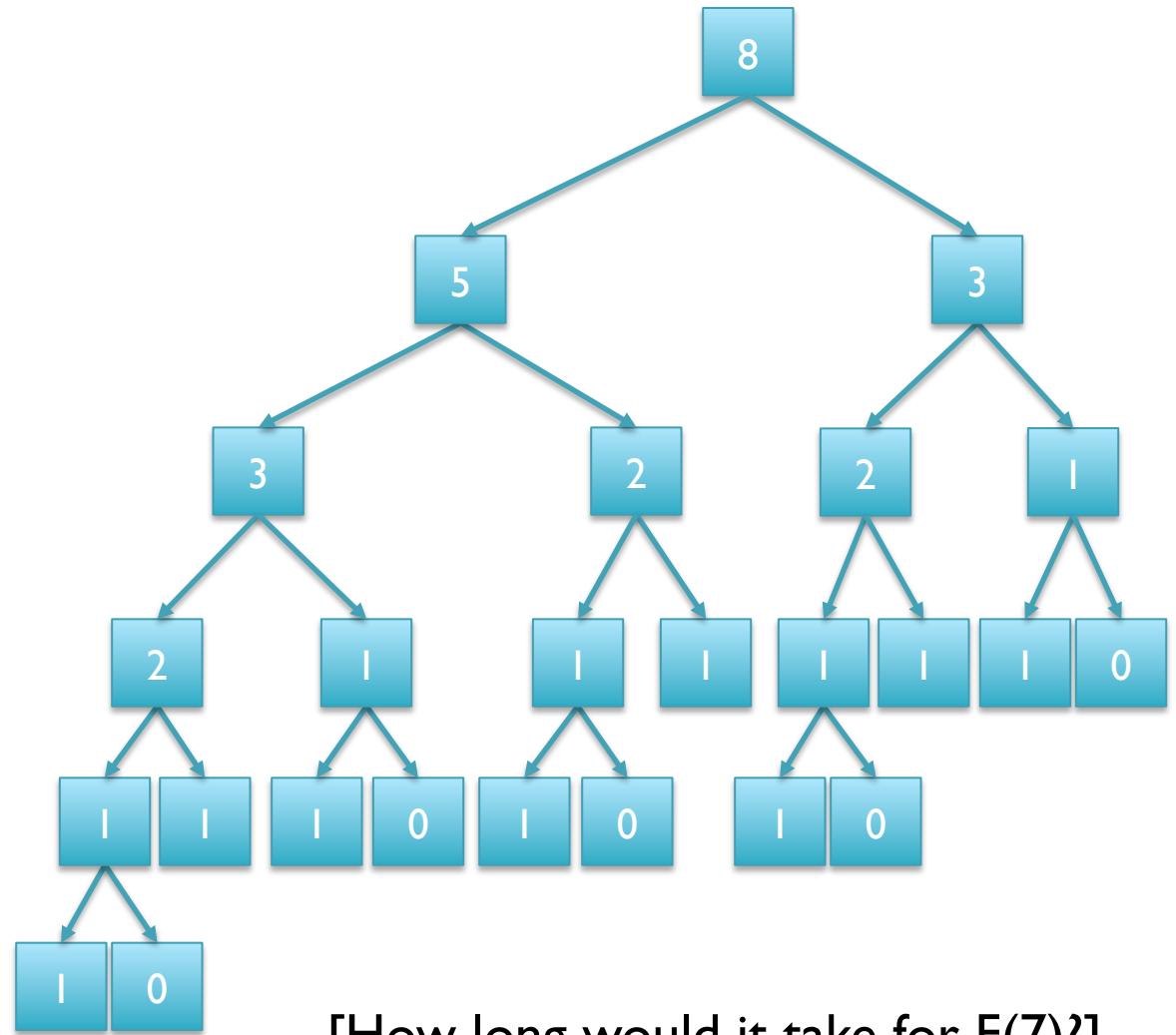
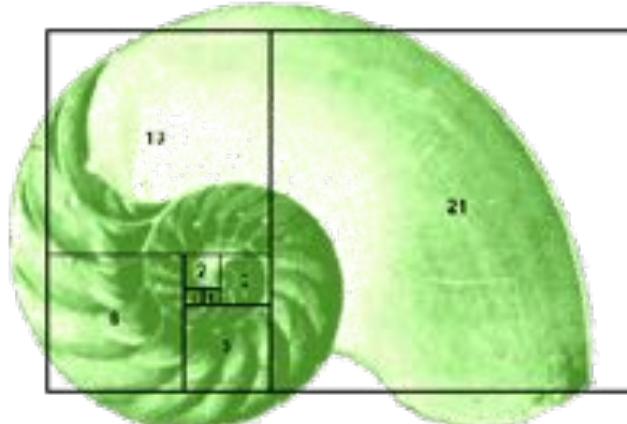
Fibonacci Sequence

```
int fib(n)
{
    if n == 0 return 0
    if n == 1 return 1
    return fib(n-1) + fib(n-2)
}
```



Fibonacci Sequence

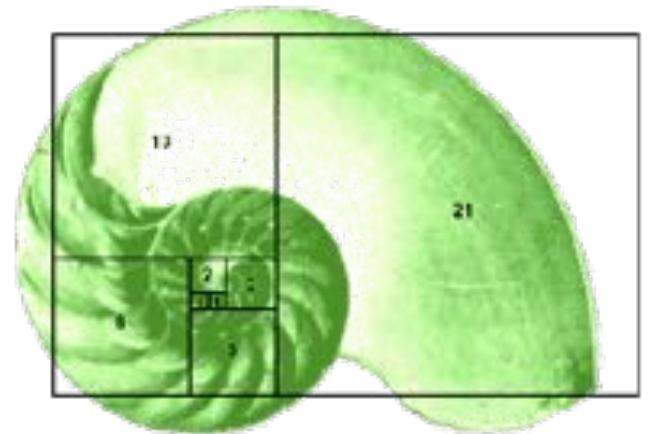
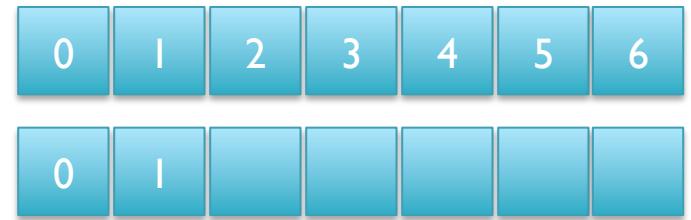
```
int fib(n)
{
    if n == 0 return 0
    if n == 1 return 1
    return fib(n-1) + fib(n-2)
}
```



[How long would it take for $F(7)$?]
[What is the running time?]

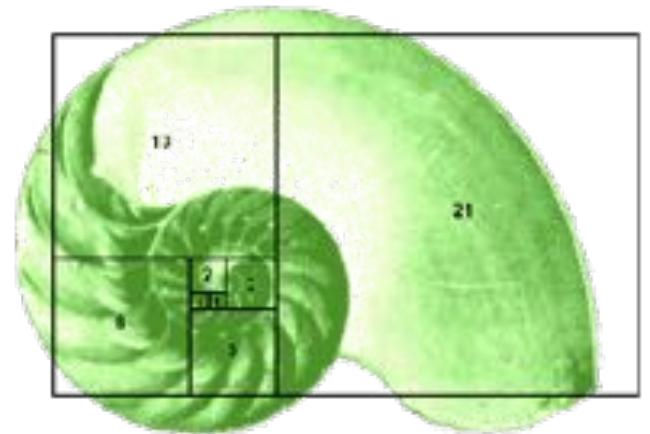
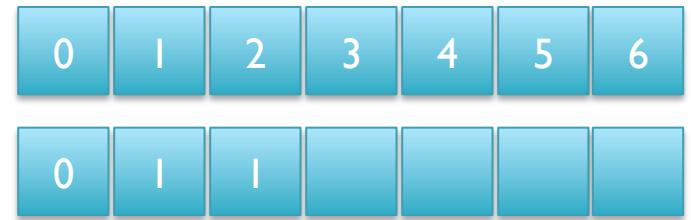
Bottom-up Fibonacci Sequence

```
int fib(n)
{
    ans[0] = 0
    ans[1] = 1
    for (int i = 2; i <= n; i++)
        ans[i] = ans[i-1] + ans[i-2]
    return ans[n];
}
```



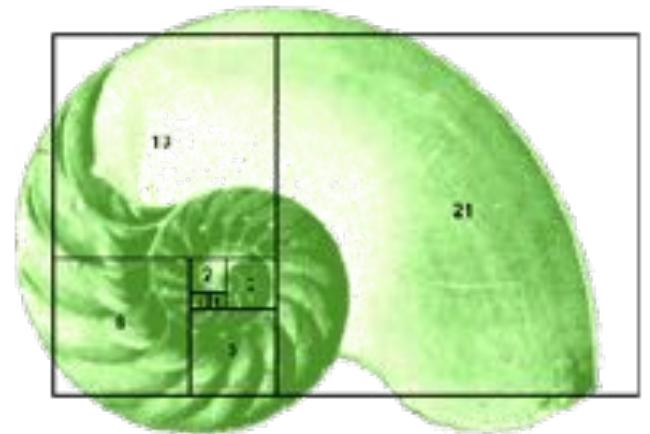
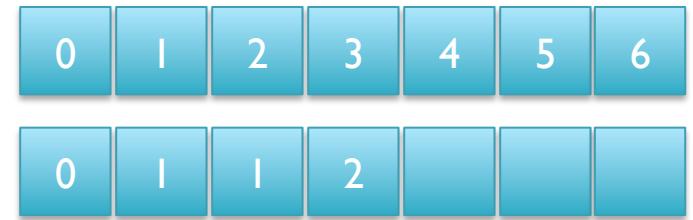
Bottom-up Fibonacci Sequence

```
int fib(n)
{
    ans[0] = 0
    ans[1] = 1
    for (int i = 2; i <= n; i++)
        ans[i] = ans[i-1] + ans[i-2]
    return ans[n];
}
```



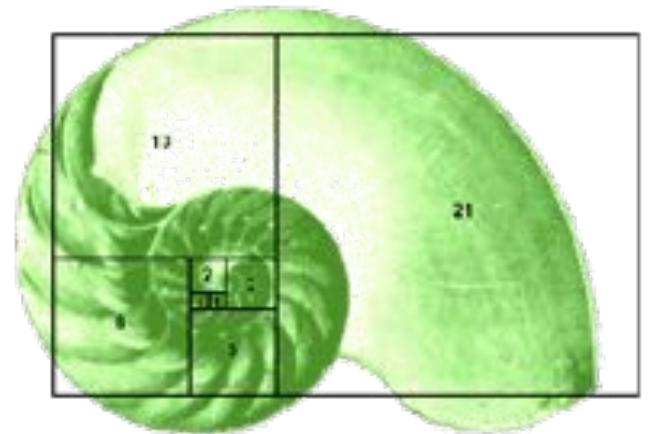
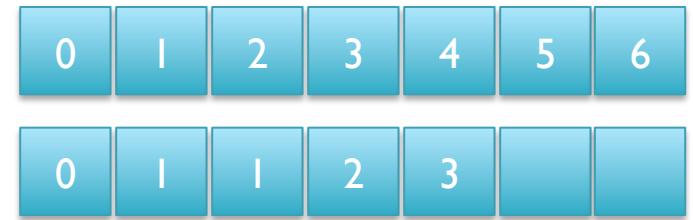
Bottom-up Fibonacci Sequence

```
int fib(n)
{
    ans[0] = 0
    ans[1] = 1
    for (int i = 2; i <= n; i++)
        ans[i] = ans[i-1] + ans[i-2]
    return ans[n];
}
```



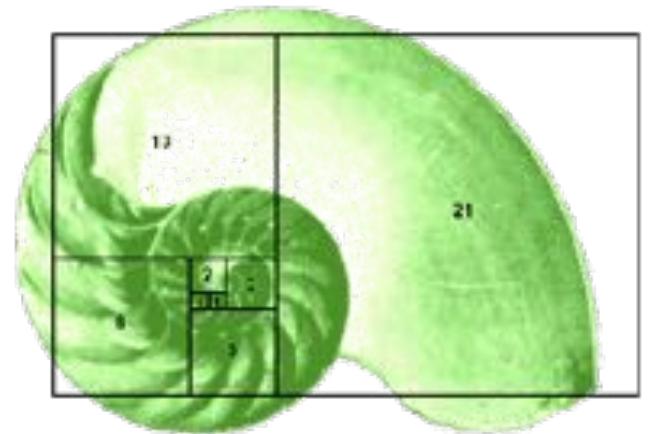
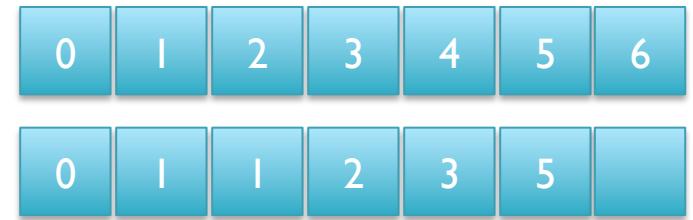
Bottom-up Fibonacci Sequence

```
int fib(n)
{
    ans[0] = 0
    ans[1] = 1
    for (int i = 2; i <= n; i++)
        ans[i] = ans[i-1] + ans[i-2]
    return ans[n];
}
```



Bottom-up Fibonacci Sequence

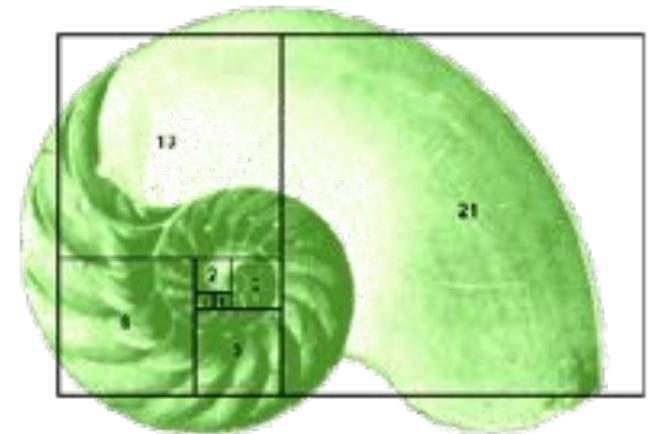
```
int fib(n)
{
    ans[0] = 0
    ans[1] = 1
    for (int i = 2; i <= n; i++)
        ans[i] = ans[i-1] + ans[i-2]
    return ans[n];
}
```



Bottom-up Fibonacci Sequence

```
int fib(n)
{
    ans[0] = 0
    ans[1] = 1
    for (int i = 2; i <= n; i++)
        ans[i] = ans[i-1] + ans[i-2]
    return ans[n];
}
```

0	1	2	3	4	5	6
0	1	1	2	3	5	8

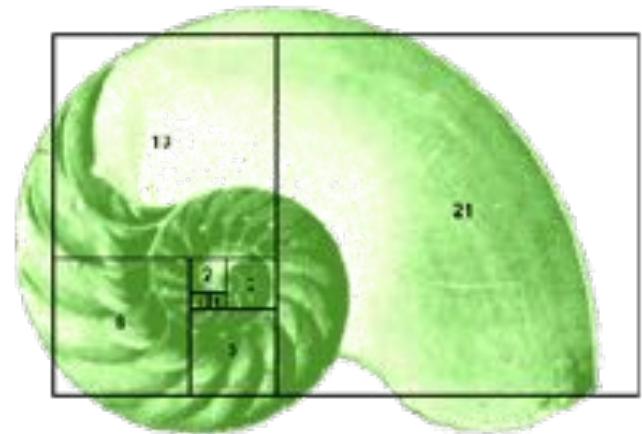


Bottom-up Fibonacci Sequence

```
int fib(n)
{
    ans[0] = 0
    ans[1] = 1
    for (int i = 2; i <= n; i++)
        ans[i] = ans[i-1] + ans[i-2]
    return ans[n];
}
```

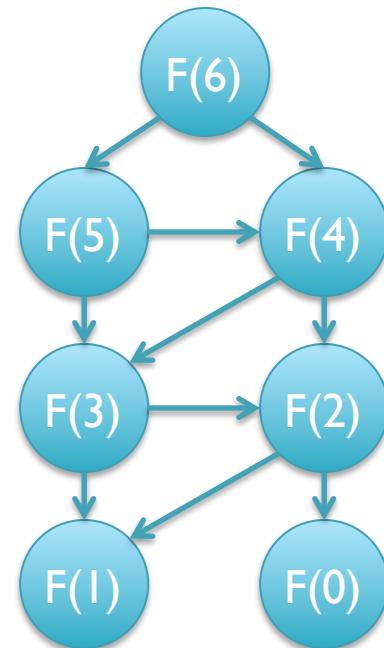
0	1	2	3	4	5	6
0	1	1	2	3	5	8

[How long will it take for F(7)?]
[What is the running time?]



Dynamic Programming

- General approach for solving (some) complex problems
 - When applicable, the method takes far less time than naive methods.
 - Polynomial time ($O(n)$ or $O(n^2)$) instead of exponential time ($O(2^n)$ or $O(3^n)$)
- Requirements:
 - Overlapping subproblems
 - Optimal substructure
- Applications:
 - Fibonacci
 - Longest Increasing Subsequence
 - Sequence alignment, Dynamic Time Warp, Viterbi
- Not applicable:
 - Traveling salesman problem, Clique finding, Subgraph isomorphism, ...
 - The cheapest flight from airport A to airport B involves a single connection through airport C, but the cheapest flight from airport A to airport C involves a connection through some other airport D.



Longest Increasing Subsequence

- Given a sequence of N numbers $A_1, A_2, A_3, \dots, A_N$, find the longest monotonically increasing subsequence
 - 29, 6, 14, 31, 39, 78, 63, 50, 13, 64, 61, 19
- Greedy approach (always extend the subsequence if you can):
 - 29, 6, 14, 31, 39, 78, 63, 50, 13, 64, 61, 19 => 4
- Brute force:
 - Try all possible $O(2^n)$ subsequences
 - 29, 6, 14, 31, 39, 78, 63, 50, 13, 64, 61, 19 => 1
 - 29, 6, 14, 31, 39, 78, 63, 50, 13, 64, 61, 19 => invalid
 - 29, 6, 14, 31, 39, 78, 63, 50, 13, 64, 61, 19 => invalid
 - 29, 6, 14, 31, 39, 78, 63, 50, 13, 64, 61, 19 => 2
 - ...

Longest Increasing Subsequence

- Idea:
 - The solution for all N numbers depends on the solution for the first N-1
 - Look through the previous values to find the longest subsequence ending at X such that $A_X < A_N$
- Dynamic Programming:
 - Def: $L[j]$ is the longest increasing subsequence ending at position j
 - Base case: $L[0] = 0$ Recurrence: $L[j] = \max_{\substack{i < j \\ A[i] < A[j]}} \{L[i]\} + 1$ LIS = $\max\{L[i]\}$

Index	0	1	2	3	4	5	6	7	8	9	10	11	12
Value	-	29	6	14	31	39	78	63	50	13	64	61	19
LIS	0	1	1	2	3	4	5	5	5	2	6	5	3
Prev	-	0	0	2	3	4	5	5	5	2	6	5	3

[What's the LIS of 0,8,4,12,2,10,6,14,1,9,5,13,3,11,7,15 ?]

Longest Increasing Subsequence

LIS Pseudocode

// Initialize

$L[0] = 0; P[0] = 0$

// Iteratively apply recurrence

for $j = 1$ to N

 // find the best LIS to extend

 best = 0

 for $i = 1$ to j

 if $((A[i] < A[j])) \&& (L[i] > L[best])$ best = i

$L[j] = L[best] + 1; P[j] = best$

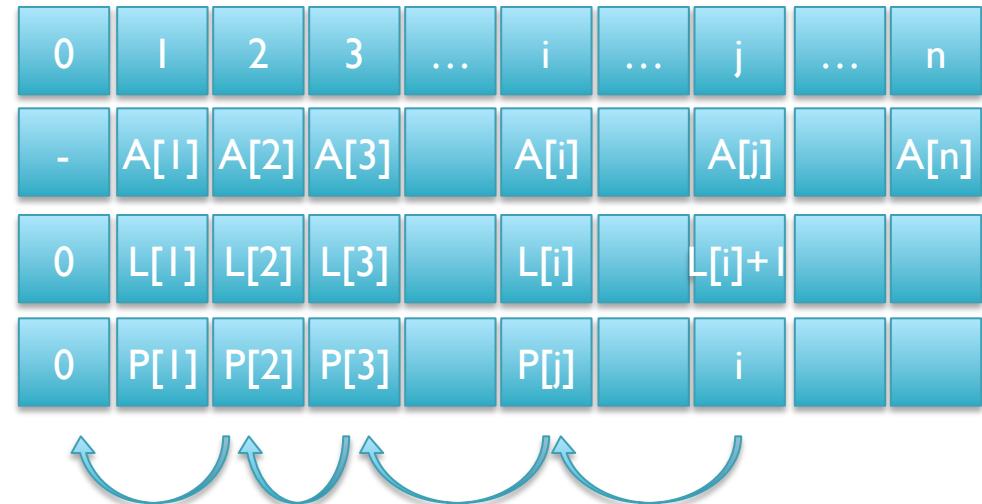
// Scan the L array to find the overall LIS

LIS = 0

for $j = 1$ to N

 if $(L[j] > LIS)$ LIS = $L[j]$

print "The LIS is \$LIS"



[What's the running time?]

In-exact alignment

- Where is GATTACA *approximately* in the human genome?
 - And how do we efficiently find them?
- It depends...
 - Define 'approximately'
 - Hamming Distance, Edit distance, or Sequence Similarity
 - Ungapped vs Gapped vs Affine Gaps
 - Global vs Local
 - All positions or the single 'best'?
 - Efficiency depends on the data characteristics & goals
 - Bowtie: BWT alignment for short read mapping
 - Smith-Waterman: Exhaustive search for optimal alignments
 - BLAST: Hash based homology searches
 - MUMmer: Suffix Tree based whole genome alignment

Similarity metrics

- Hamming distance
 - Count the number of substitutions to transform one string into another

GATTACA	ATTACCC
x	xx xx x
GATCACA	GATTACA
1	5

- Edit distance
 - The minimum number of substitutions, insertions, or deletions to transform one string into another

GATTACA	-ATTACCC
x	x xx
GATCACA	GATTAC-A
1	3

Edit Distance Example

AGCACACCA → ACACACTA in 4 steps

AGCACACCA → (1. change G to C)

ACCACACCA → (2. delete C)

ACACACCA → (3. change A to T)

ACACACTT → (4. insert A after T)

ACACACTA → done

[Is this the best we can do?]

Edit Distance Example

AGCACACCA → ACACACTA in 3 steps

AGCACACCA → (1. change G to C)

ACCACACCA → (2. delete C)

ACACACCA → (3. insert T after 3rd C)

ACACACTA → done

[Is this the best we can do?]

Reverse Engineering Edit Distance

$$D(\text{AGCACACA}, \text{ ACACACTA}) = ?$$

Imagine we already have the optimal alignment of the strings, the last column can only be 1 of 3 options:

... M	... I	... D
...A	...-	...A
...A	...A	...-

The optimal alignment of last two columns is then 1 of 9 possibilities

... MM ... IM ... DM	... MI ... II ... DI	... MD ... ID ... DD
...CA ...-A ...CA	...A- ...-- ...A-	...CA ...-A ...CA
...TA ...TA ...-A	...TA ...TA ...-A	...A- ...A- ...--

The optimal alignment of the last three columns is then 1 of 27 possibilities...

... M I D ...
...X...	...-...	...X...
...Y...	...Y...	...-...

Eventually spell out every possible sequence of {I,M,D}

Reverse Engineering Edit Distance

$$D(\text{AGCACACA}, \text{ ACACACTA}) = ?$$

Imagine we already have the optimal alignment of the strings, the last column can only be 1 of 3 options:

...M	...I	...D
...A	...-	...A
...A	...A	...-
D=2	D=2	D=3

If these are the edit distances of the three possible optimal alignments up to the last column, the overall optimal alignment must use option M and the final distance is $2+0=2$

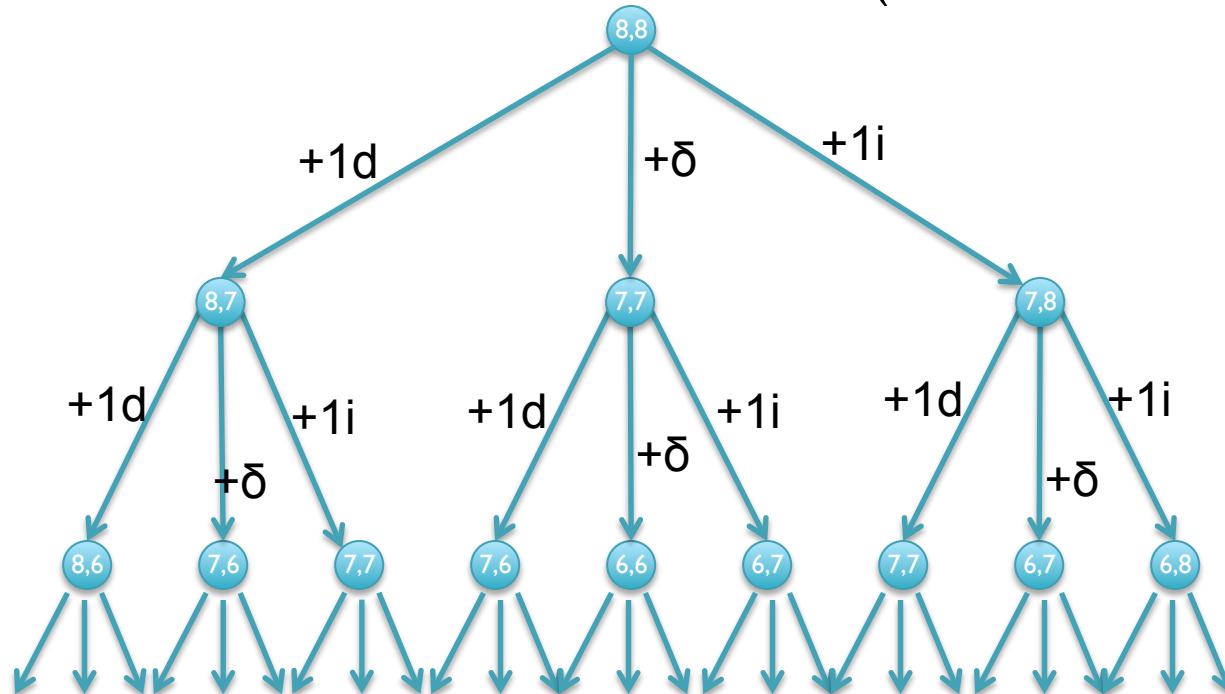
This suggests an overall structure to the computation:

The edit distance of the overall alignment is computed from the edit distance of three possible alignments with the last column removed.... these values are themselves computed by removing the second to last column, and so forth down to the trivial base case of an empty string.

Recursive solution

- Computation of D is a recursive process.
 - At each step, we only allow matches, substitutions, and indels
 - $D(i,j)$ in terms of $D(i',j')$ for $i' \leq i$ and $j' \leq j$.

$$D(\text{AGCACACA}, \text{ACACACTA}) = \min\{D(\text{AGCACACA}, \text{ACACACT}) + I, \\ D(\text{AGCACAC}, \text{ACACACTA}) + I, \\ D(\text{AGCACAC}, \text{ACACACT}) + \delta(A, A)\}$$



[What is the running time?]

Dynamic Programming

- We could code this as a recursive function call...
...with an exponential number of function evaluations
- There are only $(n+1) \times (m+1)$ pairs i and j
 - We are evaluating $D(i,j)$ multiple times
- Compute $D(i,j)$ bottom up.
 - Start with smallest $(i,j) = (1,1)$.
 - Store the intermediate results in a table.
 - Compute $D(i,j)$ after $D(i-1,j)$, $D(i,j-1)$, and $D(i-1,j-1)$

Recurrence Relation for D

Find the edit distance (minimum number of operations to convert one string into another) in $O(mn)$ time

- Base conditions:

- $D(i,0) = i$, for all $i = 0, \dots, n$
- $D(0,j) = j$, for all $j = 0, \dots, m$

- For $i > 0, j > 0$:

$$D(i,j) = \min \{ \begin{aligned} & D(i-1,j) + 1, && // \text{align 0 chars from S, 1 from T} \\ & D(i,j-1) + 1, && // \text{align 1 chars from S, 0 from T} \\ & D(i-1,j-1) + \delta(S(i),T(j)) && // \text{align 1+1 chars} \end{aligned} \}$$

[Why do we want the min?]

Dynamic Programming Matrix

		A	C	A	C	A	C	T	A
	0	I	2	3	4	5	6	7	8
A	I								
G	2								
C	3								
A	4								
C	5								
A	6								
C	7								
A	8								

[What does the initialization mean?]

Dynamic Programming Matrix

		A	C	A	C	A	C	T	A
	0	I	2	3	4	5	6	7	8
A	I	0							
G	2								
C	3								
A	4								
C	5								
A	6								
C	7								
A	8								

$$D[A, A] = \min\{D[A,] + 1, D[, A] + 1, D[,] + \delta(A, A)\}$$

Dynamic Programming Matrix

		A	C	A	C	A	C	T	A
	0	I	2	3	4	5	6	7	8
A	I	0	I						
G	2								
C	3								
A	4								
C	5								
A	6								
C	7								
A	8								

$$D[A, AC] = \min\{D[A, A]+1, D[AC]+1, D[A]+\delta(A, C)\}$$

Dynamic Programming Matrix

		A	C	A	C	A	C	T	A
	0	I	2	3	4	5	6	7	8
A	I	0	I	2					
G	2								
C	3								
A	4								
C	5								
A	6								
C	7								
A	8								

$$D[A,ACA] = \min\{D[A,AC]+1, D[ACA]+1, D[AC]+\delta(A,A)\}$$

Dynamic Programming Matrix

		A	C	A	C	A	C	T	A
	0	1	2	3	4	5	6	7	8
A	I	0	I	2	3	4	5	6	7
G	2								
C	3								
A	4								
C	5								
A	6								
C	7								
A	8								

$$D[A, ACACACTA] = 7$$

-----A

***** |

ACACACTA

[What about the other A?]

Dynamic Programming Matrix

		A	C	A	C	A	C	T	A
	<u>0</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>	<u>7</u>	<u>8</u>
A	I	0	I	2	3	<u>4</u>	5	6	7
G	2	I	I	2	3	4	<u>5</u>	<u>6</u>	<u>7</u>
C	3								
A	4								
C	5								
A	6								
C	7								
A	8								

$$D[AG, ACACACTA] = 7$$

----AG--

**** | ***

ACACACTA

Dynamic Programming Matrix

		A	C	A	C	A	C	T	A
	<u>0</u>	1	2	3	4	5	6	7	8
A	1	<u>0</u>	1	2	3	4	5	6	7
G	2	<u>1</u>	1	2	3	4	5	6	7
C	3	2	<u>1</u>	2	2	3	4	5	6
A	4	3	2	<u>1</u>	2	2	3	4	5
C	5	4	3	2	<u>1</u>	2	2	3	4
A	6	5	4	3	2	<u>1</u>	2	3	3
C	7	6	5	4	3	2	<u>1</u>	<u>2</u>	3
A	8	7	6	5	4	3	2	2	<u>2</u>

$$D[AGCACACA, ACACACTA] = 2$$

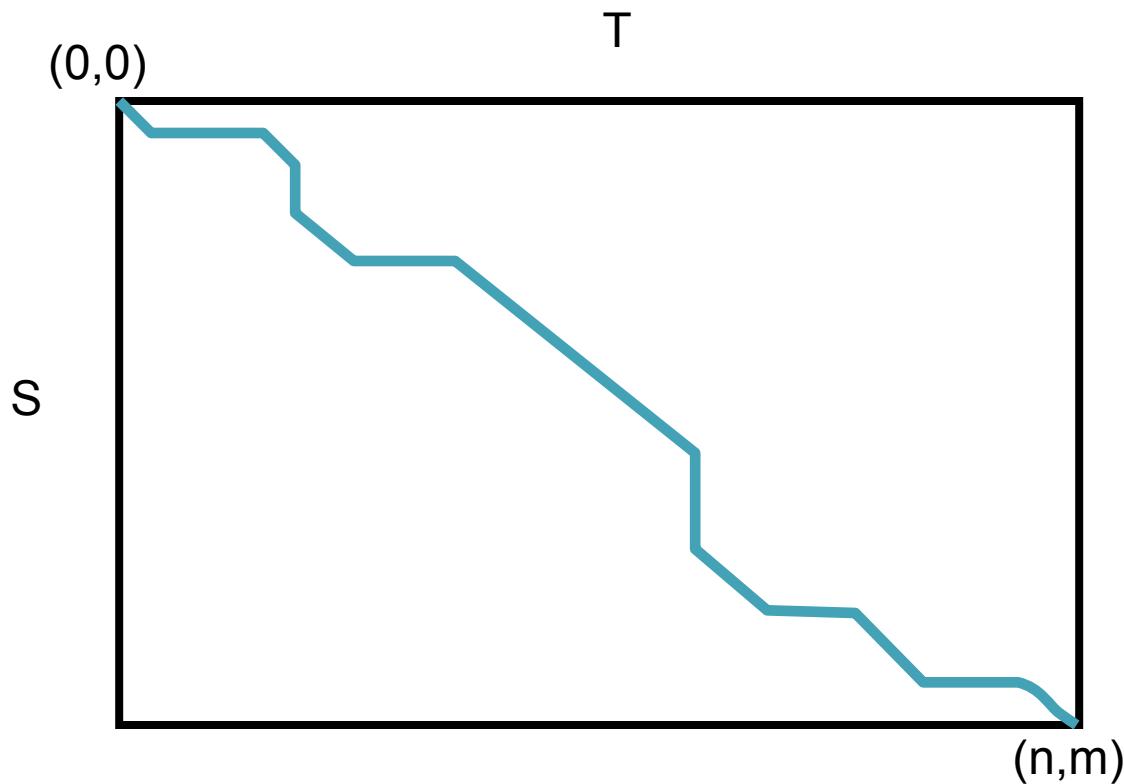
AGCACAC-A

| * | | | | *

A-CACACTA

[Can we do it any better?]

Global Alignment Schematic



- A high quality alignment will stay close to the diagonal
 - If we are only interested in high quality alignments, we can skip filling in cells that can't possibly lead to a high quality alignment
 - Find the global alignment with at most edit distance d : $O(2dn)$

Break

Local vs. Global Alignment

- The Global Alignment Problem tries to find the best path between vertices $(0,0)$ and (n,m) in the edit graph.
- The Local Alignment Problem tries to find the best path among paths between **arbitrary vertices** (i,j) and (i',j') in the edit graph.

[How many $(i,j) \times (i',j')$ pairs are there?]

Local vs. Global Alignment (cont'd)

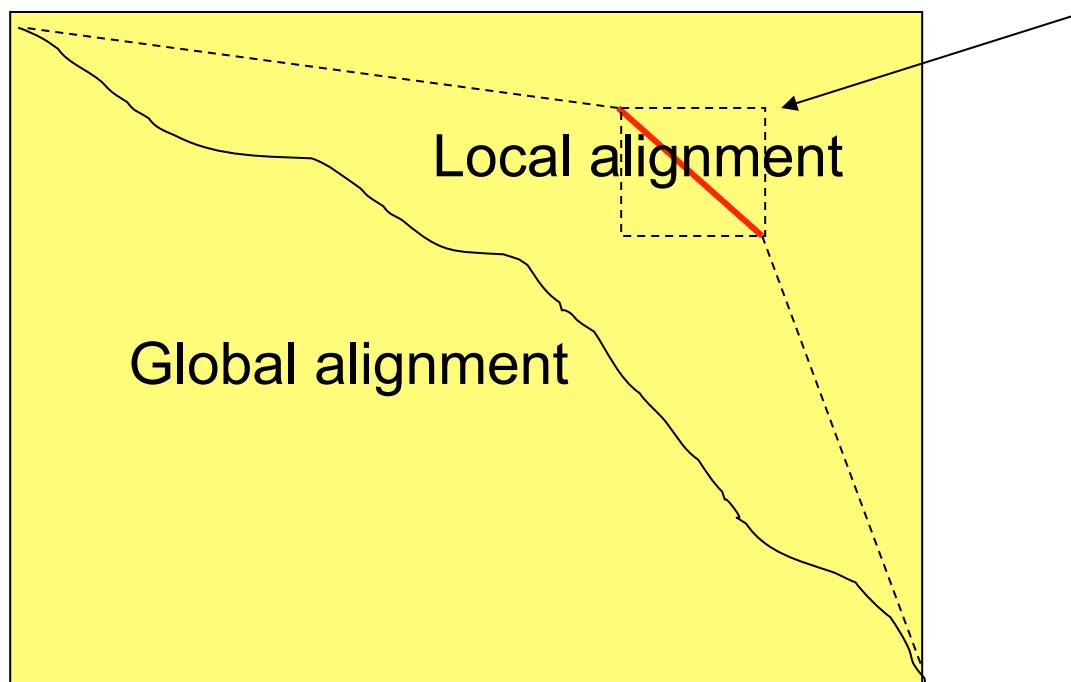
- **Global Alignment**

```
--T--CC-C-AGT--TATGT-CAGGGGACACG-A-GCATGCAGA-GAC  
| | | | | | | | | | | | | | | | | | | | | | | | | | |  
AATTGCCGCC-GTCGT-T-TTCAG----CA-GTTATG-T-CAGAT--C
```

- **Local Alignment**—better alignment to find conserved segment

```
tccCAGTTATGTCAGggacacgagcatgcagagac  
|||||||||||  
aattgccgccgtcgtttcagCAGTTATGTCAGatc
```

Local Alignment: Example



Compute a “mini”
Global Alignment to
get Local

Edit Distance and Global Similarity

$$D(i,j) = \min \{ \begin{aligned} & D(i-l, j) + l, \\ & D(i, j-l) + l, \\ & D(i-l, j-l) + \delta(S(i), T(j)) \end{aligned} \}$$

$s = 4 \times 4$ or 20×20 scoring matrix

$$S(i,j) = \max \{ \begin{aligned} & S(i-l, j) + l, \\ & S(i, j-l) + l, \\ & S(i-l, j-l) + s(S(i), T(j)) \end{aligned} \}$$

[Why max?]

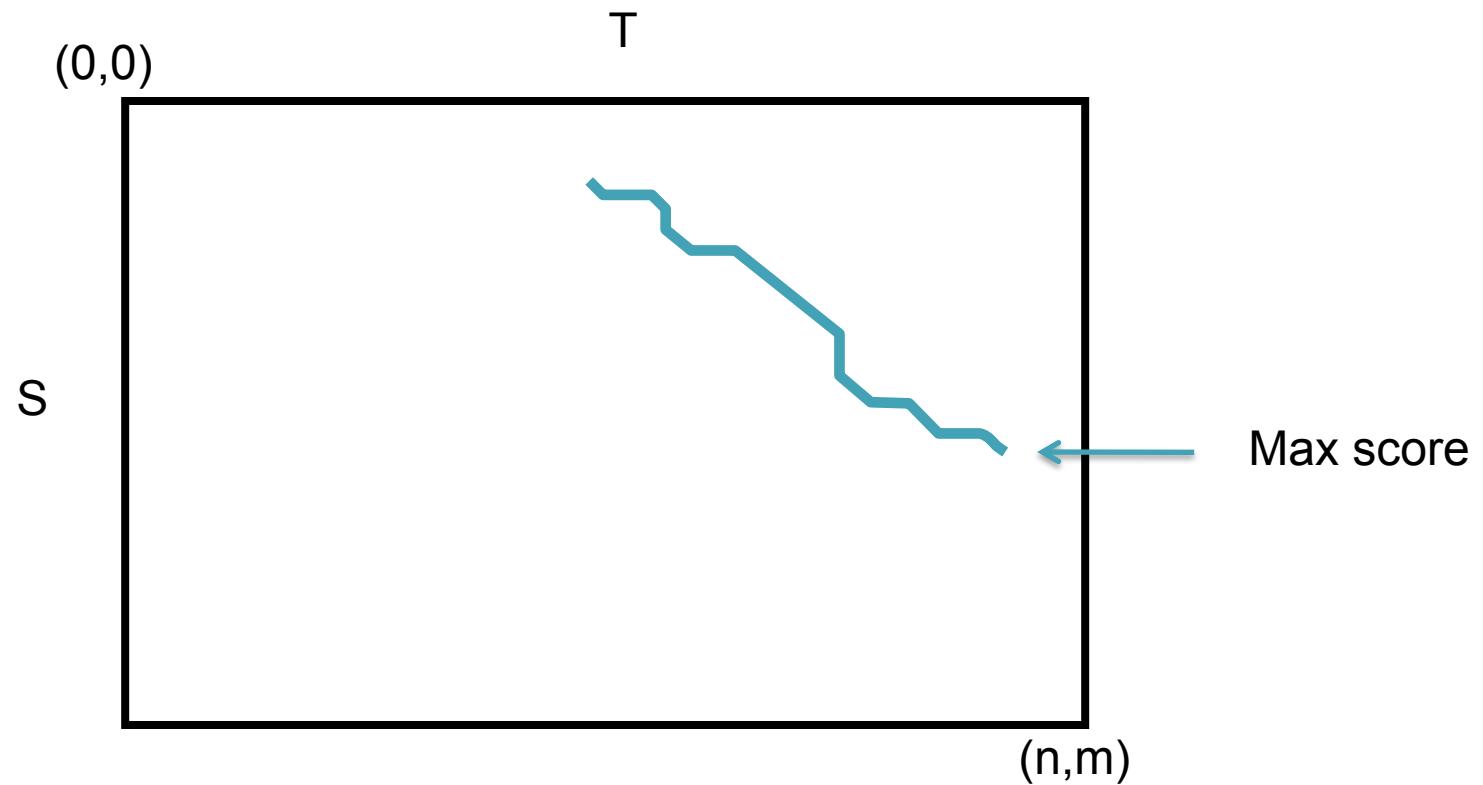
The Local Alignment Recurrence

- The largest value of $s_{i,j}$ over the whole edit graph is the score of the best local alignment.
- The recurrence:

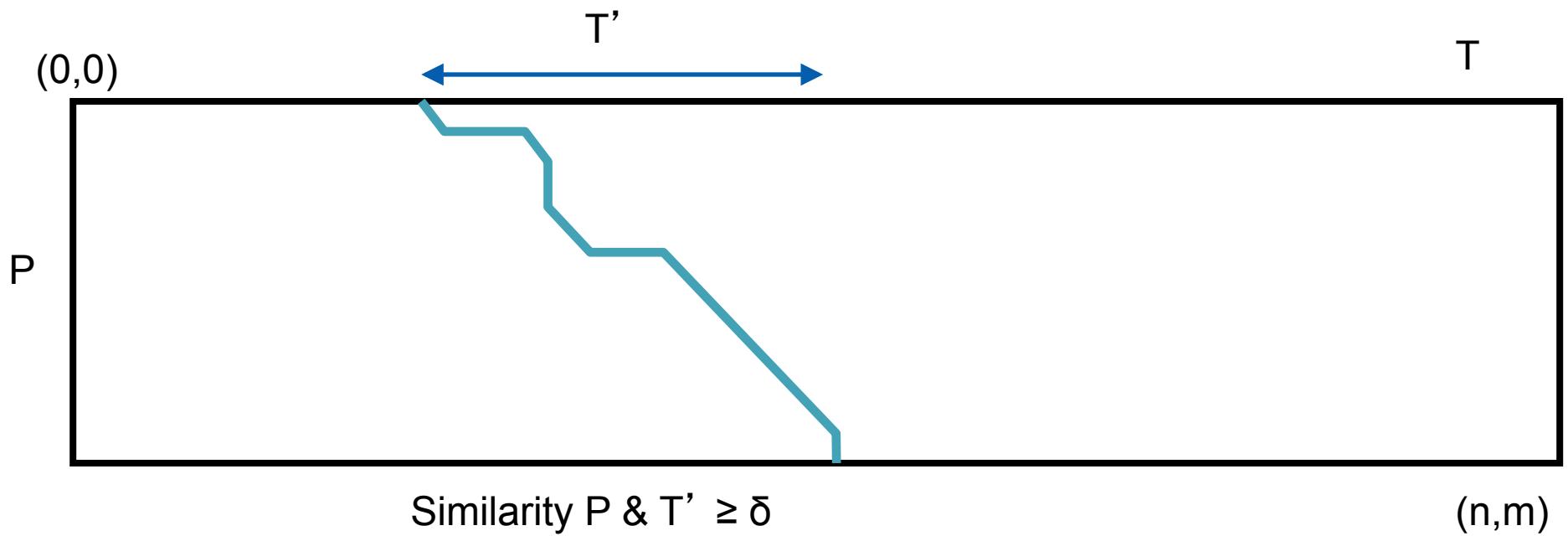
$$s_{i,j} = \max \begin{cases} 0 & \xleftarrow{\hspace{1cm}} \\ s_{i-1,j-1} + \delta(v_i, w_j) \\ s_{i-1,j} + \delta(v_i, -) \\ s_{i,j-1} + \delta(-, w_j) \end{cases}$$

Power of ZERO: there is only this change from the original recurrence of a Global Alignment - since there is only one “free ride” edge entering into every vertex

Local Alignment Schematic



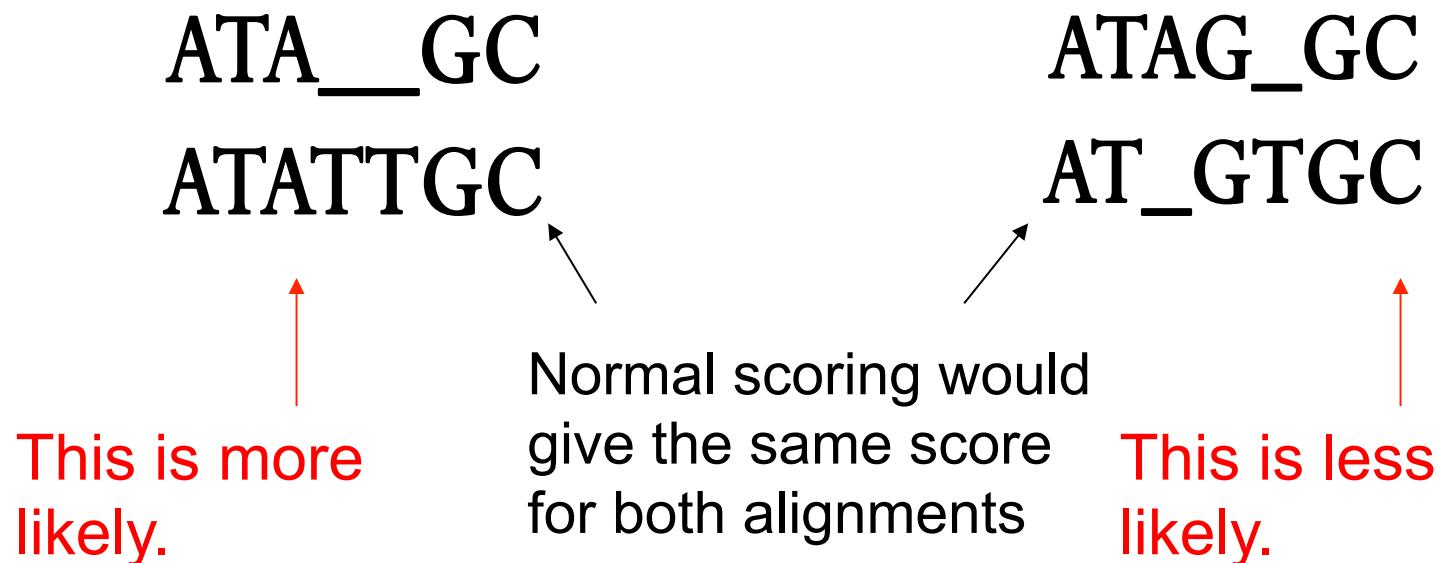
Searching for GATTACA



- Don't “charge” for optimal alignment starting in cells $(0,j)$
 - Baseconds: $D(0,j) = 0, D(i,0) = \sum_{k \leq i} s(S(k), '-')$
- Don't “charge” for ending alignment at end of P (but not necc. T)
 - Find cell (n,j) with edit distance $\leq \delta$

Affine Gap Penalties

- In nature, a series of k indels often come as a single event rather than a series of k single nucleotide events:

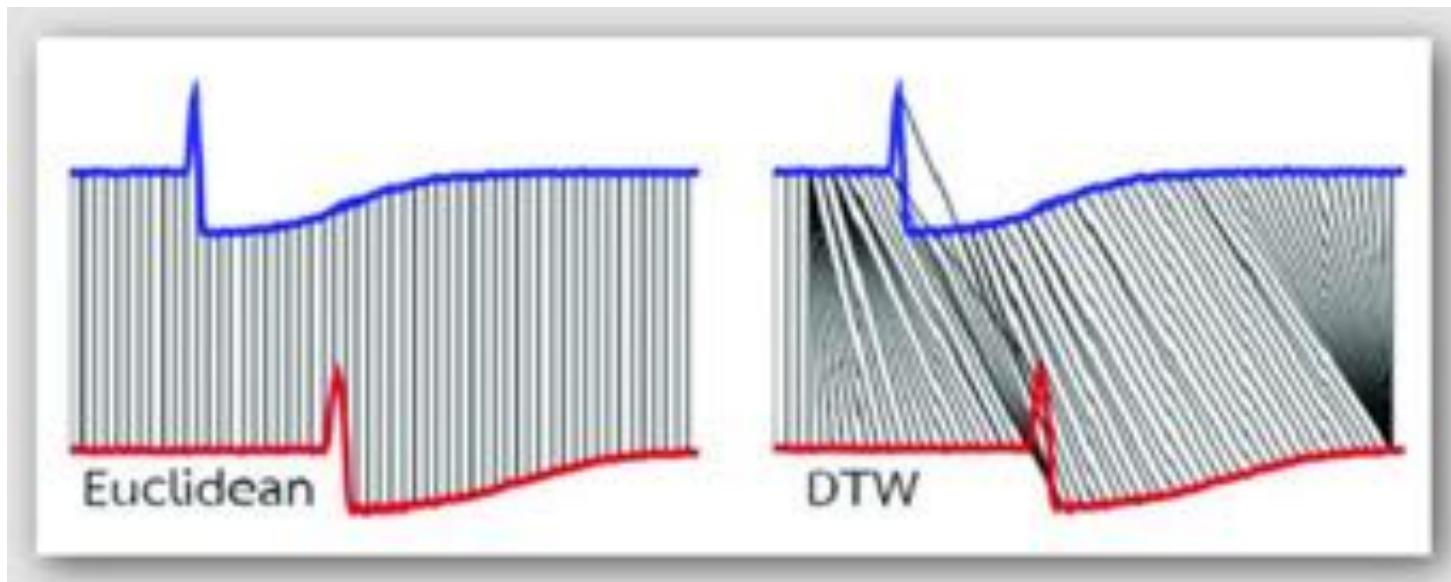


Accounting for Gaps

- *Gaps*- contiguous sequence of spaces in one of the rows
- Score for a gap of length x is: $-(\rho + \sigma x)$
where $\rho > 0$ is the **gap opening penalty**
 ρ will be large relative to **gap extension penalty** σ
 - Gap of length 1: $-(\rho + \sigma) = -6$
 - Gap of length 2: $-(\rho + \sigma 2) = -7$
 - Gap of length 3: $-(\rho + \sigma 3) = -8$
- Smith-Waterman-Gotoh incorporates affine gap penalties without increasing the running time $O(mn)$

Dynamic Time Warp

- Algorithm for measuring the similarity between two sequences of numeric values that vary in time or speed
 - Computes a non-linear mapping for sequence A to sequence B
 - Many applications for video, audio, and graphics
 - Speech processing: Recognize speech patterns coping with different speaking speeds
 - EEG processing: Identify anomalies in brain or heart activity

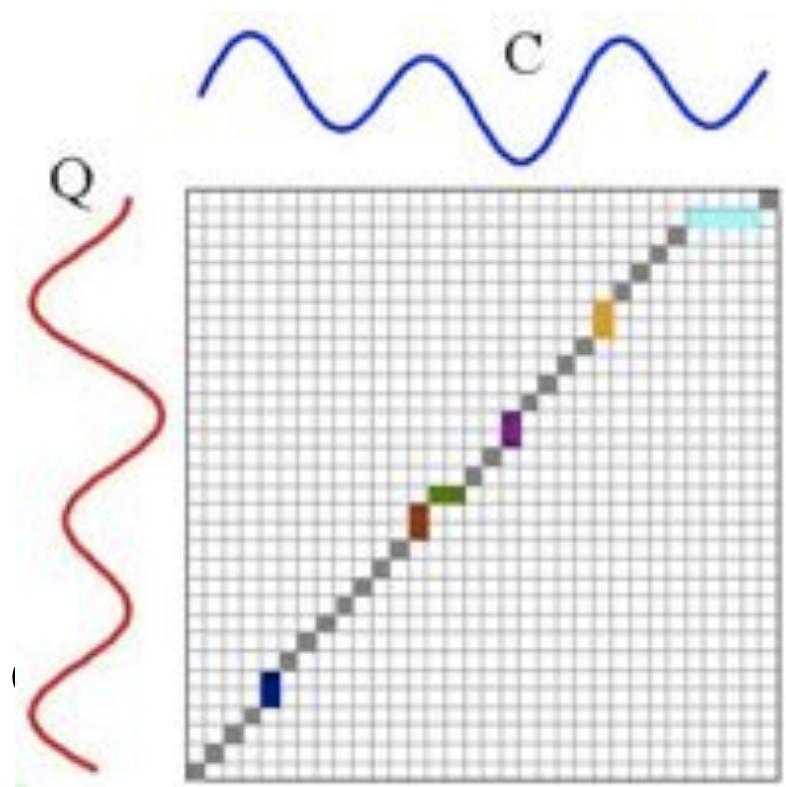
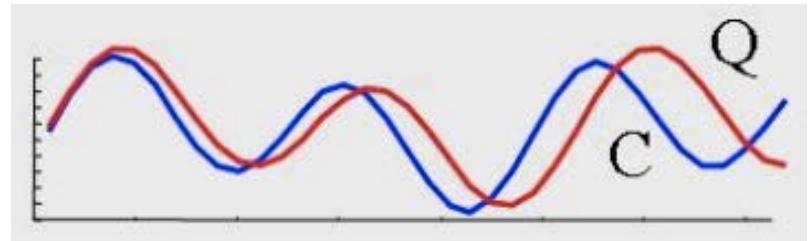


DTW Algorithm

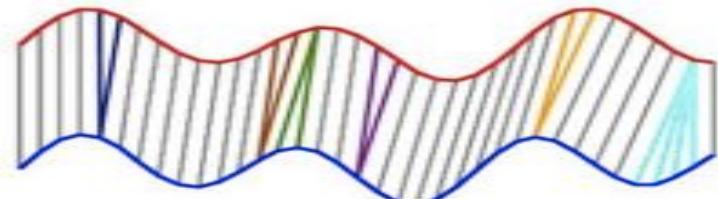
- *DP Algorithm*
 - Input: two time series C and Q
 - Compute the time warping matrix d

$$d(0,0) = 0; d(i,0) = d(0,j) = \infty$$

$$d(i,j) = |c_i - q_j| + \min \begin{cases} d(i-1, j) \\ d(i, j-1) \\ d(i-1, j-1) \end{cases}$$



- Warping matrix projects sequence C to sequence Q, allowing for non-linear contractions and expansions.



Break

Basic Local Alignment Search Tool

- Rapidly compare a sequence Q to a database to find all sequences in the database with a score above some cutoff S .
 - Which protein is most similar to a newly sequenced one?
 - Where does this sequence of DNA originate?
- Speed achieved by using a procedure that typically finds “most” matches with scores $> S$.
 - Tradeoff between sensitivity and specificity/speed
 - Sensitivity – ability to find all related sequences
 - Specificity – ability to reject unrelated sequences

(Altschul et al. 1990)

Seed and Extend

FAKDFLAGGVAAAISKTAVAPIERVKLLLQVQHASKQITADKQYKGIIDCVVRIPKEQGV
F D +GG AAA+SKTAVAPIERVKLLLQVQ ASK I DK+YKGI+D ++R+PKEQGV
FLIDLASGGTAAAVSKTAVAPIERVKLLLQVQDASKAIAVDKRYKGIMDVLRVPKEQGV

- Homologous sequence are likely to contain a **short high scoring word pair**, a seed.
 - Unlike *Baeza-Yates*, BLAST *doesn't* make explicit guarantees
- BLAST then tries to extend high scoring word pairs to compute **maximal high scoring segment pairs (HSPs)**.
 - Heuristic algorithm but evaluates the result statistically.

Sequence Similarity

- **Similarity score generalizes edit distance**
 - Certain mutations are much more likely than others
 - Hydrophilic -> Hydrophilic much more likely than Hydrophilic -> Hydrophobic
 - BLOSUM62
 - Empirically measure substitution rates among proteins that are 62% identical
 - Positive score: more likely than chance, Negative score: less likely

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
Ala	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0
Arg	-1	5	0	-2	-3	1	-4	-2	-3	-1	-2	-2	-1	-3	-1	-1	-1	-4	-2	-1
Asn	-2	0	6	0	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
Asp	-2	-2	1	6	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
Cys	0	-3	-3	-3	9	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
Gln	-1	1	0	0	-3	5	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
Glu	-1	0	0	2	-4	2	5	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2
Gly	0	-2	0	-1	-3	-2	-2	6	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2
His	-2	0	1	-1	-3	0	0	-2	8	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2
Ile	-1	-3	-3	-3	-1	-3	-3	-3	4	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
Leu	-1	-2	-3	-4	-1	-2	-3	-3	-2	4	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
Lys	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-1	-1	-1	-1	-1	-1	-1
Met	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	6	-1	-1	-1	-1	-1	-1	-1
Phe	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-1	-1	-1	-1	-1	-1
Pro	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-1	-2	-4	7	-1	-1	-1	-1
Ser	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	-1	4	-1	-1	-1
Thr	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	-1	1	6	-1	-1
Trp	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	-2	-1
Tyr	-2	-2	-2	-3	-2	-1	-2	-3	-2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1
Val	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

BLAST - Algorithm -

- Step 1: Preprocess Query
Compile **the short-high scoring word list** from query.
The length of query word, w, is 3 for protein scoring
Threshold T is 13

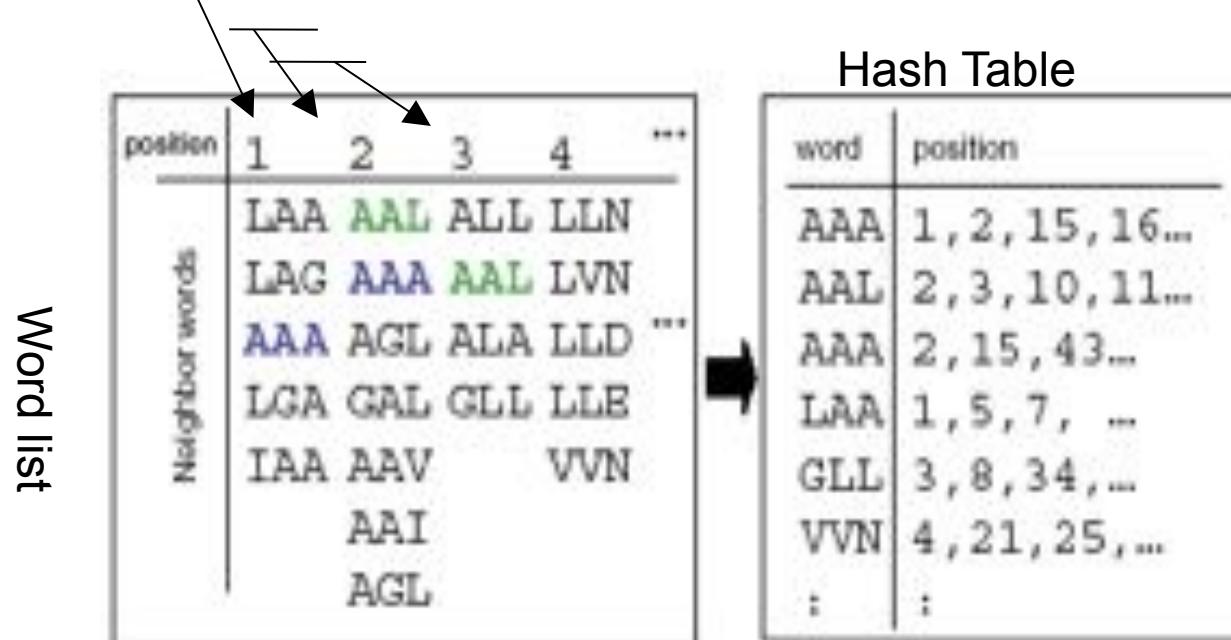
Query : LAALLNKCKT	PQG	ORLNVNQWIKQPLMDKIRRIEE
Query word (W=3)	PQG	18
	PEG	15
	PRC	14
	PKG	14
	PNG	13
neighborhood words	PDG	13
	PHG	13
	PMG	13
	PSG	13
	PQA	12
	PQN	12

neighborhood score
Threshold (T=13)

BLAST - Algorithm -

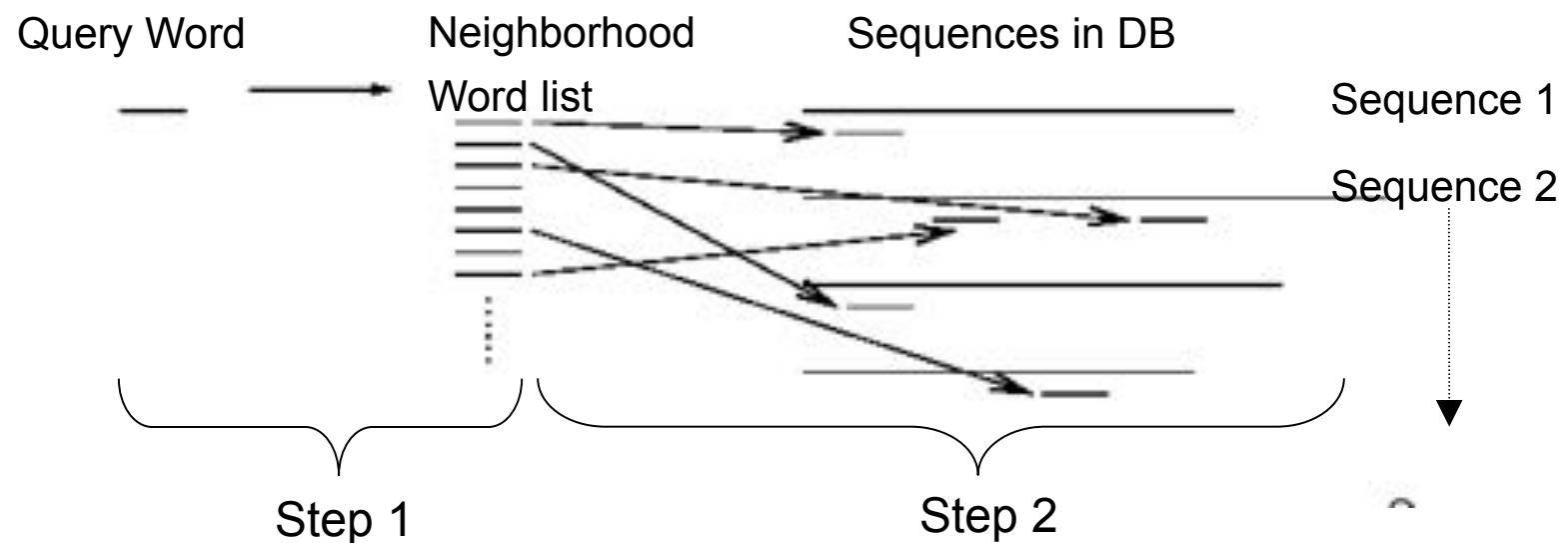
- Step 2: Construct Query Word Hash Table

Query: LAALLNKCKTPQGQQLVNQWIKQPLMD



BLAST - Algorithm -

- Step 3: Scanning DB
Identify all exact matches with DB sequences



BLAST - Algorithm -

- Step 4 (Search optimal alignment)
For each hit-word, extend ungapped alignments in both directions.
Let S be a score of hit-word
- Step 5 (Evaluate the alignment statistically)
Stop extension when **E-value** (depending on score S) become less than threshold. The extended match is called High Scoring Segment Pair.

E-value = the number of HSPs having score S (or higher) expected to occur **by chance**.

→ Smaller E-value, more significant in statistics

Bigger E-value , by chance

$$E[\# \text{ occurrences of a string of length } m \text{ in reference of length } L] \sim L/4^m$$

BLAST E-values

The expected number of HSPs with the score at least S is :

$$E = K * n * m * e^{-\lambda S}$$

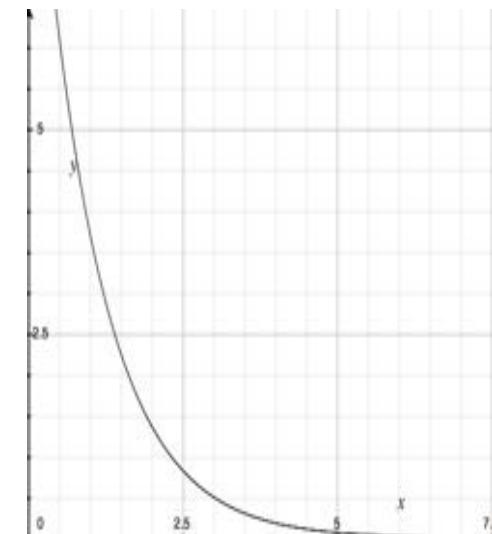
K, λ are constant depending on model

n, m are the length of query and sequence

The probability of finding at least one such HSP is:

$$P = 1 - e^E$$

→ If a word is hit by chance (E-value is bigger),
P become smaller.



The distribution of Smith-Waterman local alignment scores between two random sequences follows the Gumbel extreme value distribution

Parameters

- Larger values of w increases the number of neighborhood words, but decreases the number of chance matches in the database.
 - Increasing w decreases sensitivity.
- Larger values of T decrease the overall execution time, but increase the chance of missing a MSP having score $\geq S$.
 - Increases T decreases the sensitivity
- Larger values of S increase the specificity. The value of S is affected by changes in the expectation value parameter.

Very Similar Sequences

Query: HBA_HUMAN Hemoglobin alpha subunit

Sbjct: HBB_HUMAN Hemoglobin beta subunit

Score = 114 bits (285), Expect = 1e-26

Identities = 61/145 (42%), Positives = 86/145 (59%), Gaps = 8/145 (5%)

Query 2 LSPADKTNVKAAGKVGAAHAGEYGAEALERMFLSFPTTKTYFPHF-----DLSHGSAQV 55
L+P +K+ V A WGKV + E G EAL R+ + +P T+ +F F D G+ +V

Sbjct 3 LTPEEKSAVTALWGKV--NVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAMGNPKV 60

Query 56 KGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKLRDPVNFKLLSHCLLVTLAHLPA 115
K HGKKV A ++ +AH+D++ + LS+LH KL VDP NF+LL + L+ LA H

Sbjct 61 KAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLGNVLVCVLAHHFGK 120

Query 116 EFTP AVHASLDKFLASVSTVLTSKY 140
EFTP V A+ K +A V+ L KY

Sbjct 121 EFTPPVQAAYQKV VAGVANALAHKY 145

Quite Similar Sequences

Query: HBA_HUMAN Hemoglobin alpha subunit

Sbjct: MYG_HUMAN Myoglobin

Score = 51.2 bits (121), Expect = 1e-07,

Identities = 38/146 (26%), Positives = 58/146 (39%), Gaps = 6/146 (4%)

Query 2 LSPADKTNVKAAWGKVGAHAGEYGAELERMFLSFPTTKTYFPF-----DLSHGSAQV 55
LS + V WGKV A +G E L R+F P T F F D S +

Sbjct 3 LSDGEWQLVLNVWGKVEADIPGHGQEVLIRLFKGHPETLEKFDKFHKHLKSEDEMKAEDL 62

Query 56 KGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKLRDPVNFKLLSHCLLVTAAHLPA 115
K HG V AL + + L+ HA K ++ + +S C++ L + P

Sbjct 63 KKHGATVLTALGGILKKKGHHEAEIKPLAQSHATKHKIPVKYLEFISECIIQVLOSKHPG 122

Query 116 EFTPAVHASLDKFLASVSTVLTSKYR 141
+F +++K L + S Y+

Sbjct 123 DFGADAQGAMNKALELFRKDMASNYK 148

Not similar sequences

Query: HBA_HUMAN Hemoglobin alpha subunit

Sbjct: SPAC869.02c [Schizosaccharomyces pombe]

Score = 33.1 bits (74), Expect = 0.24

Identities = 27/95 (28%), Positives = 50/95 (52%), Gaps = 10/95 (10%)

Query 30 ERMFLSFPTTKTYFPHFDSLHGSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAH 89
++M ++P P+F+ +H + +A AL N ++DD+ +LSA D

Sbjct 59 QKMLGNYPEV---LPYFNKAHQISL--SQPRILAFALLNYAKNIDDL-TSLSAFMDQIVV 112

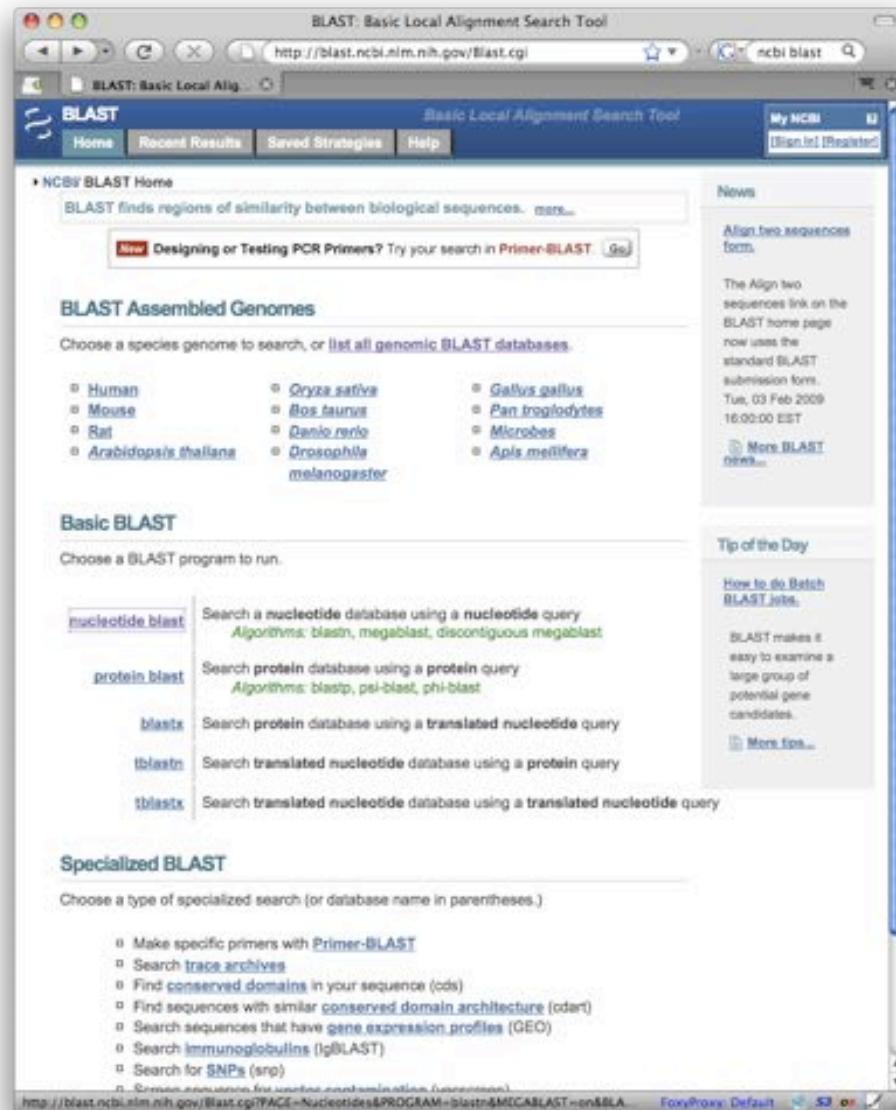
Query 90 K---LRVDPVNFKLLSHCLLVTAAHLPAEF-TPA 120
K L++ ++ ++ HCLL T+ LP++ TPA

Sbjct 113 KHVGQLQIKAEHYPIVGHCLLSTMQELLPSDVATPA 147

Blast Versions

Program	Database	Query
BLASTN	Nucleotide	Nucleotide
BLASTP	Protein	Protein
BLASTX	Protein	Nucleotide translated in to protein
TBLASTN	Nucleotide translated in to protein	Protein
TBLASTX	Nucleotide translated in to protein	Nucleotide translated in to protein

NCBI Blast



- Nucleotide Databases
 - nr:All Genbank
 - refseq: Reference organisms
 - wgs:All reads

- Protein Databases
 - nr:All non-redundant sequences
 - Refseq: Reference proteins

BLAST Exercise

>whoami

```
TTGATGCAGGTATCTGCGACTGAGACAATATGCA
ACAGTTGAATGAATCATAATGGAATGTGCACTCT
AACCAGCCAATTGATGCTGGCTGCAGAGATGC
AAGATCAAGAGGTGACACCTGCTCTGAAGAAAAG
CACAGTTGAACTGCTGGATCTGCAAACATACAGCA
GGCACTCCAGGCACCAAGACAACATCTTTACA
CCAGCAAACATGTGGATTGATATCTCCTAACAGC
AGTGATTAACAGAGACGACTGCAGGATTGCTTC
CACAAACAAAAAT
```

Sequence Alignment Summary

- Distance metrics:
 - Hamming: How many substitutions?
 - Edit Distance: How many substitutions or indels?
 - Sequence Similarity: How similar (under this model of similarity)?
- Techniques
 - Seed-and-extend: Anchor the search for in-exact using exact only
 - Dynamic Programming: Find a global optimal as a function of its parts
 - BWT Search: implicit DFS of SA/ST
- Sequence Alignment Algorithms: Pick the right tool for the job
 - Smith-Waterman: DP Local sequence alignment
 - BLAST: Homology Searching
 - MUMmer: Whole genome alignment
 - Bowtie/BWA/Novoalign: short read mapping

Thank You!

<http://schatzlab.cshl.edu>

@mike_schatz



Whole Genome Alignment with MUMmer

Slides Courtesy of Adam M. Phillippy

amp@umics.umd.edu

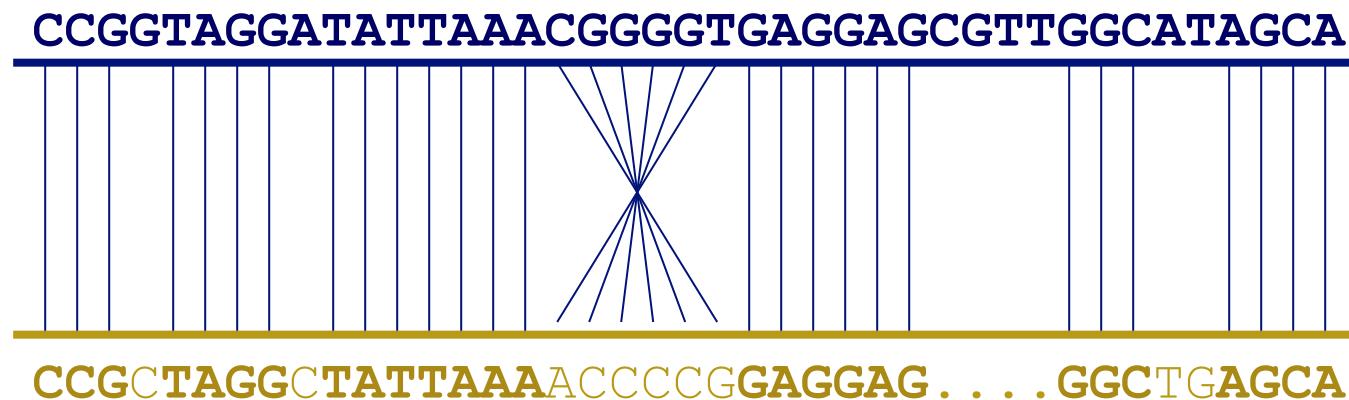
Goal of WGA

- For two genomes, A and B , find a mapping from each position in A to its corresponding position in B



Not so fast...

- Genome A may have insertions, deletions, translocations, inversions, duplications or SNPs with respect to B (sometimes all of the above)



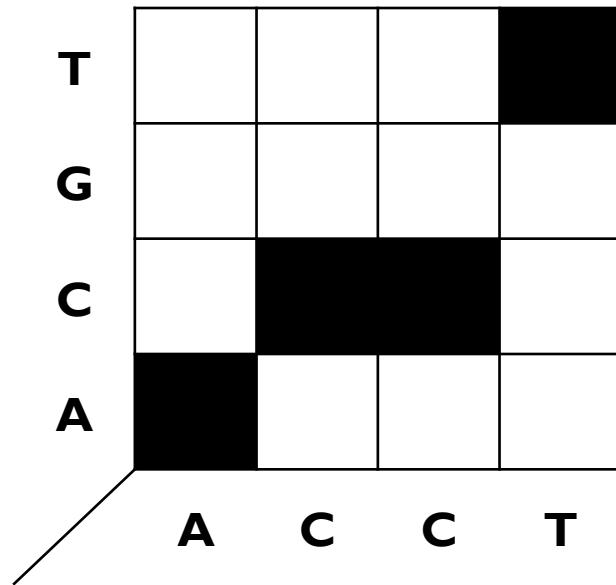
WGA visualization

- How can we visualize *whole genome* alignments?

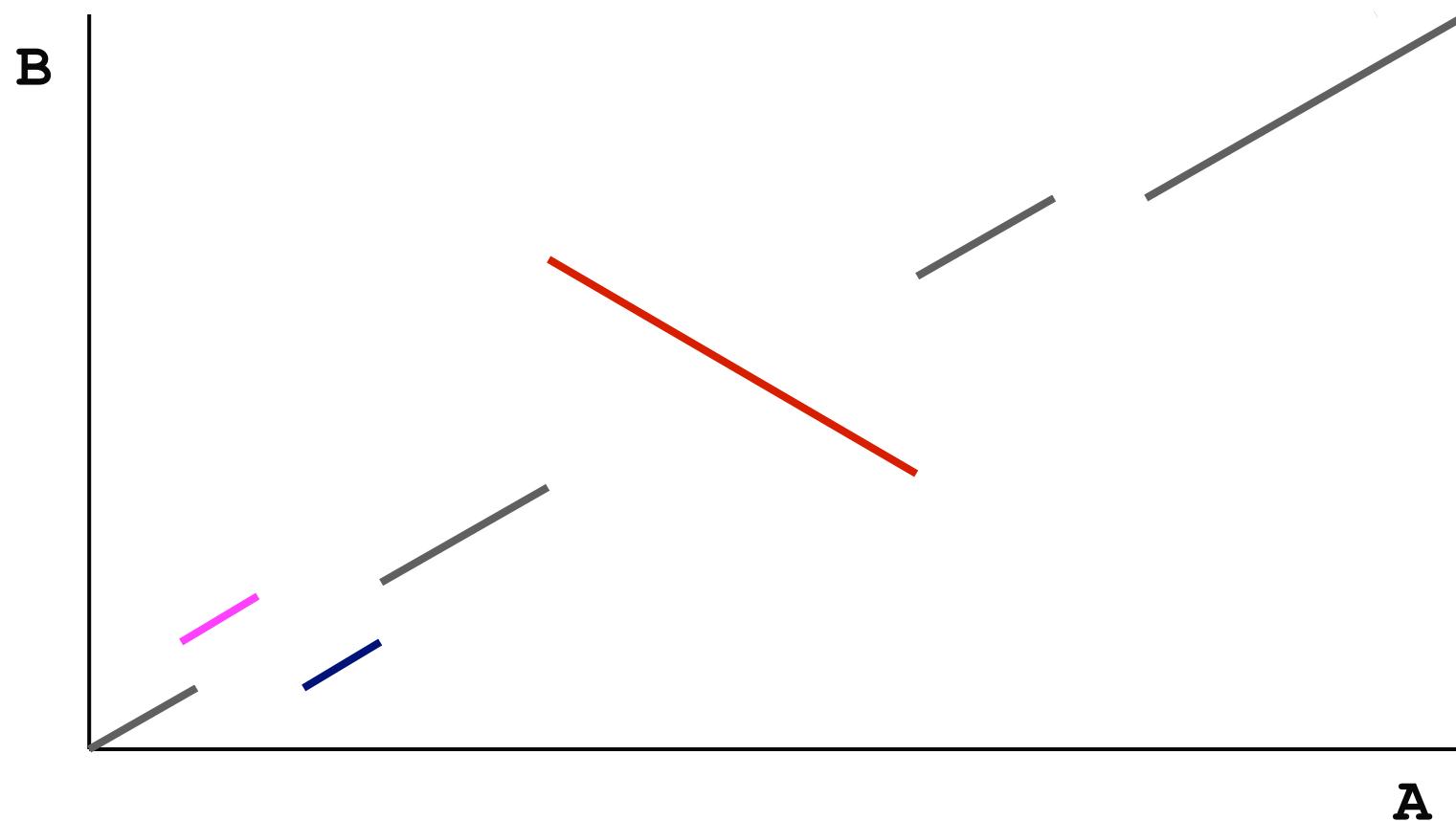
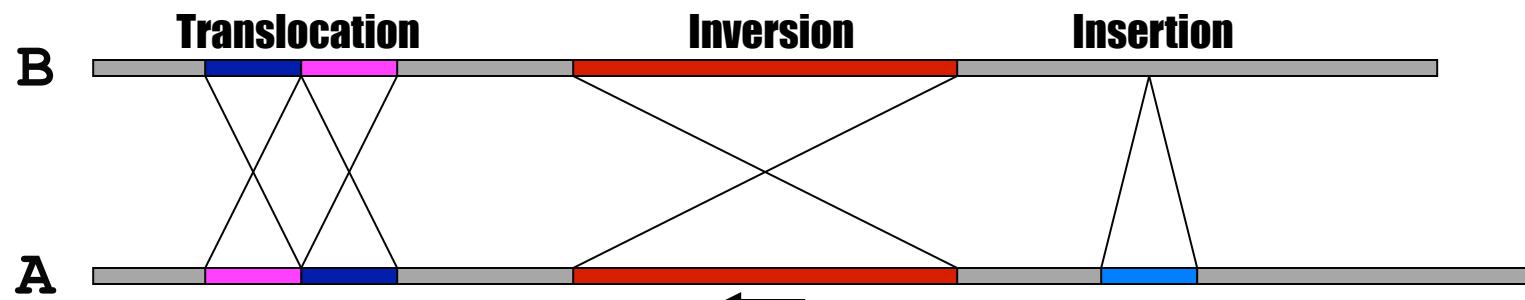
- With an alignment dot plot

- $N \times M$ matrix

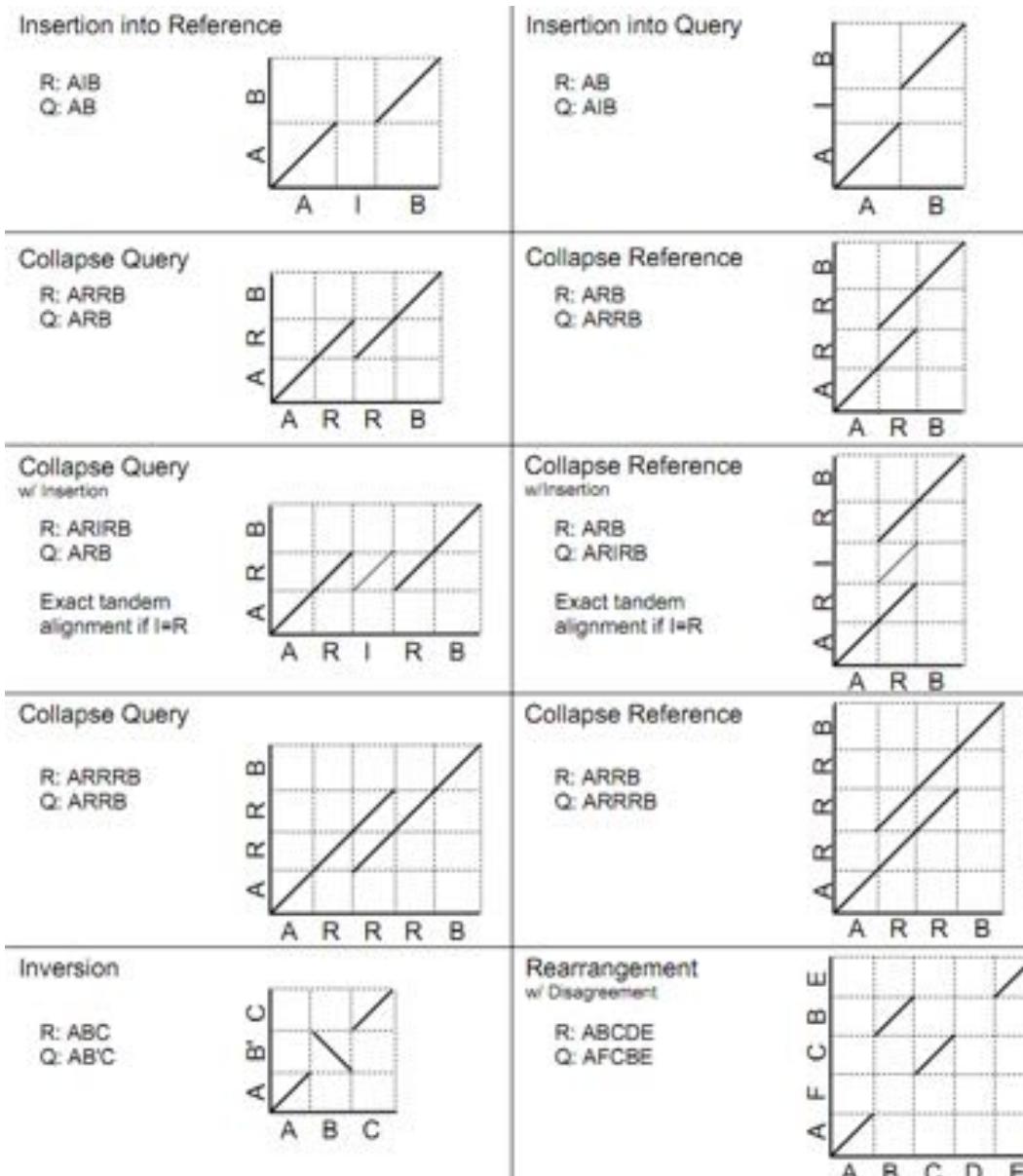
- Let i = position in genome A
 - Let j = position in genome B
 - Fill cell (i,j) if A_i shows similarity to B_j



- A perfect alignment between A and B would completely fill the positive diagonal



SV Types



- Different structural variation types / misassemblies will be apparent by their pattern of breakpoints
- Most breakpoints will be at or near repeats
- Things quickly get complicated in real genomes

[http://mummer.sf.net/manual/
AlignmentTypes.pdf](http://mummer.sf.net/manual/AlignmentTypes.pdf)

Seed-and-extend with MUMmer

How can quickly align two genomes?

I. Find maximal-unique-matches (MUMs)

- ◆ Match: exact match of a minimum length
- ◆ Maximal: cannot be extended in either direction without a mismatch
- ◆ Unique
 - ◆ occurs only once in both sequences (MUM)
 - ◆ occurs only once in a single sequence (MAM)
 - ◆ occurs one or more times in either sequence (MEM)

2. Cluster MUMs

- ◆ using size, gap and distance parameters

3. Extend clusters

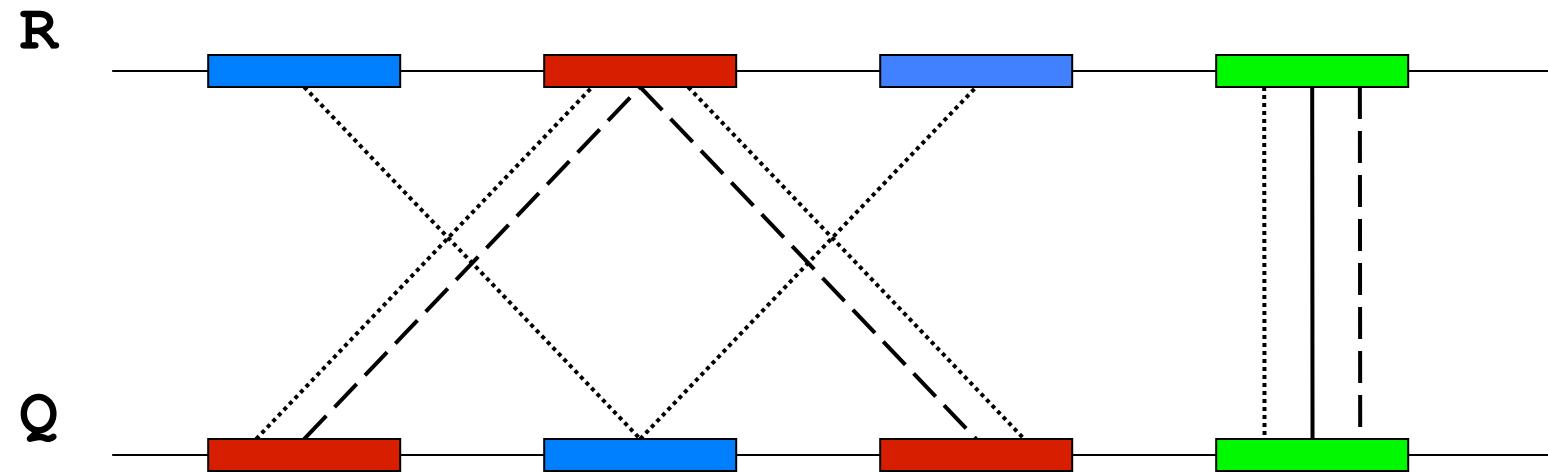
- ◆ using modified Smith-Waterman algorithm

Fee Fi Fo Fum, is it a MAM, MEM or MUM?

MUM : maximal unique match

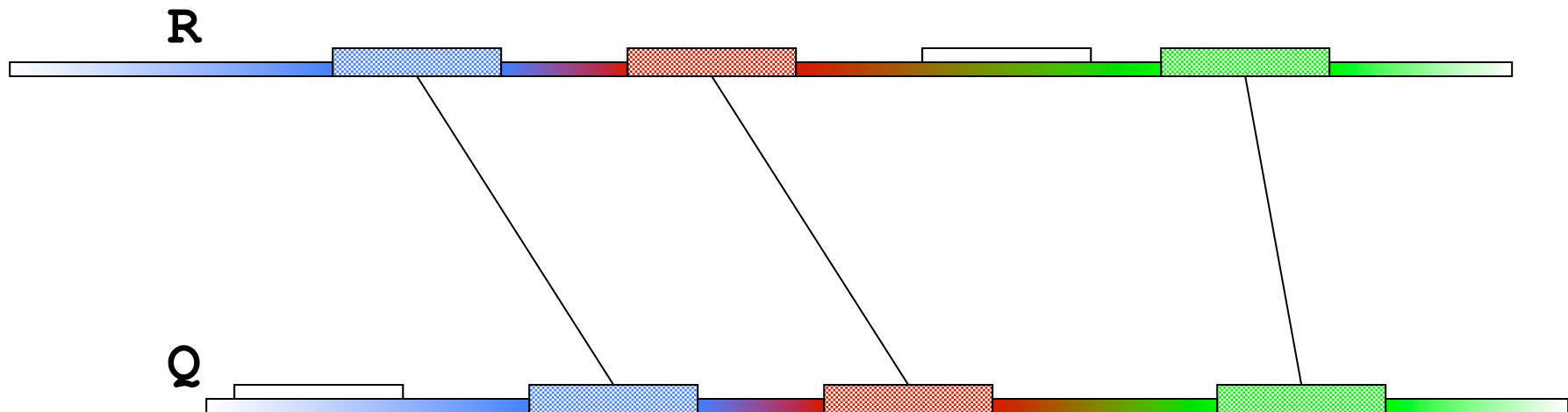
MAM : maximal almost-unique match

MEM : maximal exact match



Seed and Extend visualization

FIND all MUMs
CLUSTER consistent MUMs
EXTEND alignments



WGA example with nucmer

- *Yersina pestis* CO92 vs. *Yersina pestis* KIM
 - High nucleotide similarity, 99.86%, but extensive reshuffling
 - High repeat content

```
nucmer -maxmatch CO92.fasta KIM.fasta
-maxmatch      Find maximal exact matches (MEMs)
```

```
delta-filter -m out.delta > out.filter.m
-m      Many-to-many mapping
```

```
show-coords -r out.delta.m > out.coords
-r      Sort alignments by reference position
```

```
dnadiff out.delta.m
Construct catalog of sequence variations
```

```
mummerplot --large --layout out.delta.m
--large      Large plot
--layout    Nice layout for multi-fasta files
```

