02-716

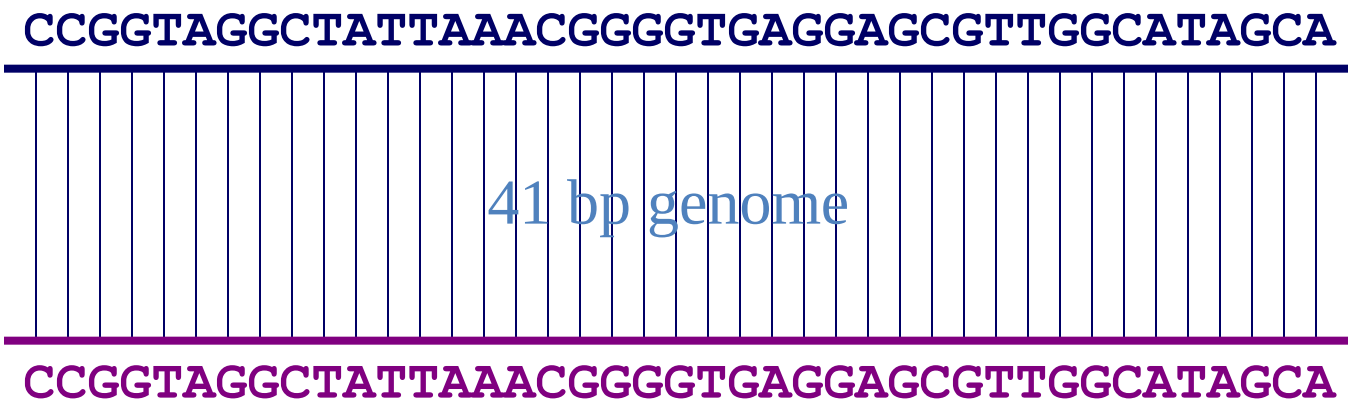
Cross species analysis of genomics data

Whole Genome Alignment

Goal of WGA

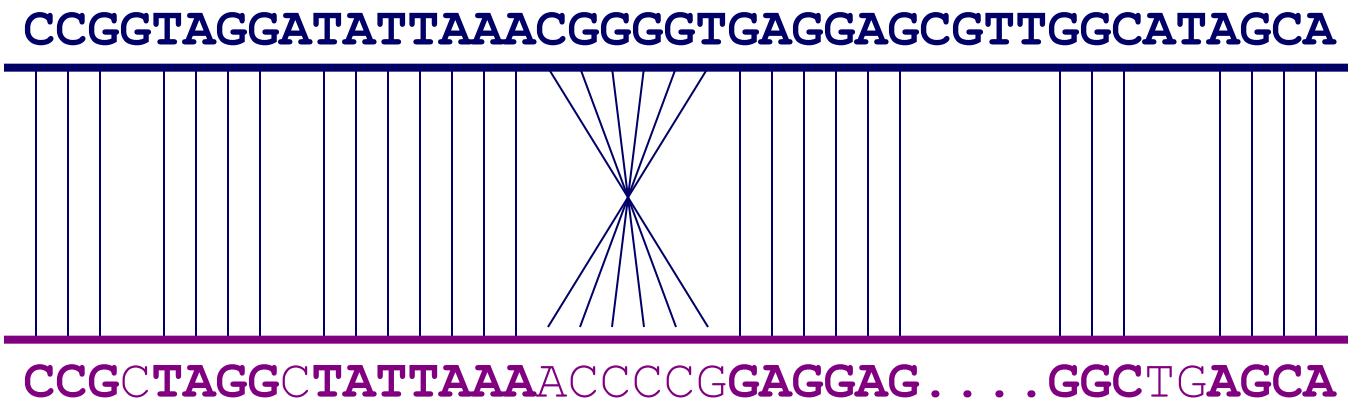
- For two genomes, *A* and *B*, find a mapping from each position in *A* to its corresponding position in *B*

What can change between the genomes?



Not so fast...

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

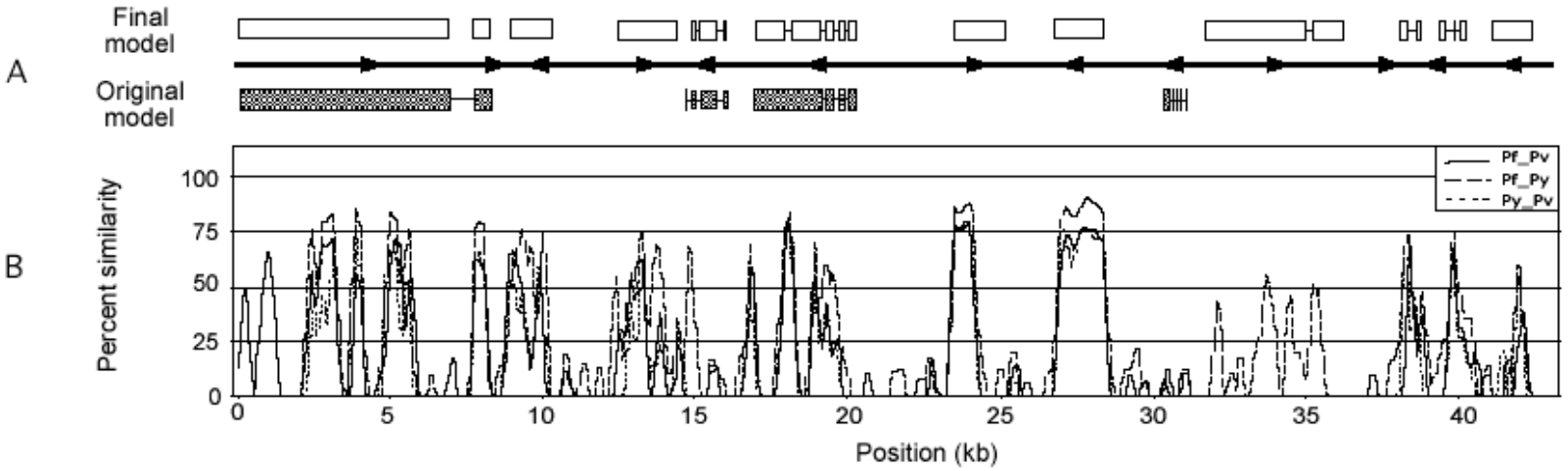


Sidetrack: Plots

- How can we visualize alignments?

- With an identity plot
  - XY plot
    - Let x = position in genome *A*
    - Let y = %similarity of A_x to corresponding position in *B*
  - Plot the identity function
  - This can reveal islands of conservation, e.g. exons

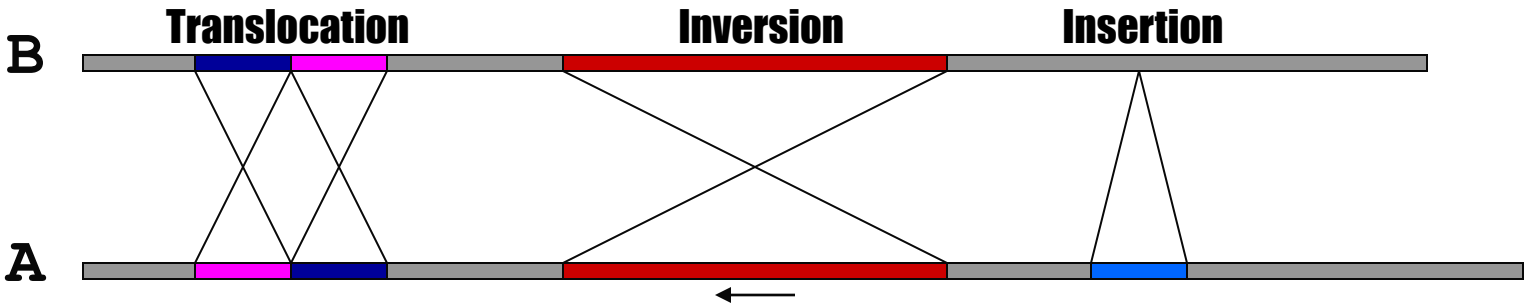
Identity plot example

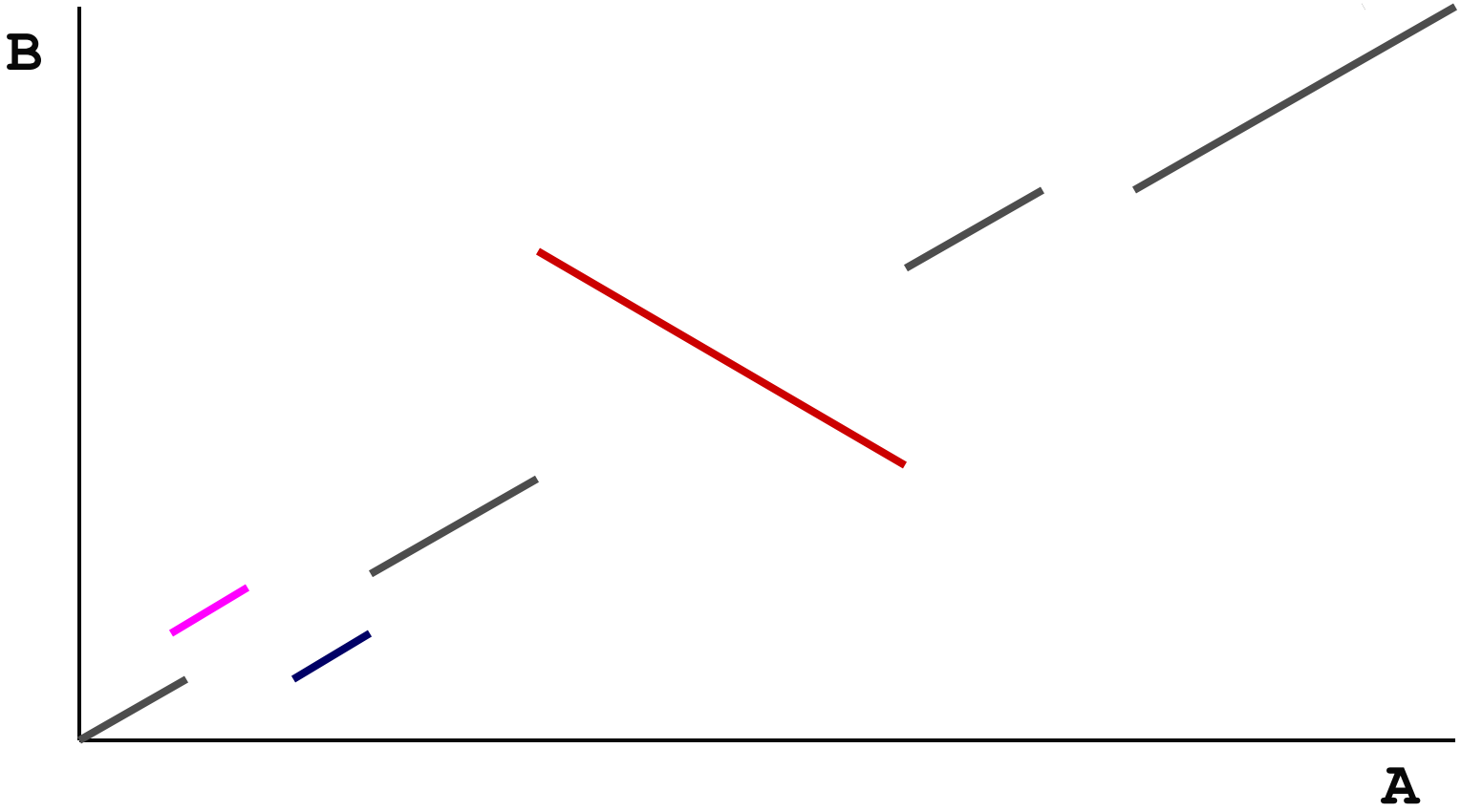


Sidetrack: Plots

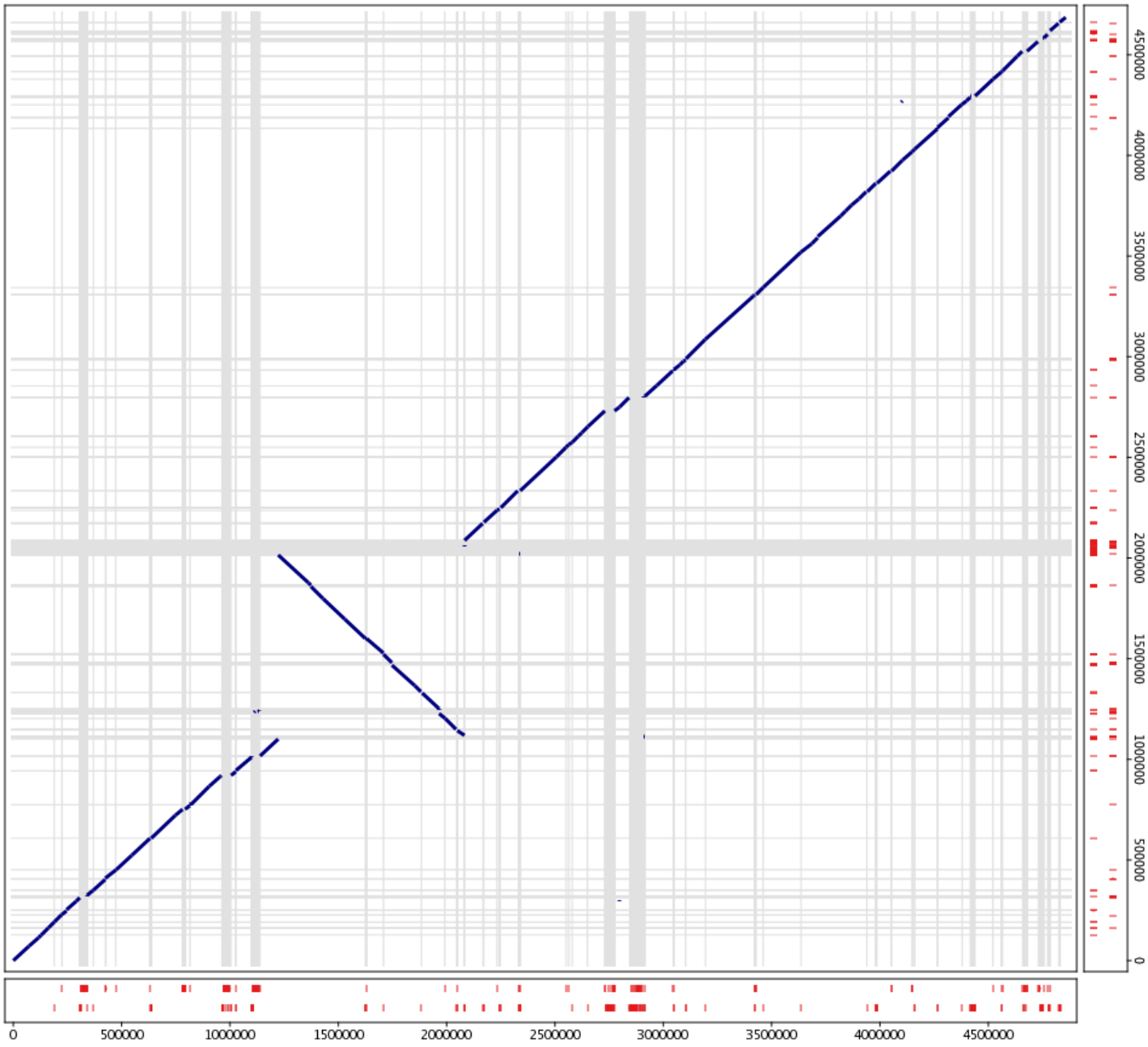
- How can we visualize *whole* genome alignments?

- With an alignment dot plot
  - *N* x *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

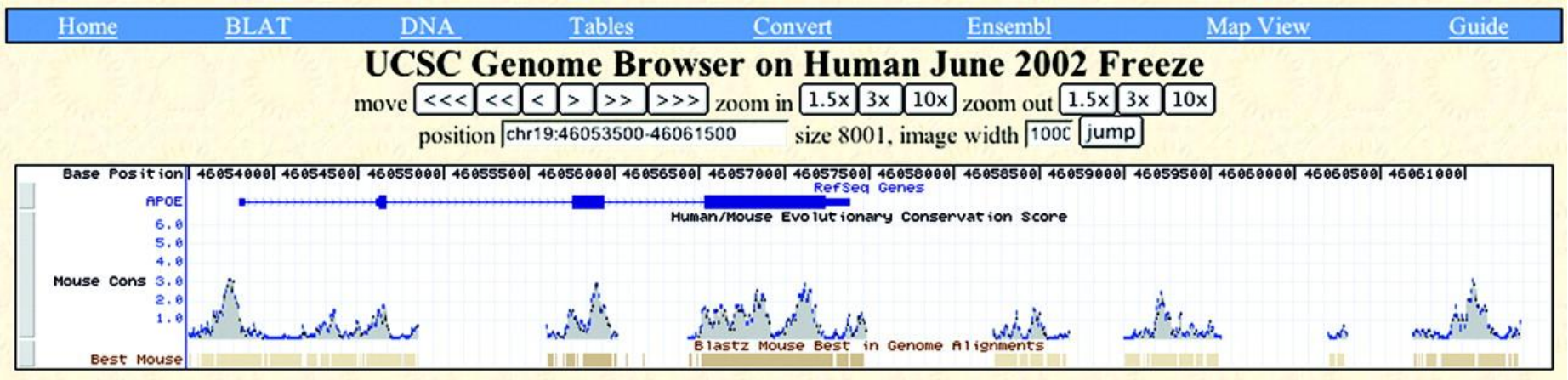




http://mummer.sourceforge.net/manual/AlignmentTypes.pdf


0
500000
1000000
1500000
2000000
2500000
3000000
3500000
4000000
4500000

UCSC genome browser

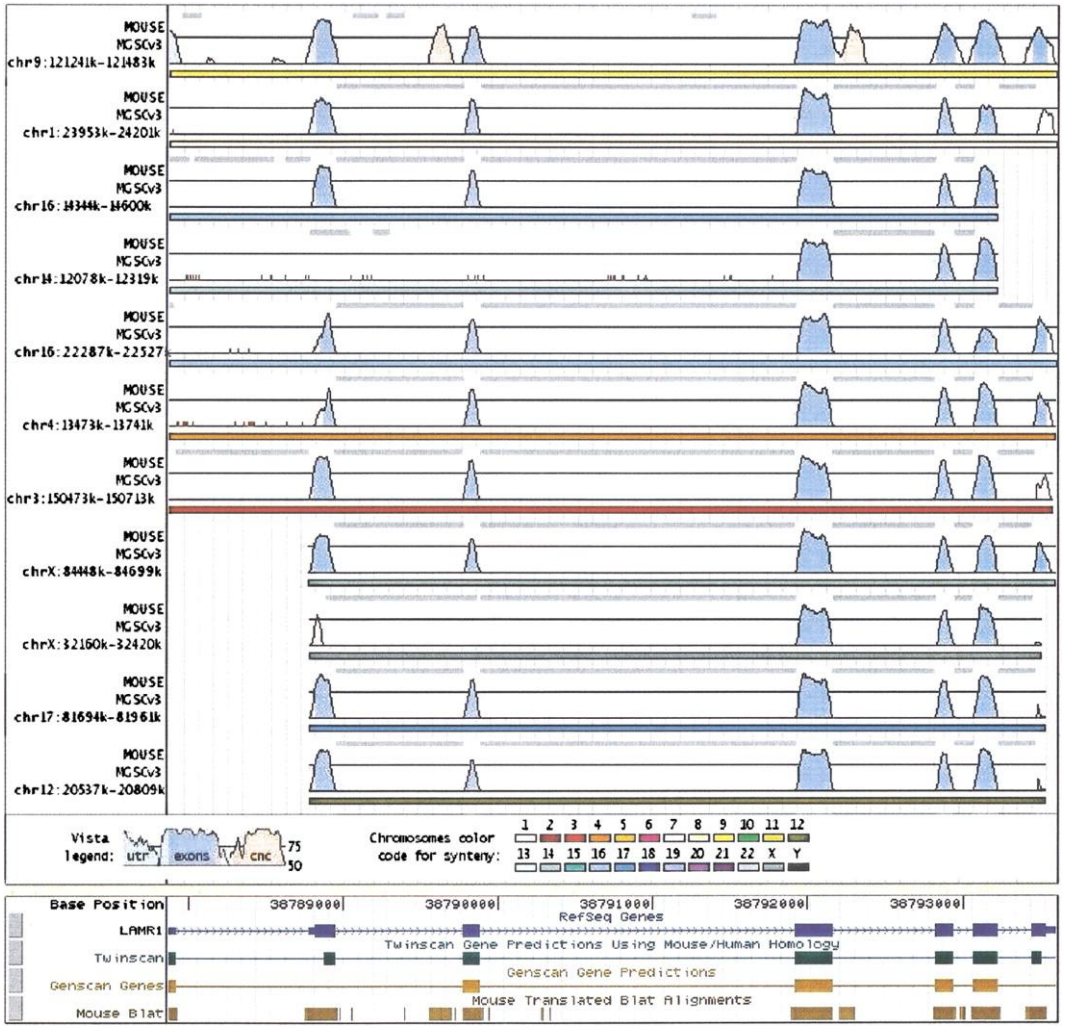


UCSC Genome Browser output for human/mouse sequence comparison of the *ApoE* gene (22). Human sequence is depicted on the *x* axis. Note the different scoring system in contrast to percent identity, with peaks representing *L-scores* that take into account the context of the level of conservation. Conservation in *relatively nonconserved regions* receives higher L-scores than similar conservation in relatively highly conserved regions. As a second display of conservation, the "best mouse" track uses blocks whose length and shading represent the conservation.

Pennachio, Rubin, J Clin Invest. 111, 1099 (2003)

local alignment: problem with repeats

About 2% of the aligned base pairs of the human genome are covered by more than one mouse sequence fragment.
In the example shown, several copies of the mouse pseudogene of Laminin B receptor (LAMR1) from different chromosomes were aligned to human LAMR1 (4th line from bottom).



Couronne, ..., Dubchak, Genome Res. 13, 73 (2003)

sensitivity

In the global human:mouse alignment more than one million regions are conserved at higher than 70% conservation at the 100-bp level – these regions cover > 200 Million bp.

Only 62% of them are covered by base pairs of a (local) BLAT hit.

This means that 38% of the conserved features are found only at the global alignment stage.

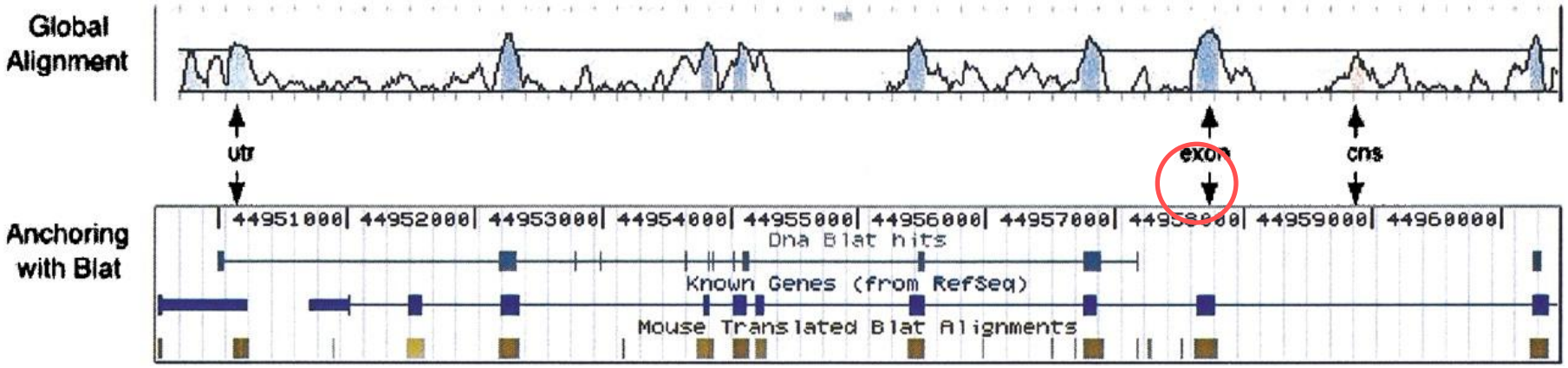
Local alignment can therefore serve as anchoring system for a subsequent global alignment that will identify many additional conserved regions outside the anchors.

Couronne, ..., Dubchak, Genome Res. 13, 73 (2003)

high sensitivity of global alignment

• Global alignment of the mouse finished sequence NT_002570 against the region found by BLAT anchors reveals conserved coding and non-coding elements not found by the BLAT program.

• The anchoring scheme is sensitive enough to provide the global alignment with the correct homology candidate.



Couronne, ..., Dubchak, Genome Res. 13, 73 (2003)

Specificity

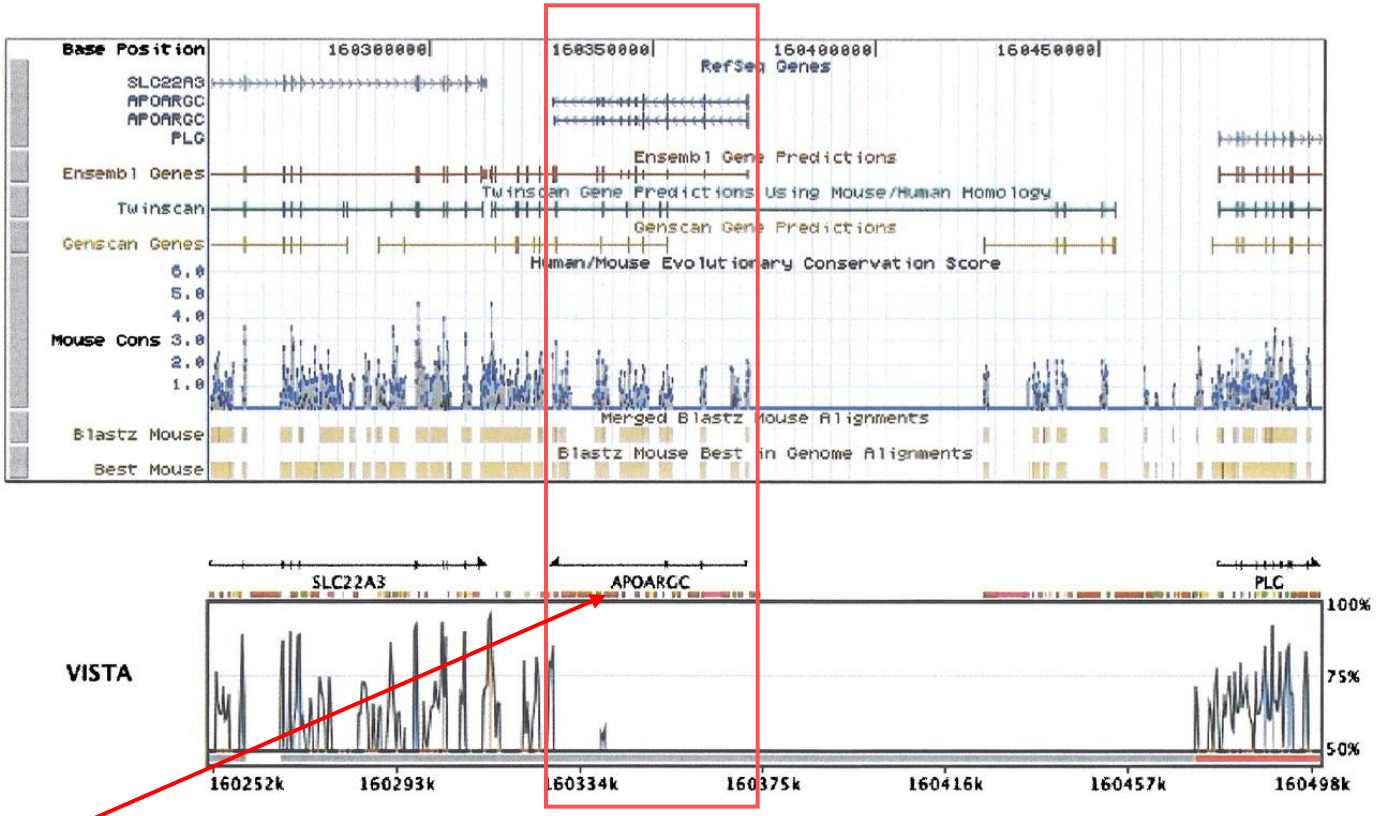
Specificity (how much of the alignment is correct?) is much more difficult to measure than sensitivity.

One test:
Measure coverage of human chromosome 20 by alignments of sequences from all mouse chromosomes except chromosome 2. (HC20 and MC2 are entirely syntenic.) Only 6% coverage is found for exons (mainly becuase of pseudogenes).

Couronne, ..., Dubchak, Genome Res. 13, 73 (2003)

Specificity: Apolipoprotein(a) region

Second test:



The expressed gene apo(a) is only present in a subset of primates - most mammals lack apo(a). Shown are the coverage in this region by the mouse sequence utilizing Blastz and that obtained by Dubchak et al. Only the Dubchak method (see line „VISTA“) predicts that apo(a) has no homology in the mouse, as has been shown experimentally.

Couronne, ..., Dubchak, Genome Res. 13, 73 (2003)

Alignment Tools

- **Whole genome alignment**
  - MUMmer*
    - Developed at TIGR, now supported and available from CBCB at the University of Maryland
  - LAGAN*, AVID
    - VISTA identity plots
- **Multiple genome alignment**
  - MGA, MLAGAN*, DIALIGN, MAVID
- **Multiple alignment**
  - Muscle[?], ClustalW*
- **Local sequence alignment**
  - BLAST*, FASTA,Vmatch

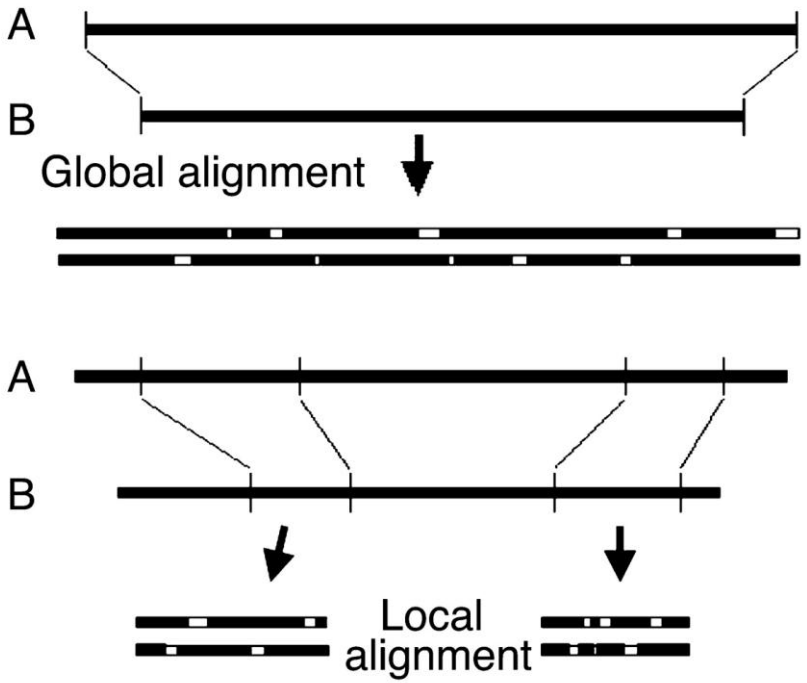
* open source

Why align whole genome sequence? Biological Reasons

Identify differences between organisms that may lead to the understanding of:

- How the two organisms evolved?
  - E.g. How do we differ from chimps?
- Why certain bacteria cause diseases while their cousins do not ? (Indels/ Mutations)
- Why certain people are more susceptible to disease while others are not? (SNPs) – molecular medicine??
- Identifying new drug targets (targets are unique to pathogen)

Local- versus global-alignment algorithm strategies



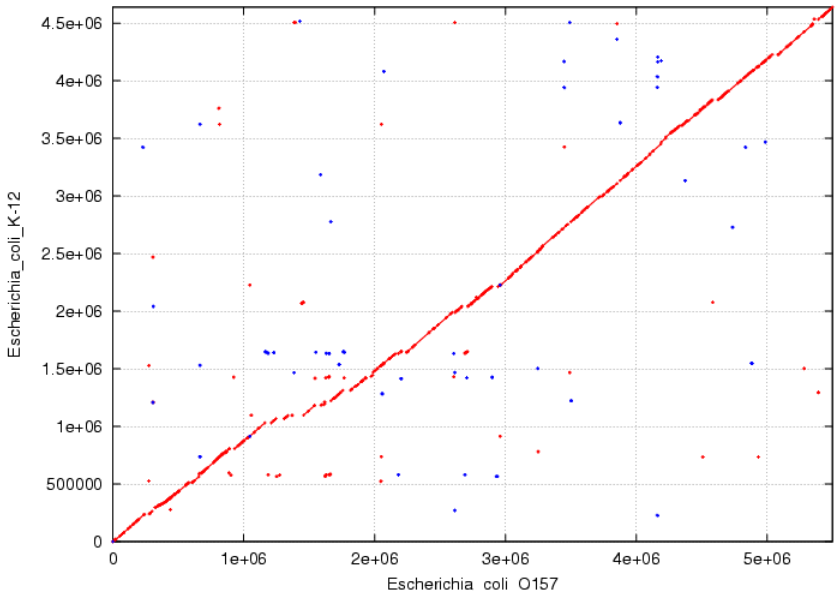
Pennachio, Rubin, J Clin Invest. 111, 1099 (2003)

Top: Global alignments are generated when two DNA sequences (A and B) are compared over the entire length of the two sequences.

Bottom: Local alignments are produced when two DNA sequences (A and B) are compared over numerous subregions along the length of the two sequences.

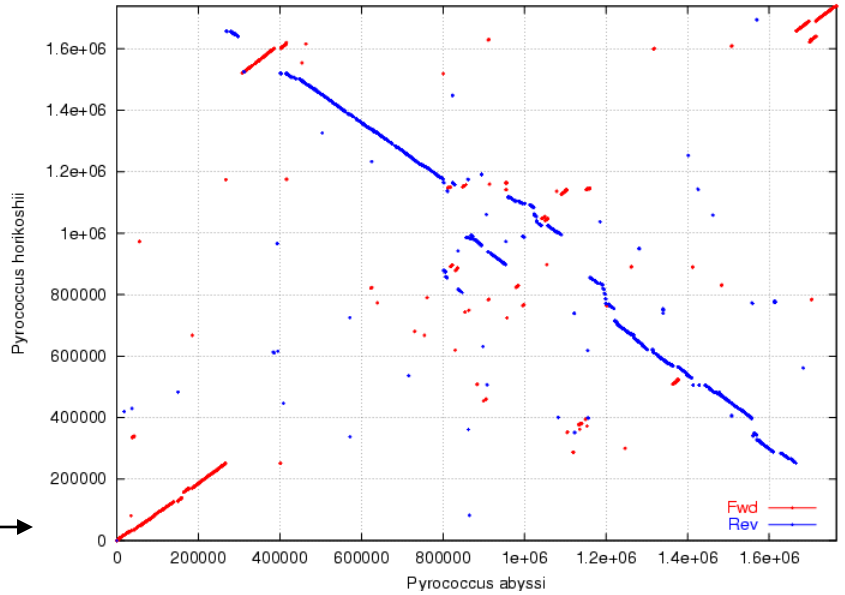
The local-alignment algorithm works by first finding very short common segments between the input sequences (A and B), and then expanding out the matching regions as far as possible.

Global vs. Local



global ok

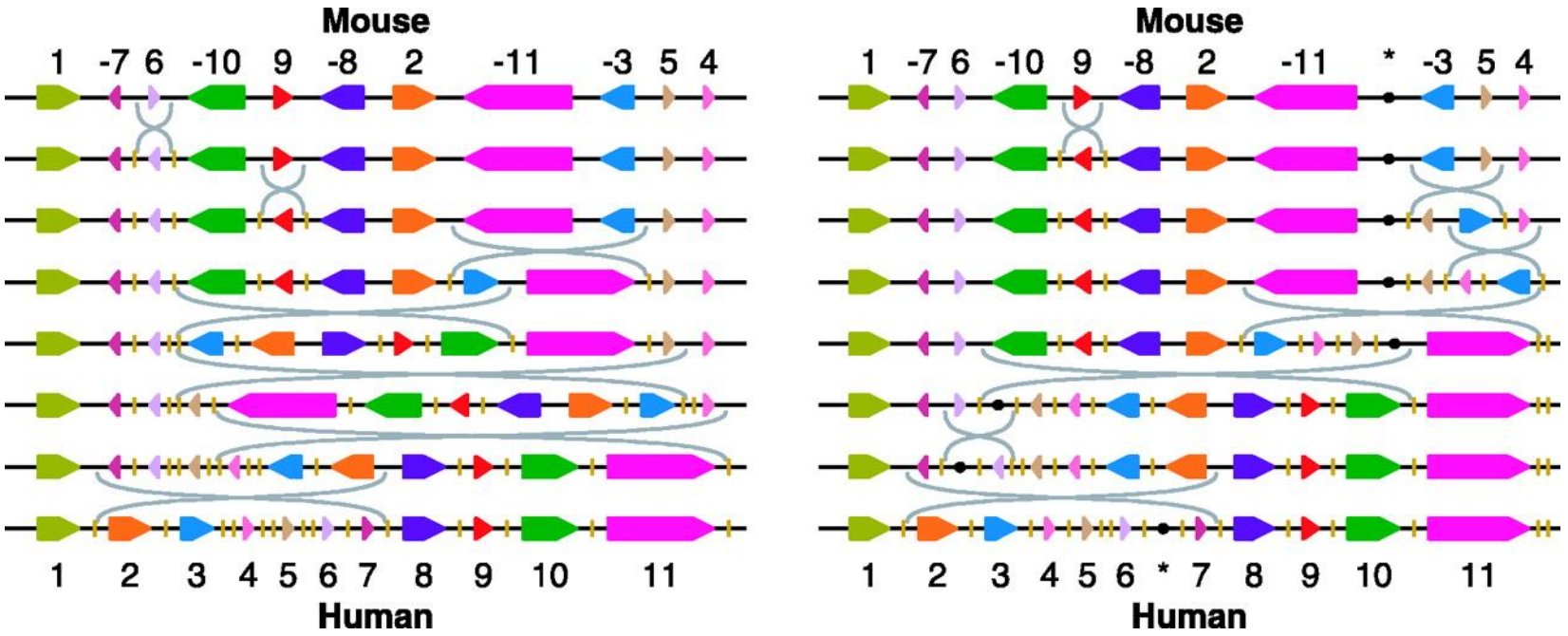
global no way



Additional information by global whole genome alignment

- Difference in repeat patterns
  - Duplication (large fragment, chromosomal)
  - Tandem repeats
- Large insertions and deletions
- translocation (moving from one part of genome to another)
- Single Nucleotide Polymorphism

Two different most parsimonious scenarios that transform the order of the 11 synteny blocks on the mouse X chromosome into the order on the human X chromosome



Pevzner P., Tesler G. PNAS 2003;100:7672-7677

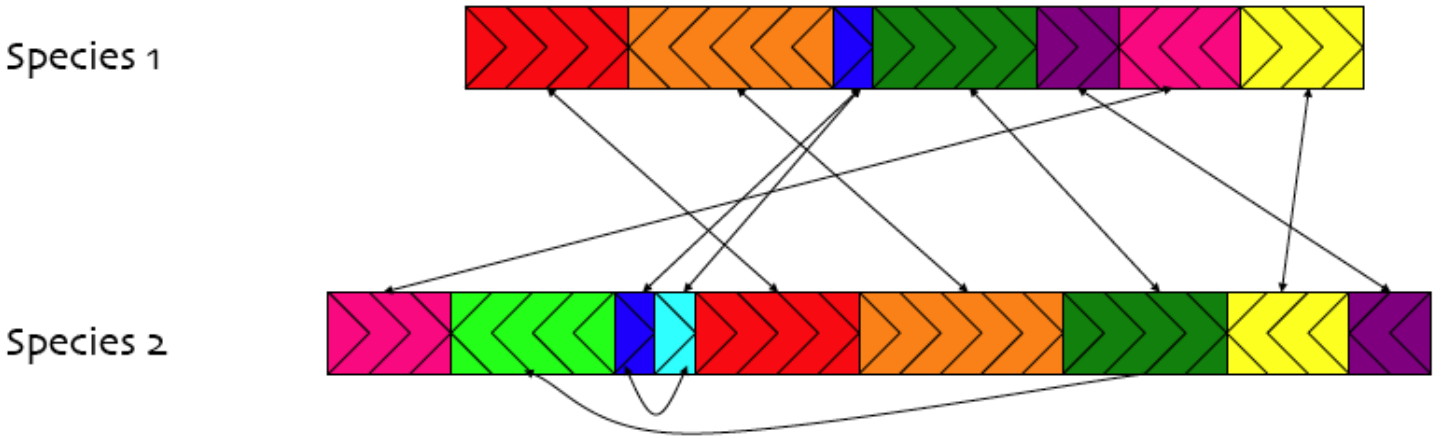
Some approaches for WGA

- MUMmer
- BLASTZ
- Pipmaker
- Mauve
- Mercator

Hierarchical alignment

- Step 1:
  - Determine colinear segments in the input genomes
  - Called a “homology map”
- Step 2:
  - Perform alignment on colinear segments
  - Multiple flavors; we’ll discuss Mummer

Homology map



Homologous Group	Genome 1 Segments	Genome 2 Segments
1	chr1:3306600-3626073:+	chr4:7084404-7540496:+
2	chr2:3626073-3645123:+	chr5:1727254-1819933:-
3	chr2:3645123-3675603:+	chr2:7045783-7084404:+ chr2:7084405-7103943:+

Slide by Colin Dewey

Methods for WGA
1: anchor-based global multiple alignment

These methods try to identify substrings of the sequences under consideration that they are likely parts of a global alignment.
These substrings form „anchors“ in the sequences to be aligned.

These methods first align the anchors and subsequently close the gaps (align the substrings between the anchors).
Anchor-based alignment methods are well suited for aligning very long sequences.

MUMmer is a very successful implementation of this strategy for aligning two genome sequences.

other implementations: QUASAR, REPuter

What is MUMmer?

- A.L. Delcher *et al*. 1999, 2002 Nucleic Acids Res.
- Assume two sequences are closely-related (highly similar)
- MUMmer can align two bacterial genomes in less than 1 minute
- Use **Suffix Tree** to find Maximal Unique Matches
- Maximal Unique Match (MUM) Definition:
  - A subsequence that occurs in two exactly matching copies, once in each input sequence, and that cannot be extended in either direction

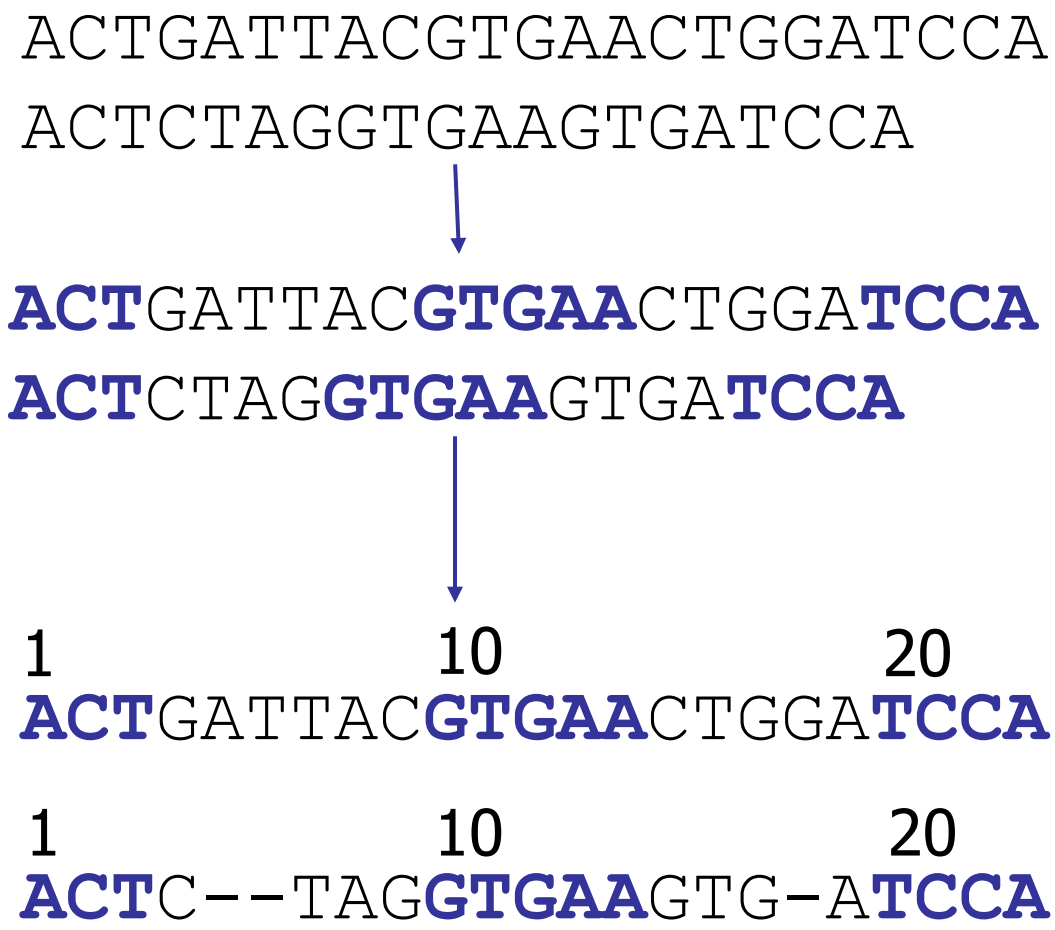
Genome A: `tcgatcGACGATCGCGGCCGTAGATCGAATAACGAGAGAGCATAAcgactta`
Genome B: `gcattaGACGATCGCGGCCGTAGATCGAATAACGAGAGAGCATAAtccagag`

- Key idea: a significantly long MUM is certainly going to be part of the global alignment

Delcher et al. Nucleic Acids Res 27, 2369 (1999)

MUMmer: Key Steps

- Locating MUMs (user-defined length)



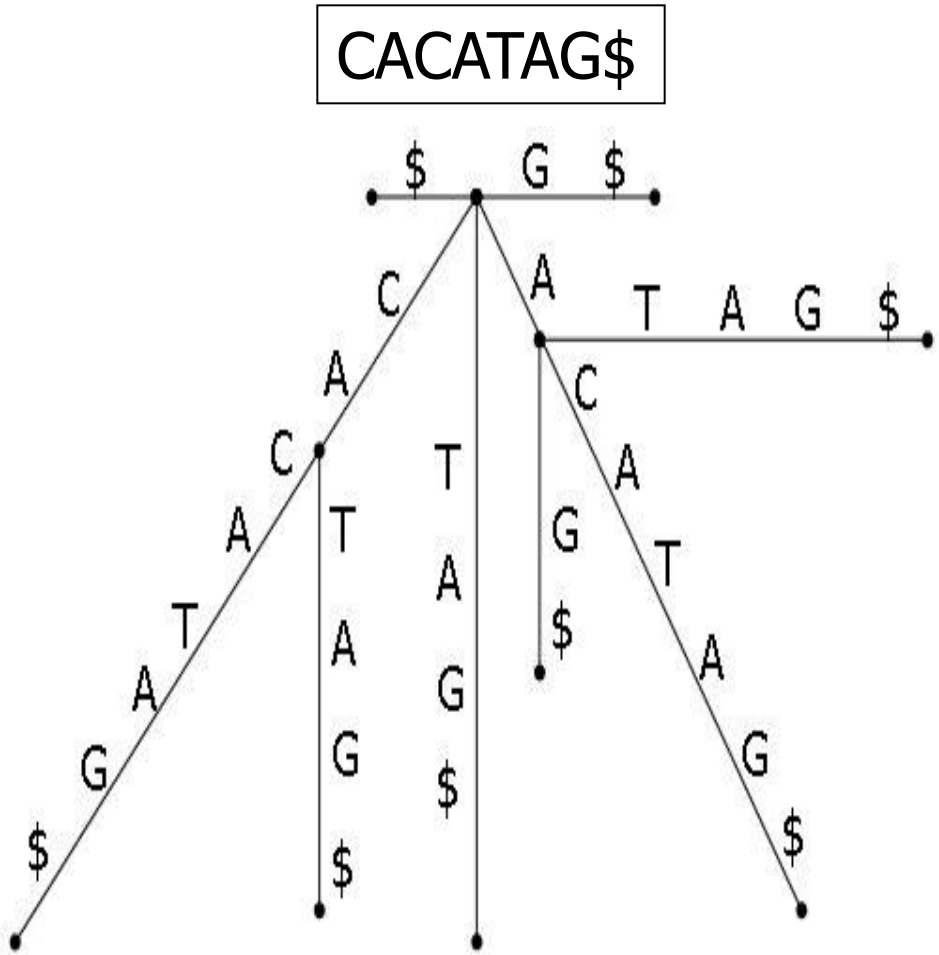
Definition of MUMmers

- For two strings S1 and S2 and a parameter *l*
- The substring *u* is an MUM sequence if:
  - $|u| > l$
  - *u* occurs exactly once in S1 and it occurs exactly once in S2 (uniqueness)
  - For any character *a* neither *ua* nor *au* occurs both in S1 and in S2 (maximality)

How to find MUMs?

- Naïve Approach
  - Compare all subsequences of genome A with all subsequences of genome B $O(n^n)$
- Suffix Tree
  - Naïve approach to build a suffix tree
    - Quadratic time, space
  - McCreight's algorithm
    - Linear time, linear space
    - Clever use of pointers

Suffix Tree



Suffix trees are well-established and have been used for parsing text for over 20 years.

Some properties:

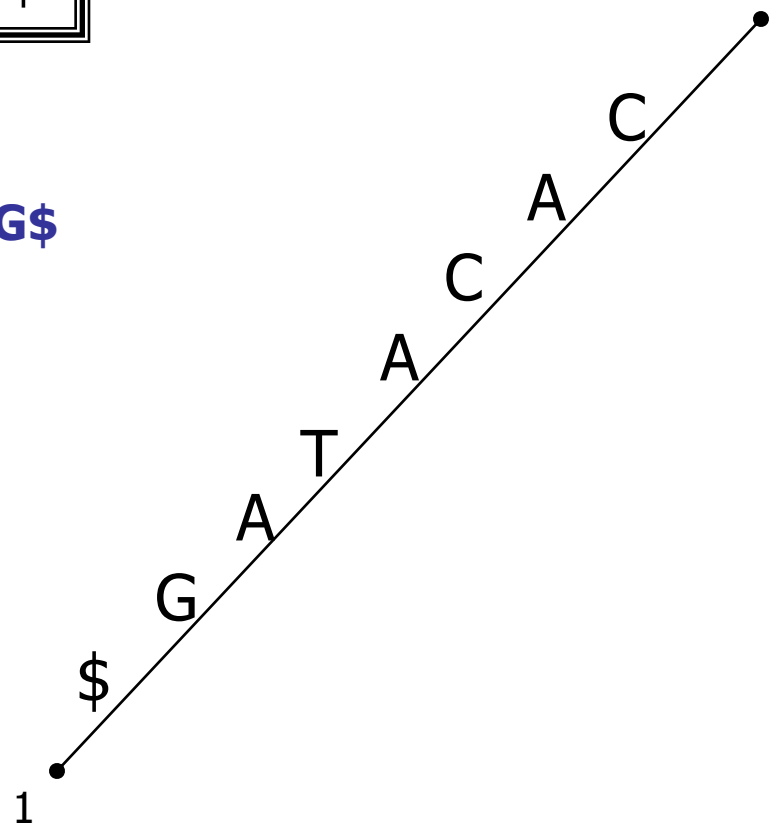
- a “suffix” starts at any position *I* of the sequence and goes until its end.
- sequence of length N string has N suffixes
- N leaves
- Each internal node has at least 2 child nodes
- No 2 edges out of the same node can have edge beginning with the same character
- add $ to the end

Constructing a Suffix Tree

CACATAG$

Suffixes:

1. CACATAG$

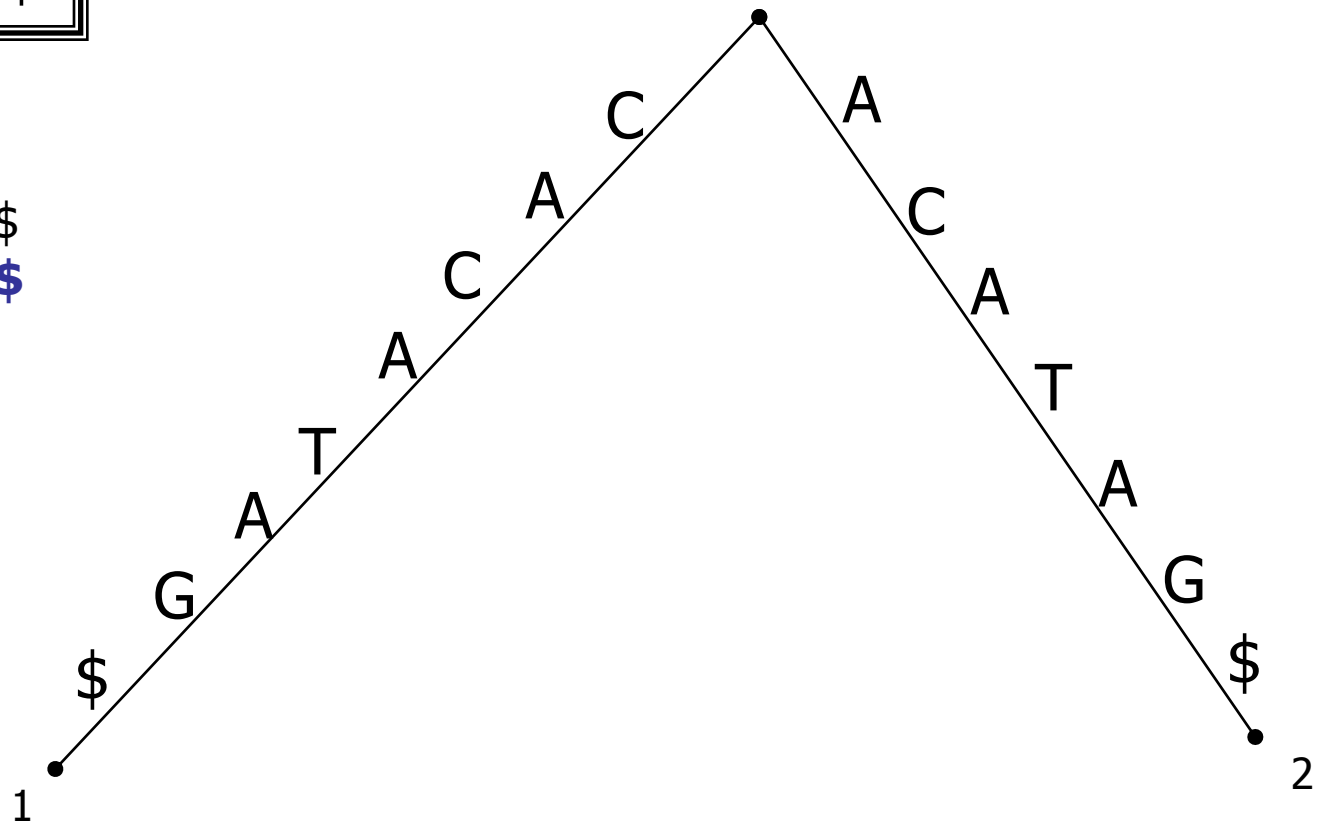


Constructing a Suffix Tree

CACATAG$

Suffixes:

1. CACATAG$
2. ACATAG$

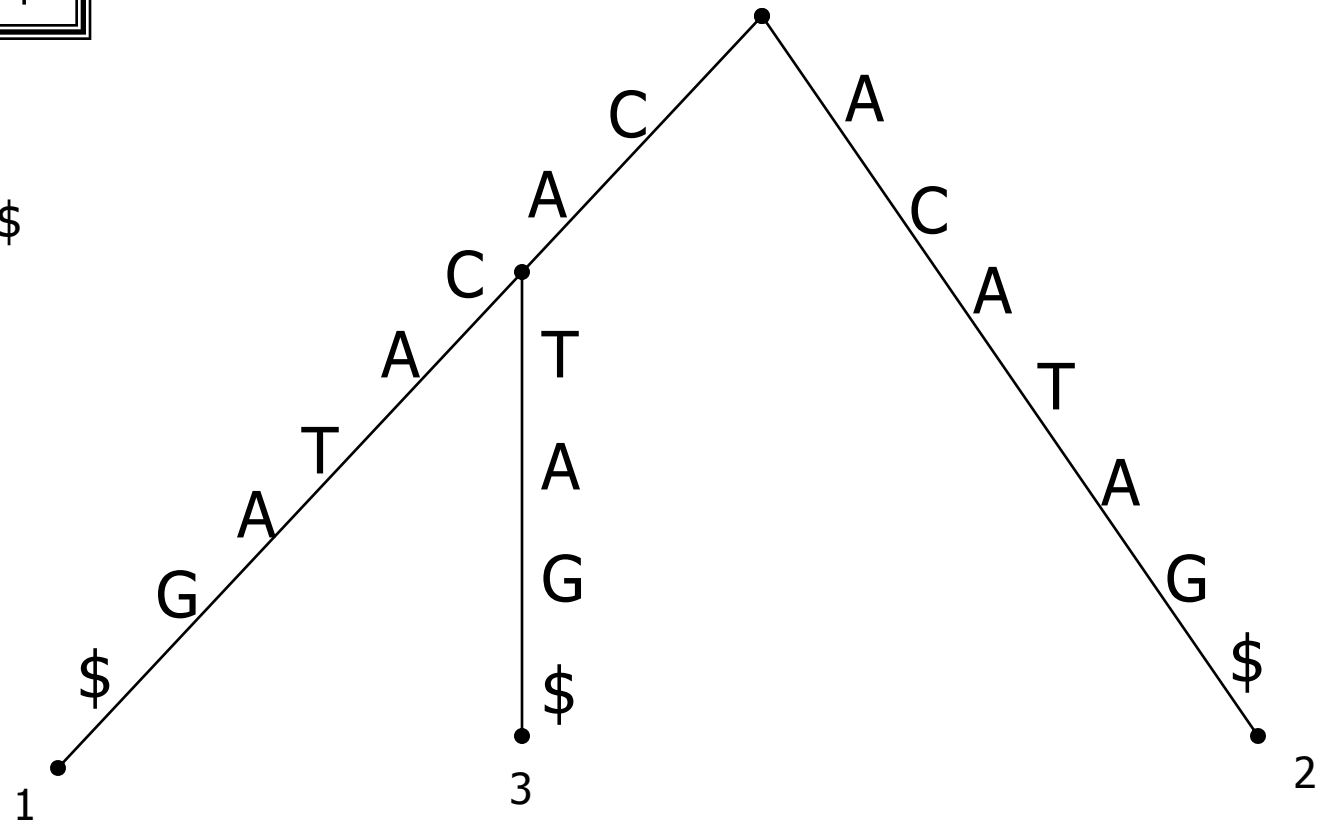


Constructing a Suffix Tree

CACATAG$

Suffixes:

1. CACATAG$
2. ACATAG$
3. **CATAG$**

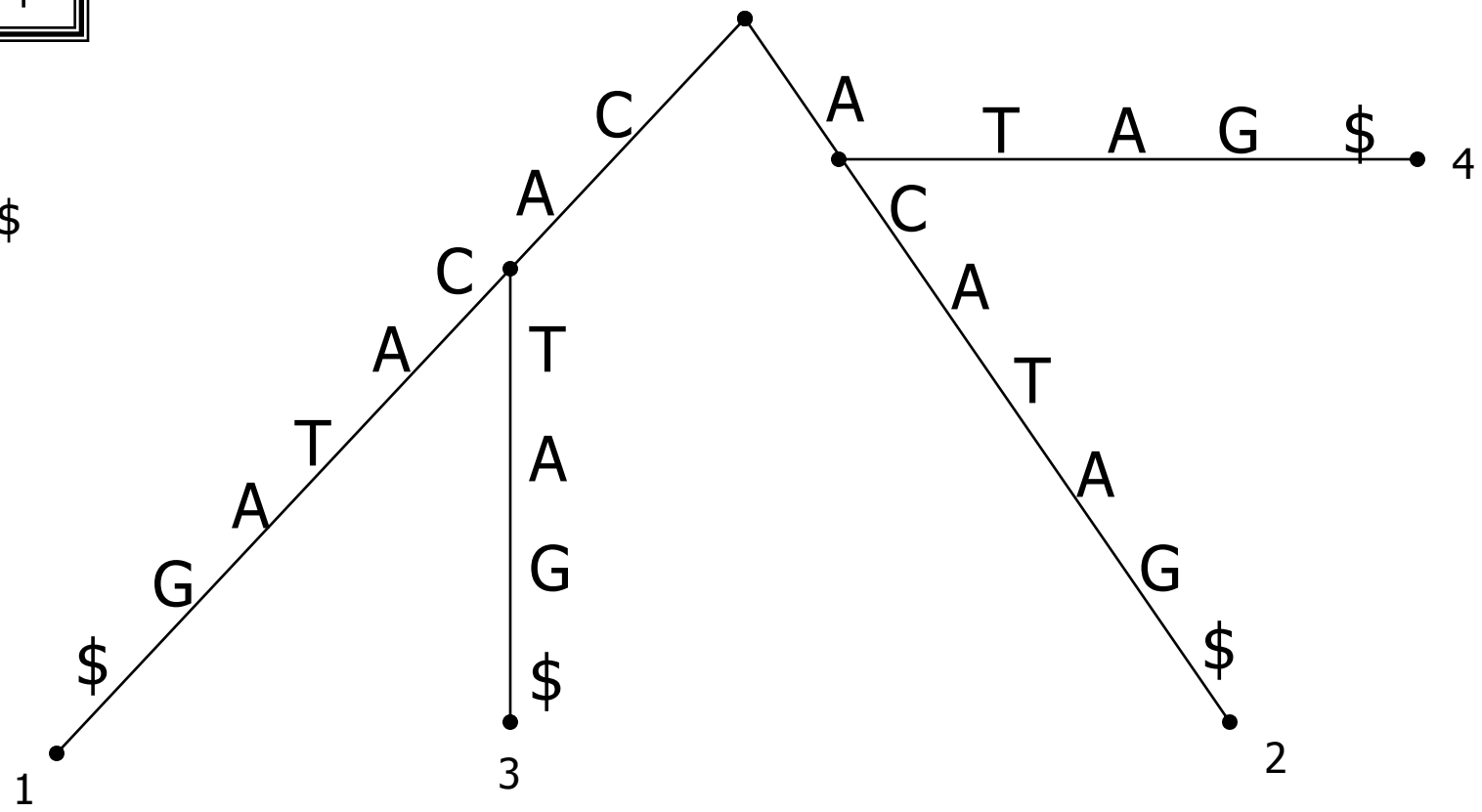


Constructing a Suffix Tree

CACATAG$

Suffixes:

1. CACATAG$
2. ACATAG$
3. CATAG$
4. **ATAG$**

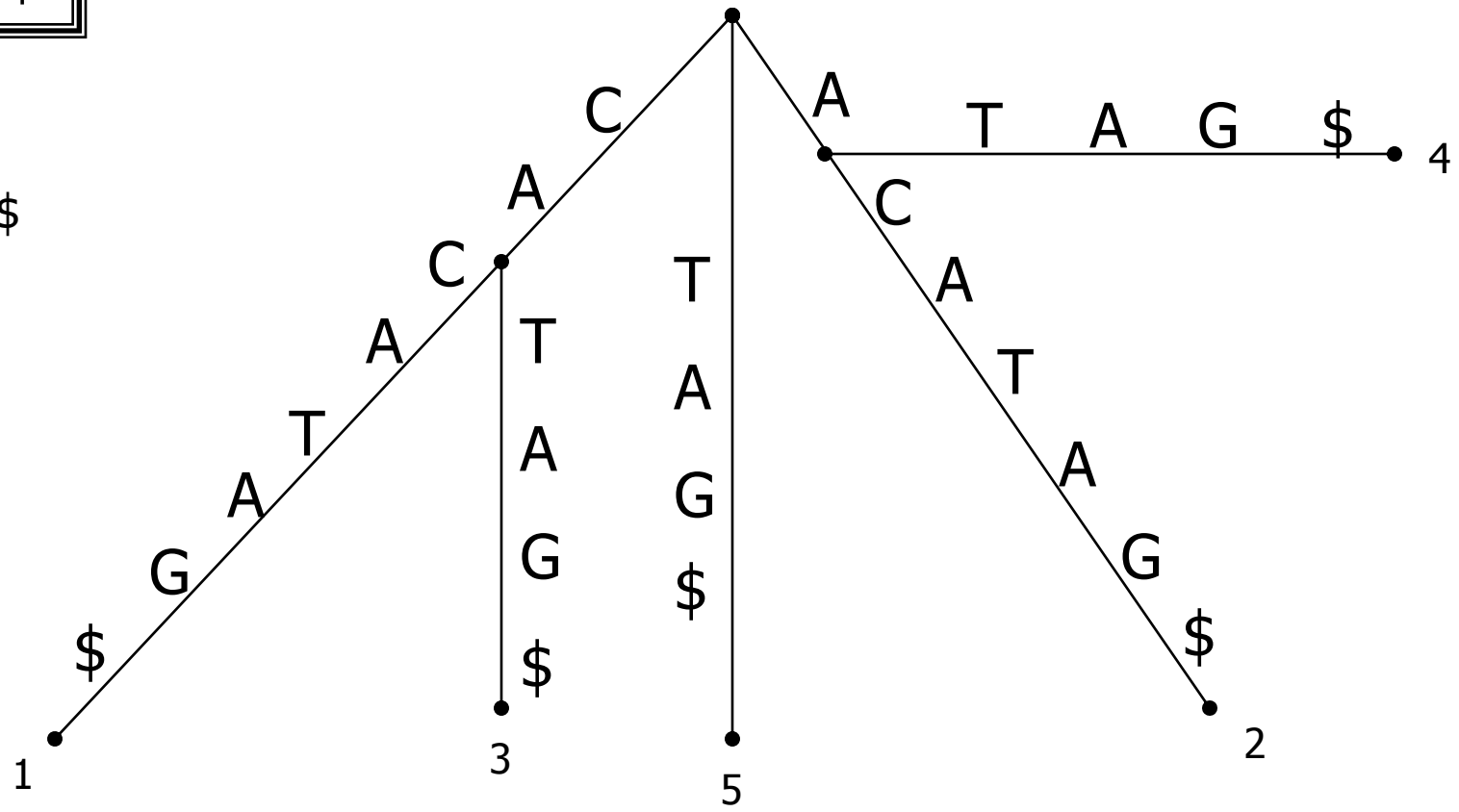


Constructing a Suffix Tree

CACATAG$

Suffixes:

1. CACATAG$
2. ACATAG$
3. CATAG$
4. ATAG$
5. **TAG$**

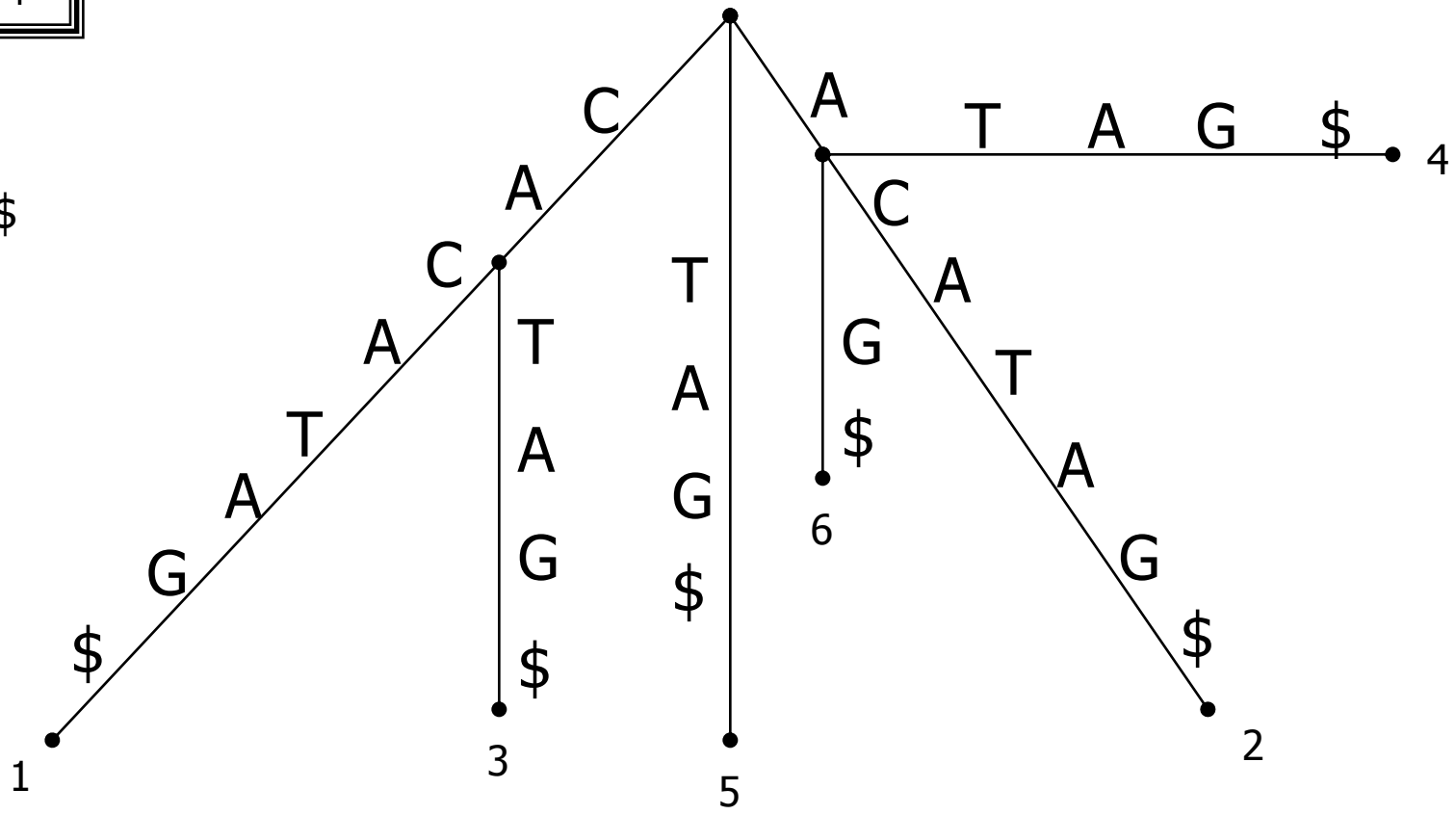


Constructing a Suffix Tree

CACATAG$

Suffixes:

1. CACATAG$
2. ACATAG$
3. CATAG$
4. ATAG$
5. TAG$
6. **AG$**

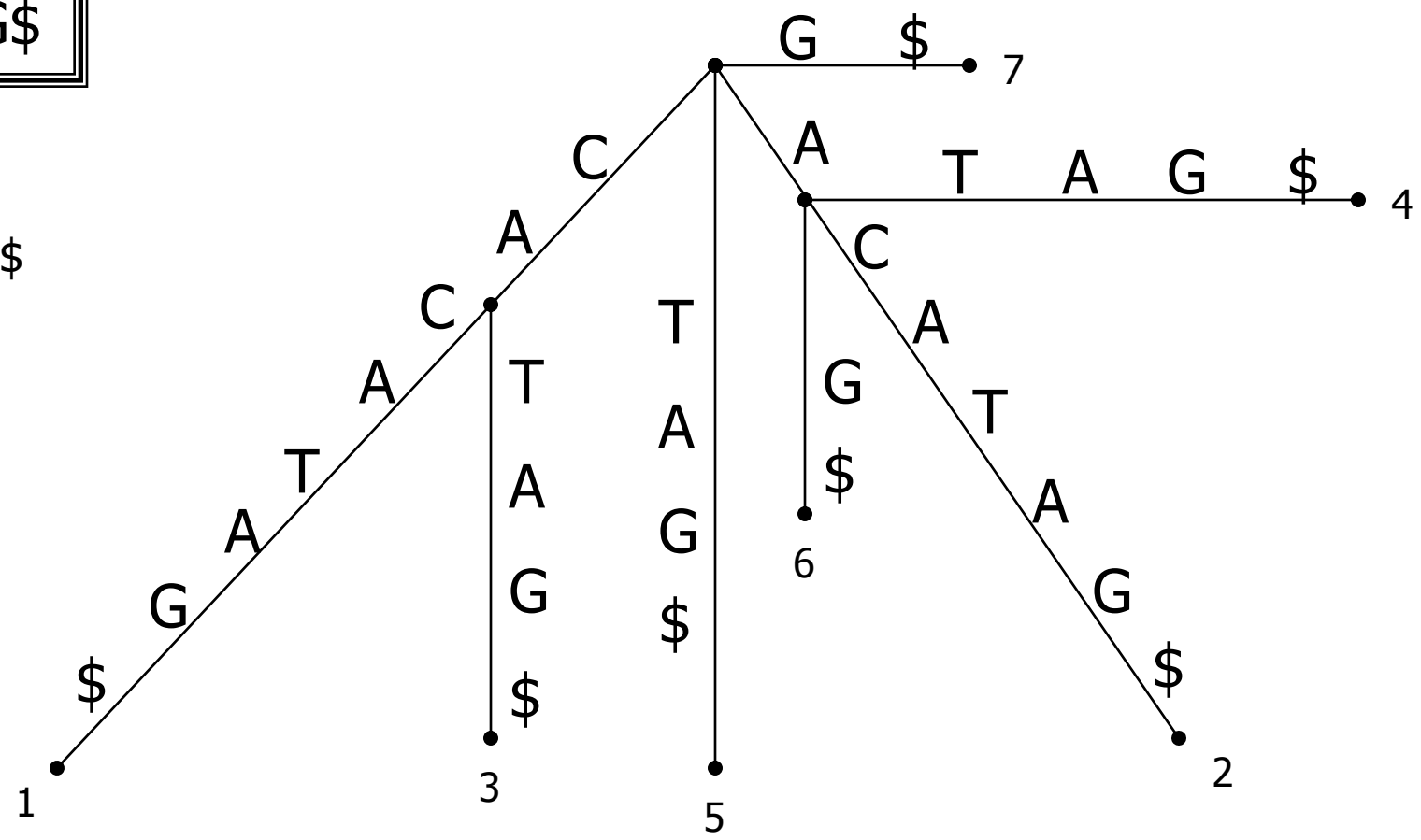


Constructing a Suffix Tree

CACATAG$

Suffixes:

1. CACATAG$
2. ACATAG$
3. CATAG$
4. ATAG$
5. TAG$
6. AG$
7. **G$**

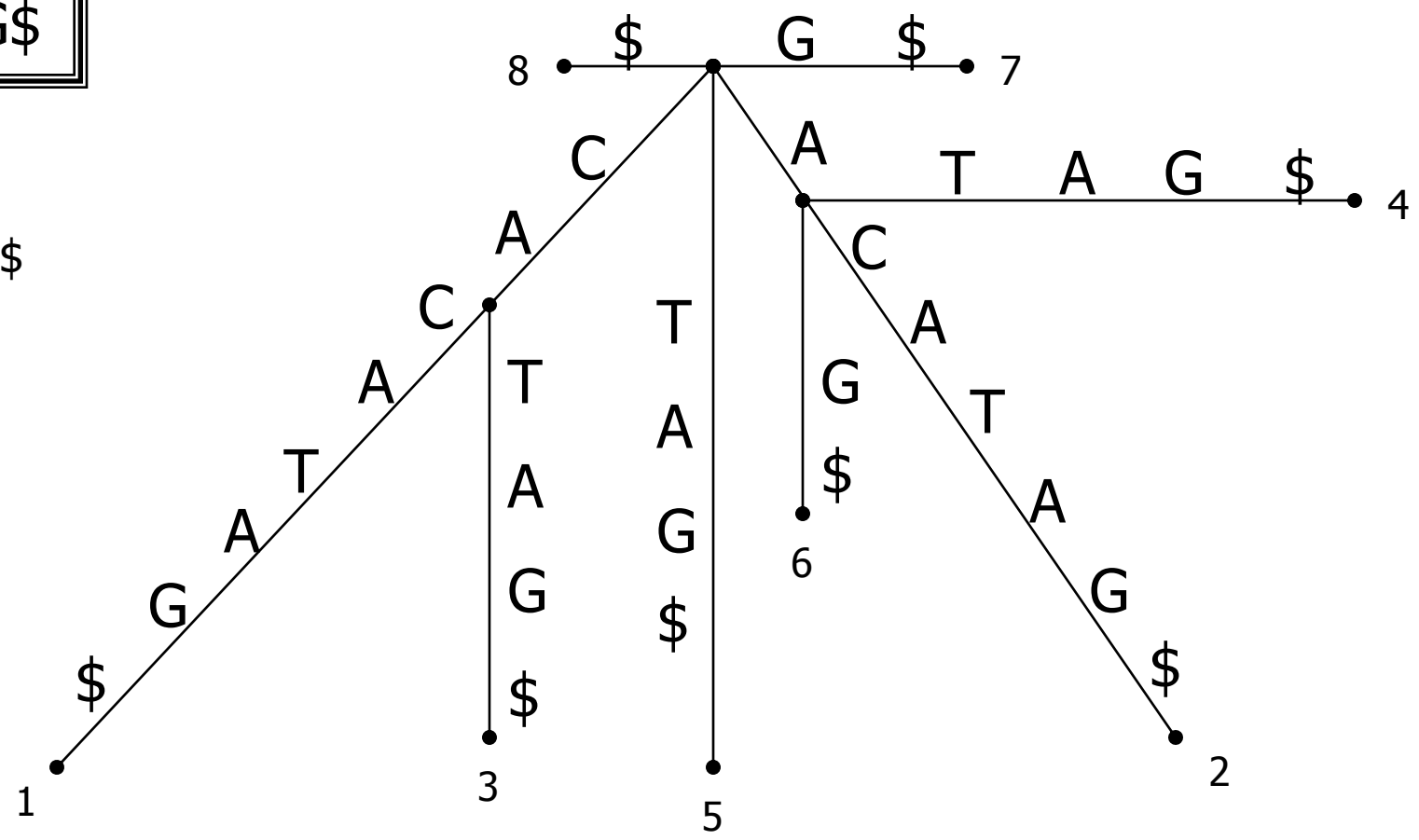


Constructing a Suffix Tree

CACATAG$

Suffixes:

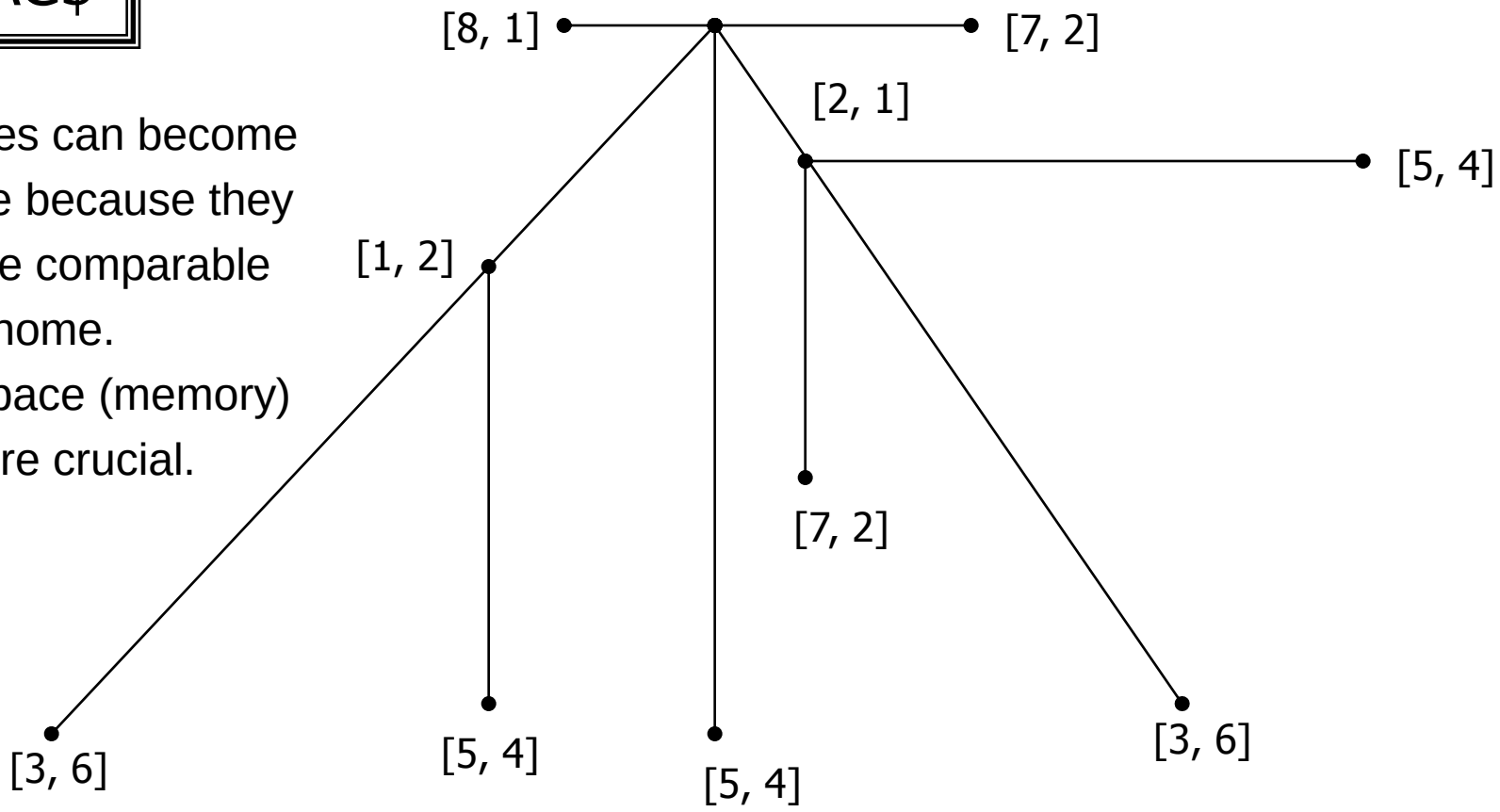
1. CACATAG$
2. ACATAG$
3. CATAG$
4. ATAG$
5. TAG$
6. AG$
7. G$
8. **$**



Saving Space

CACATAG$

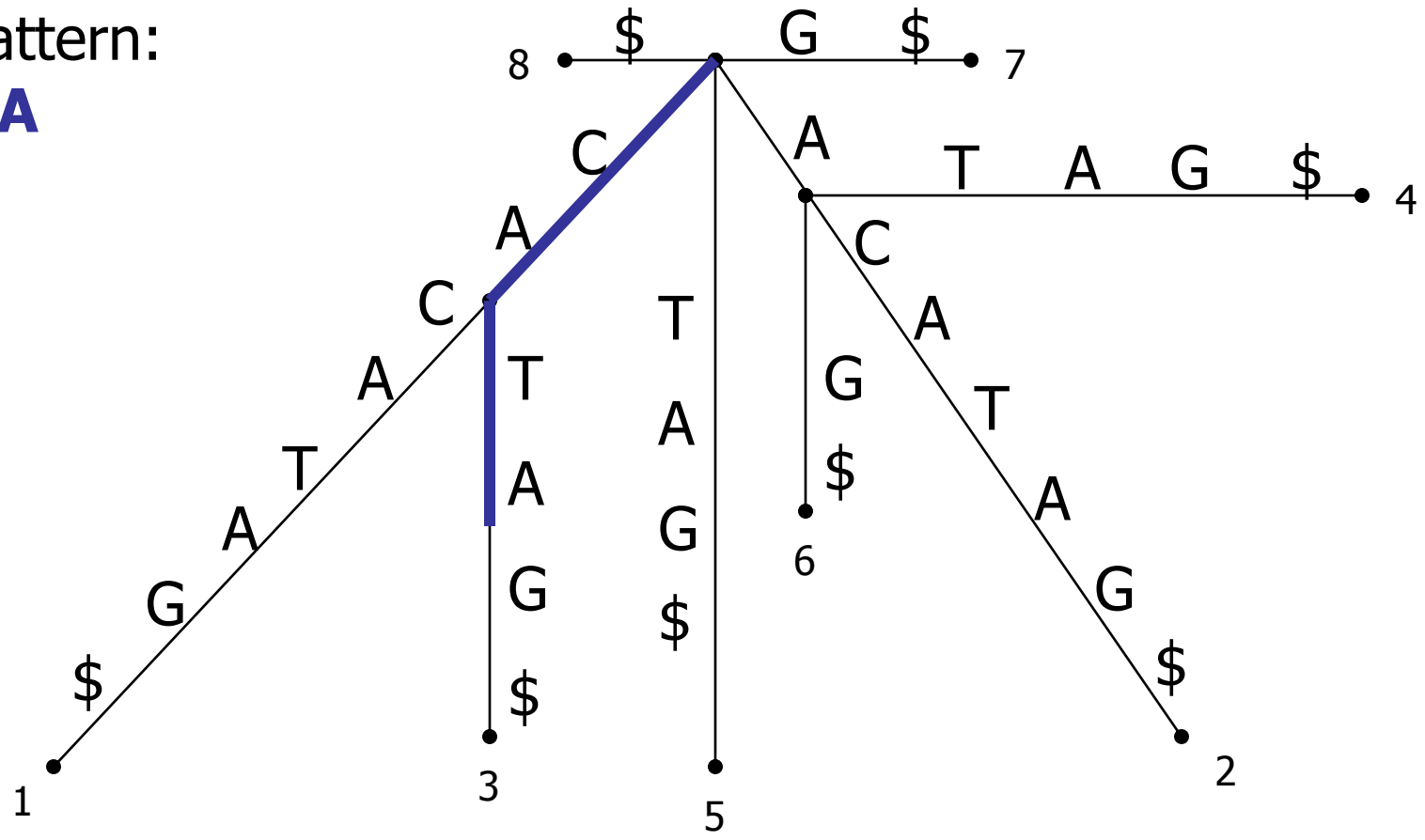
Suffix trees can become very large because they are of size comparable to the genome.
Saving space (memory) is therefore crucial.



Searching a Suffix Tree

Search Pattern:

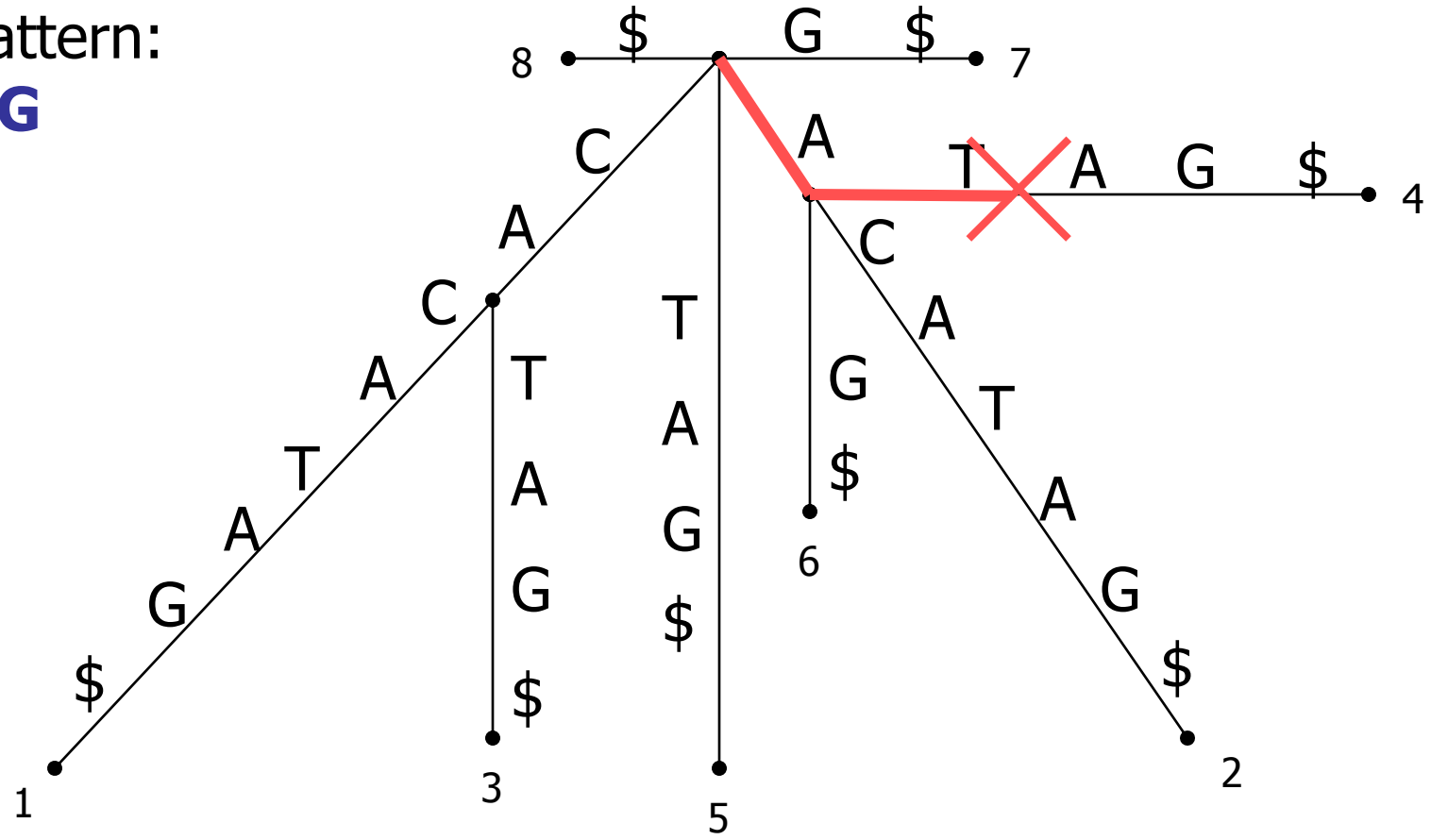
CATA



Searching a Suffix Tree

Search Pattern:

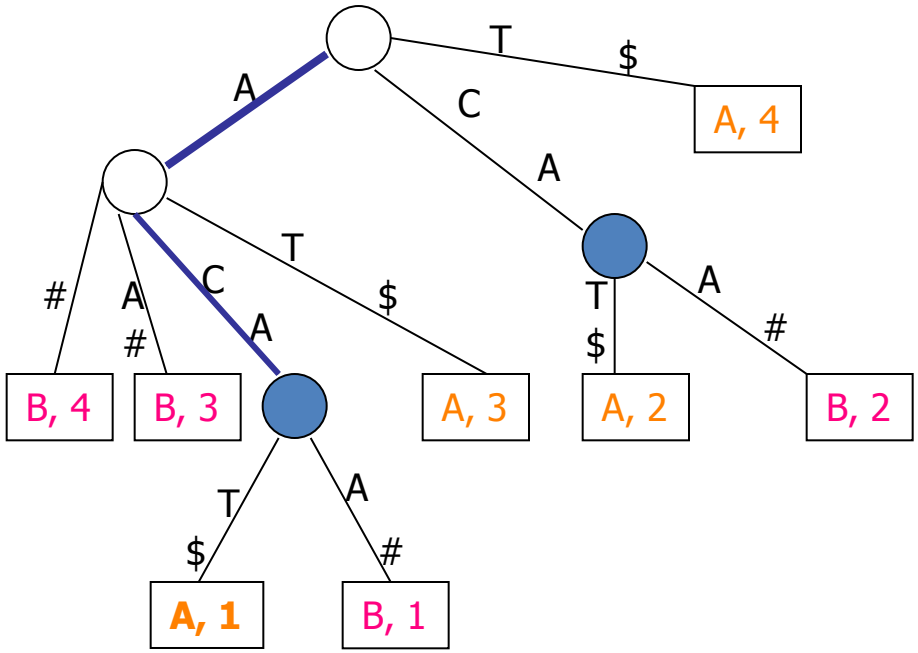
ATCG



MUMmer 1.0: How to find MUMs?

- Build a suffix tree from all suffixes of genome A
- Insert every suffix of genome B into the suffix tree
- Label each leaf node with the genome it represents

How MUMmer 1.0 finds MUMs?

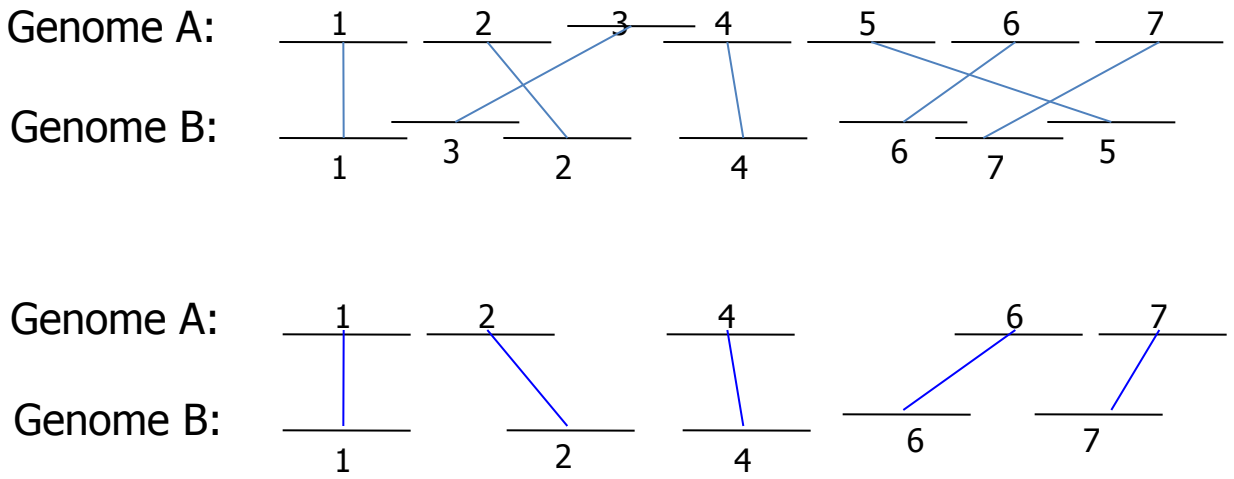


Genome A: ACAt$
Genome B: ACAa#

- **Unique Match:**
  - **internal node with 2 child nodes where the leaf nodes are from different genomes!**
- Unique matches may not be maximal
- Maximal matches can be found by investigating the unique match furthest away from the root node
- User specifies the minimum length of MUMs to be identified

Sorting the MUMs

- MUMs are sorted according to their positions in genome A
- Use a variation of Longest Increasing Subsequence (LIS) to find sequences in ascending order in both genomes
  - Takes into account lengths of sequences represented by MUMs, and overlaps
  - O(*k*log*k*) running time, *k* = number of MUMs



Top alignment shows all MUMs. The shift of MUM 5 in Genome *B* indicates a transposition. The shift of MUM 3 could be simply a random match or part of an inexact repeat sequence. Bottom alignment shows just LIS of MUMs in Genome *B*.

4 types of gaps in MUM alignment

These examples are drawn from the alignment of the two *M.tuberculosis* genomes.

1. SNP: exactly one base (indicated by ^) differs between the two sequences. It is surrounded by exact-match sequence.

```
Genome A:  cgtcatgggcgttcgtcgttg
Genome B:  cgtcatgggcattcgtcgttg
                     ^
```

2. Insertion: a sequence that occurs in one genome but not the other.

```
Genome A:  cggggtaaccgc.................cctggtcggg
Genome B:  cggggtaaccgcgttgctcggggtaaccgccctggtcggg
                     ^^^^^^^^^^^^^^^^^^
```

3. Highly polymorphic region: many mutations in a short region.

```
Genome A:  ccgcctcgcctgg.gctggcgcccgctc
Genome B:  ccgcctcgccagttgaccgcgcccgctc
                   ^ ^^ ^^^
```

4. Repeat sequence: the repeat is shown in uppercase. Note that the first copy of the repeat in Genome *B* is imperfect, containing one mismatch to the other three identical copies.

```
Genome A:  cTGGGTGGGACAACGTaaaaaaaaaTGGGTGGGACAACGTc
Genome B:  aTGGGTGGGGCgACGTgggggggggTGGGTGGGACAACGTa
           ^          ^                           ^
```

Delcher et al.
Nucleic Acids Res 27, 2369

Closing the Gaps

- SNP
  - Simple case: gap of one base between adjacent MUMs
  - when adjacent to repeat sequences: treated as tandem repeats
- Variable / Polymorphic Region
  - Small region: dynamic programming alignment
  - Large region: recursively apply MUMmer with reduced minimum cut-off length
- Insertions / Deletion
  - Transposition: out of alignment order
  - Simple insertion: not in the alignment
- Repeats
  - Tandem repeats are detected by overlapping MUMs
  - Other repeats (i.e. duplication) are treated as gaps

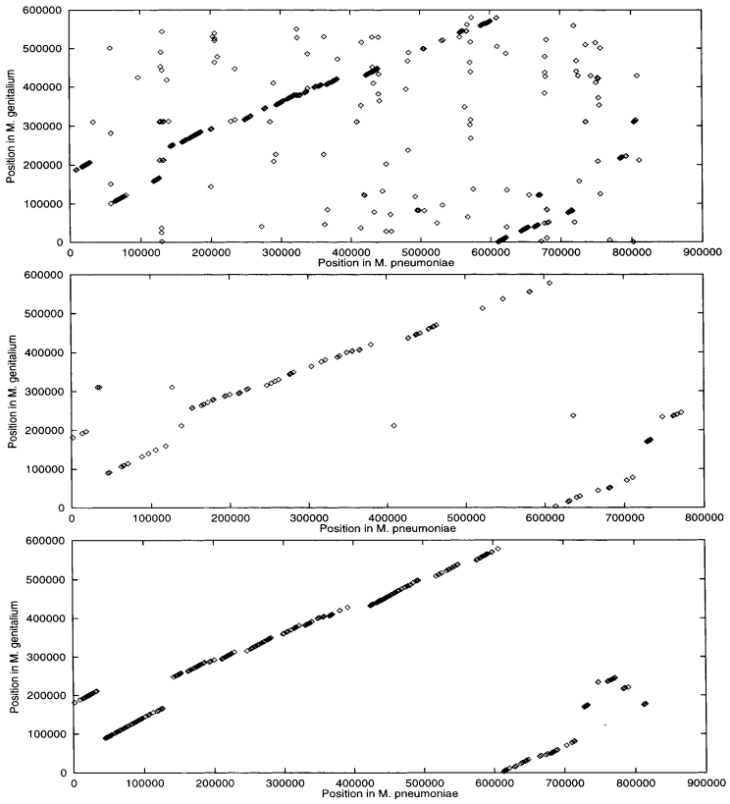
some positions are not uniquely defined

```
Genome A:   uniqueAAGGAAGGAAGGsequence
Genome B:   uniqueAAGGAAGG....sequence
            |           |           |
Position:   0          10          20
```

Repeat sequences surrounded by unique sequences. For the purposes of illustration, other characters besides the four DNA nucleotides are used.

Delcher et al. Nucleic Acids Res 27, 2369 (1999)

Example: alignment of 2 micoorganisms

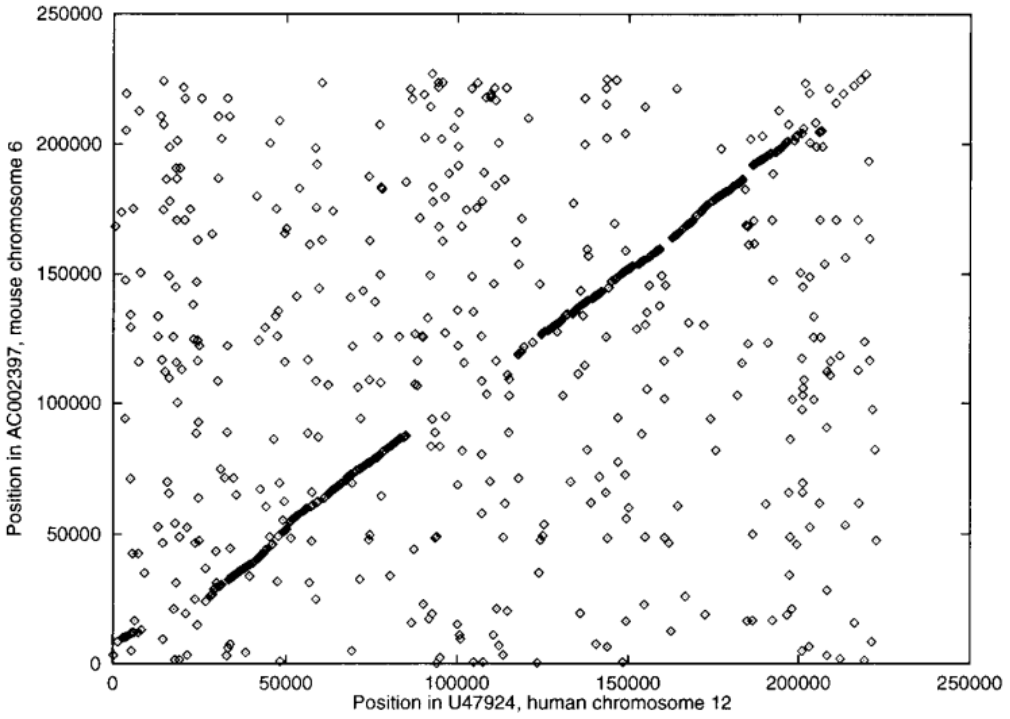


The genome of *M.genitalium* is only 2/3 the size of *M.pneumoniae*. Alignment of *M.genitalium* and *M.pneumoniae* using FASTA (top), 25mers (middle) and MUMs (bottom). In the FASTA plot a point corresponds to similar genes. In the 25mer plot, each point indicates a 25-base sequence that occurs exactly once in each genome. In the MUM plot, points correspond to MUMs.

Delcher et al. Nucleic Acids Res 27, 2369 (1999)

Example: alignment human:mouse

Here: alignment of a 222 930 bp subsequence of human chromosome 12p13, subsequence of mouse chromosome 6. Each point in the plot corresponds to an MUM of15 bp.



MUMmer 2.0: Improvements

- Memory Requirement
- Finding Initial Exact Matches
  - Store only one sequence as suffix tree
  - Suffixes of second sequence are streamed against the tree
  - Saves 1/2 of memory usage
- Ability to align protein sequences

- Whole Genome Alignments can now be done on 100 Mb genomes within minutes or less

Current disadvantages

linear space still means huge memory requirement for medium size genomes – e.g. 100mb genome => ~4Gb of working memory

- Repeats are not handled specifically
    - tandem repeats as overlapping MUMs
    - duplications are ignored

- Smith-Waterman DP used to fill in small gaps; large gaps ignored
- not efficient for aligning distantly related genomes which have few MUMs and many gaps

Other methods

- Mauve
  - Constructs an orthology map along with the alignment
  - Based on MUMs/suffix trees
- Mercator
  - Constructs an orthology map based on “landmarks”
  - Uses MAVID to do the alignment

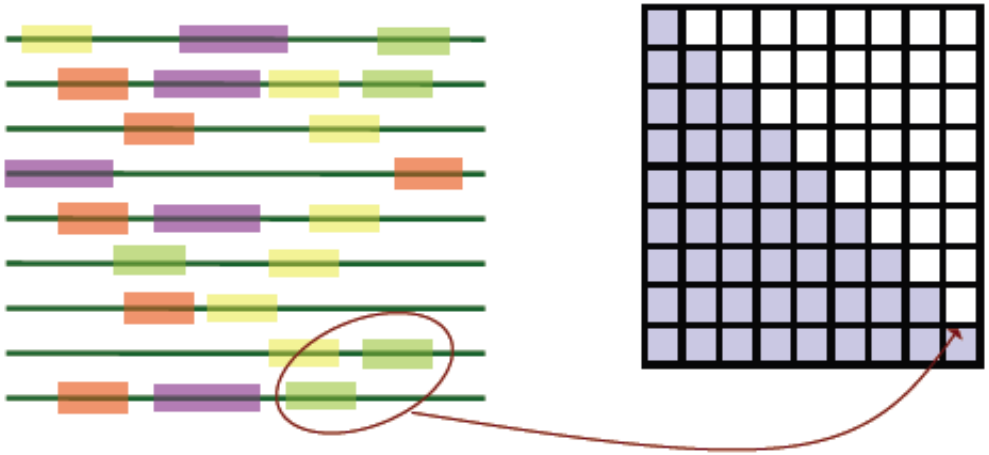
Mauve

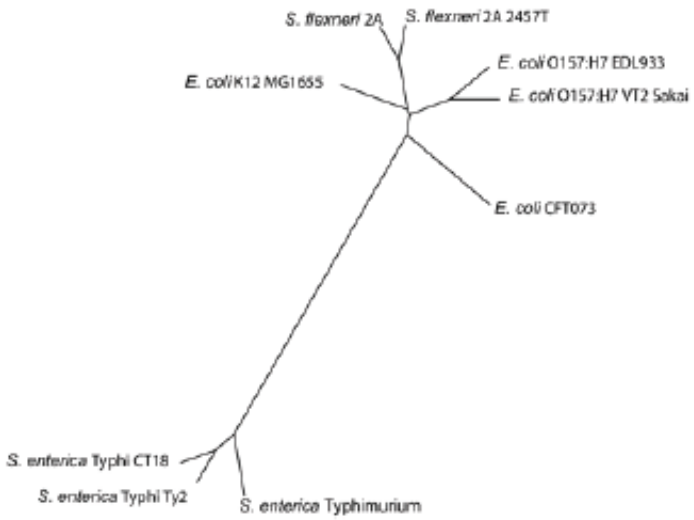
- Proceeds as follows:
  - Finds “multi-MUMs”
  - Generates a guide tree based on MUMs
  - Find colinear blocks
  - Perform a progressive alignment of blocks using previously generated guide tree

1. find multi-MUMs in sequences

2. calculate pairwise distances

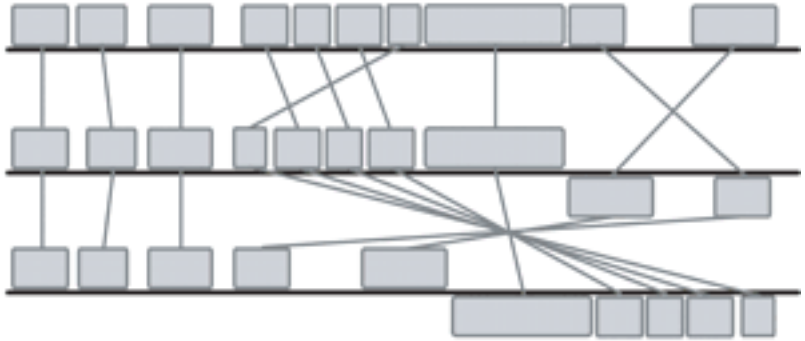
3. run neighbor-joining to get guide tree;



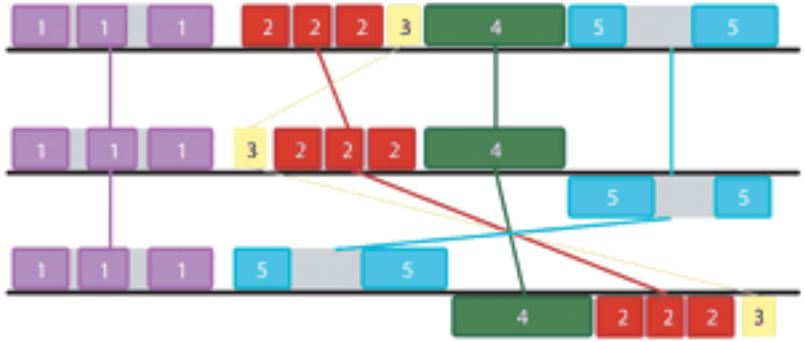


Slide from Colin Dewey

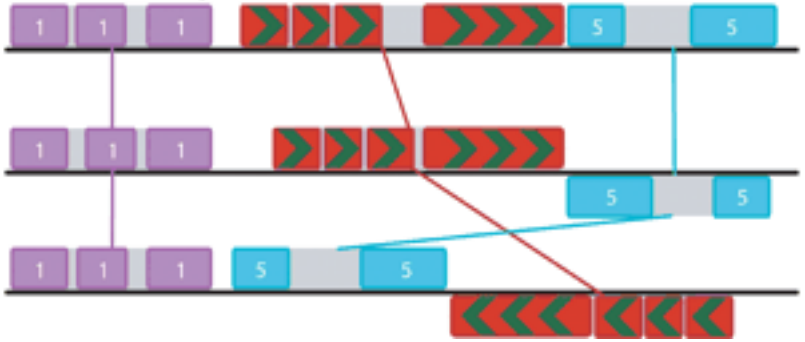
A) The initial set of matching regions:

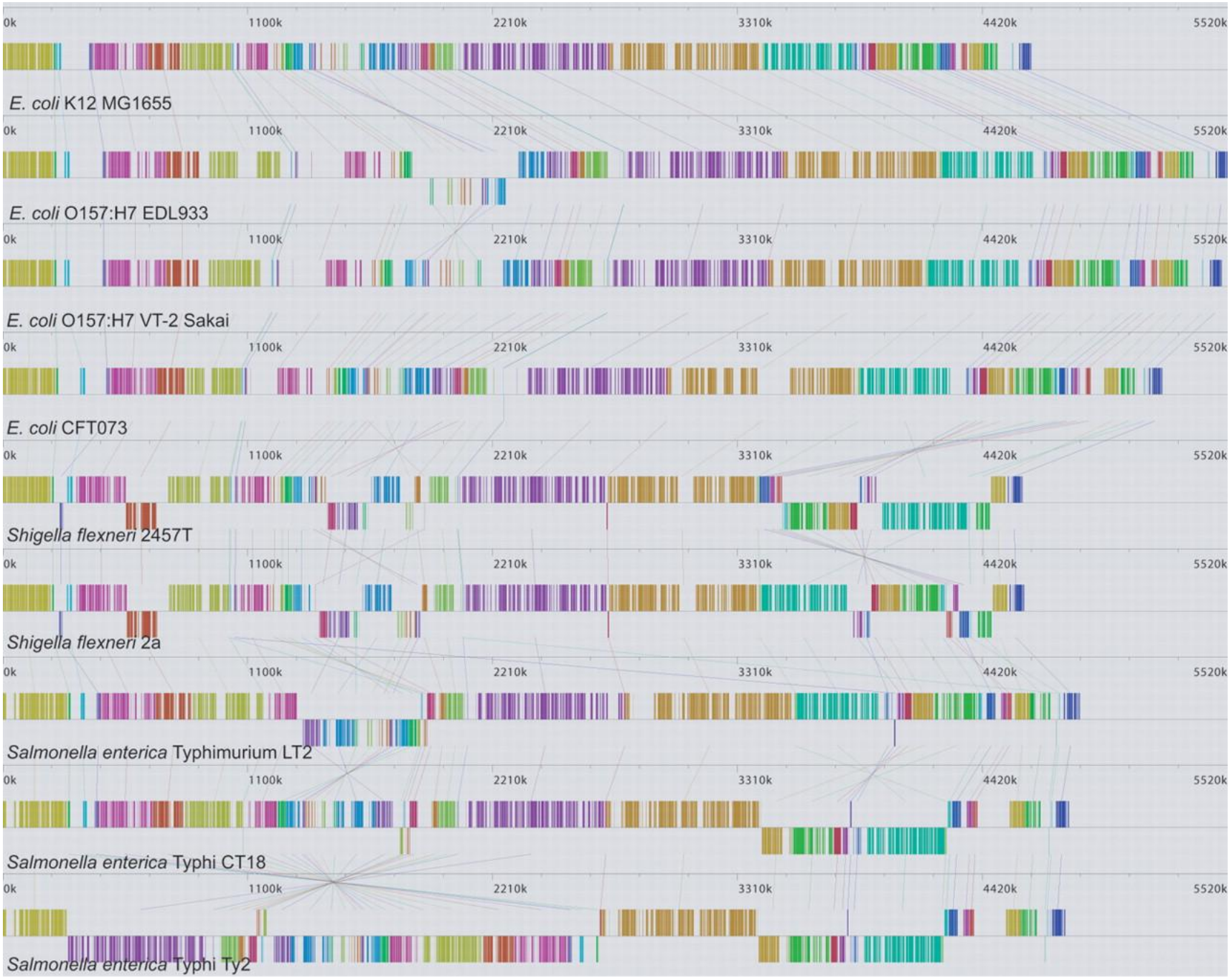


B) Minimum partitioning into collinear blocks:



C) After removing block 3:




0k
1100k
2210k
3310k
4420k
5520k
E. coli K12 MG1655
E. coli O157:H7 EDL933
E. coli O157:H7 VT-2 Sakai
E. coli CFT073
Shigella flexneri 2457T
Shigella flexneri 2a
Salmonella enterica Typhimurium LT2
Salmonella enterica Typhi CT18
Salmonella enterica Typhi Ty2

Methods for WGA
2: Extending shot local alignments

• These Methods follow a general strategy of iteratively merging two multiple alignments of two disjoint subsets of sequences into a single multiple alignment of the union of those subsets.

• Construct a hash table on either the query string, or the database string (or both) for all possible substrings of a pre-specified size (say *l*)

• Find exactly matching substrings of length *l* using this hash table (seeds).

• In the second phase, these seeds are extended in both directions, and combined if possible, in order to find better alignments.

These methods require fast local alignment tools, for example:

BLAST, MegaBLAST, BL2SEQ, Wu-blast etc.

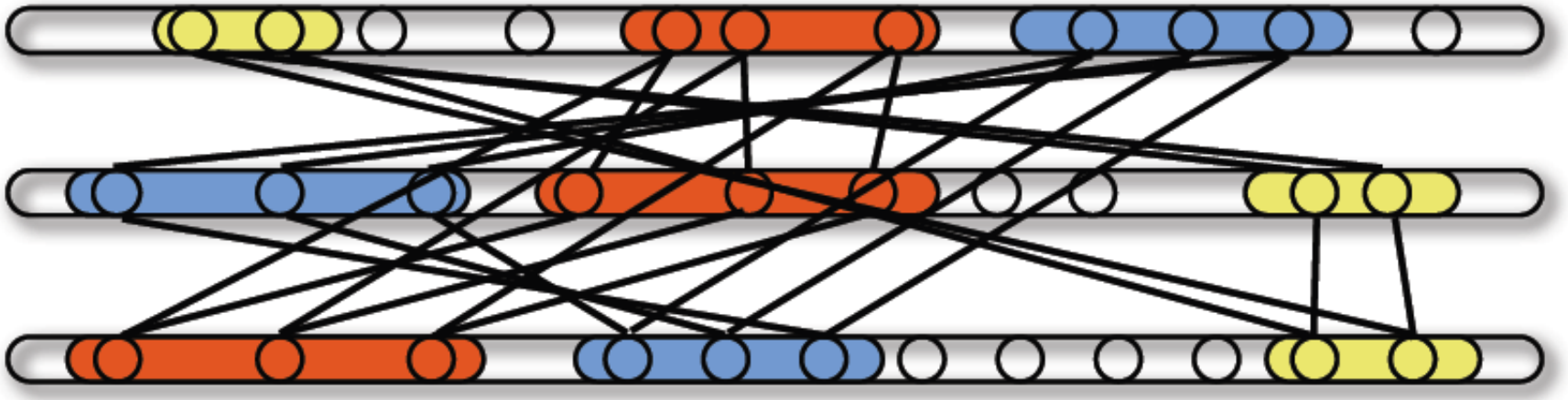
Gene based alignment

- Several methods start with a known collection of genes as anchors
- These are compared in an all vs. all manner
- A graph is then constructed where nodes are known genes and edges are BLAT results weighted by alignment score

Overview

k-partite graph with edge weights

vertices = intervals, edges = sequence similarity



Slide from Colin Dewey

Other uses

- Filling gaps in draft genome sequences (see later classes)
- Reconstructing ancestral genomes
- Understanding important food-making processes

Comparative genomic websites for various computational tools and databases

Comparative genomic visualization tools	Websites
VISTA	http://www-gsd.lbl.gov/vista/
PipMaker	http://bio.cse.psu.edu/pipmaker/
Whole-genome annotation browsers	
NCBI Map Viewer	http://www.ncbi.nlm.nih.gov
UCSC Genome Browser	http://genome.ucsc.edu/
Ensembl	http://www.ensembl.org/
Whole-genome comparative genomic browsers	
UCSC Genome Browser	http://genome.ucsc.edu/
VISTA Genome Browser	http://pipeline.lbl.gov/
PipMaker	http://bio.cse.psu.edu/genome/hummus/
Custom comparisons to whole genomes	
GenomeVista (AVID)	http://pipeline.lbl.gov/cgi-bin/GenomeVista
UCSC Genome Browser (BLAT)	http://genome.ucsc.edu/
ENSEMBL (SSAHA)	http://www.ensembl.org/
NCBI (BLAST)	http://www.ncbi.nlm.nih.gov/blast/

Pennachio, Rubin, J Clin Invest. 111, 1099 (2003)

Scoring Alignments

• Alignments can be scored by comparing the resulting alignment to a background (random) model.

Independent

$$P(x, y \mid I) = \prod_i q_{x_i} \prod_j q_{x_j}$$

Related

$$P(x, y \mid M) = \prod_i p_{x_i y_i}$$

Score for alignment:

$$S = \sum_i s(x_i, y_i)$$

where:

$$s(a, b) = \log\left(\frac{p_{a,b}}{q_a q_b}\right)$$

Can be computed for each pair of letters

Scoring Alignments

• Alignments can be scored by comparing the resulting alignment to a background (random) model.

In other words, we are trying to find an alignment that maximizes the likelihood ratio of the aligned pair compared to the background model

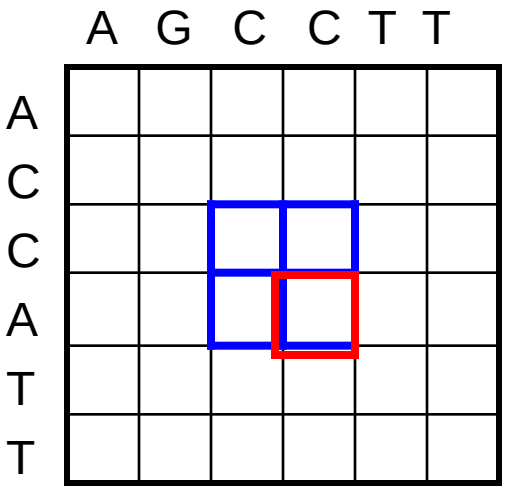
Score for alignment: $S = \sum_i s(x_i, y_i)$

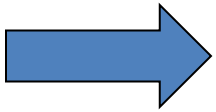
where: $s(a,b) = \log\left(\frac{p_{a,b}}{q_a q_b}\right)$

Computing optimal alignment: The Needham-Wuncsh algorithm

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

d is a penalty for a gap





F(i-1,j-1)	*F(i-1,j)*
F(i,j-1)	*F(i,j)*

Example

Assume a simple model where *S(a,b) = 1* if *a=b* and -5 otherwise.

Also, assume that d = -1

		A	G	C	C	T	T
	0	-1	-2	-3	-4	-5	-6
A	-1						
C	-2						
C	-3						
A	-4						
T	-5						
T	-6						

Example

Assume a simple model where $S(a,b) = 1$ if $a=b$ and -5 otherwise.

Also, assume that d = -1

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

		A	G	C	C	T	T
	0	-1	-2	-3	-4	-5	-6
A	-1	1					
C	-2						
C	-3						
A	-4						
T	-5						
T	-6						

Example

Assume a simple model where $S(a,b) = 1$ if $a=b$ and -5 otherwise.

Also, assume that d = -1

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

		A	G	C	C	T	T
	0	-1	-2	-3	-4	-5	-6
A	-1	1	0				
C	-2	0					
C	-3						
A	-4						
T	-5						
T	-6						

Example

Assume a simple model where $S(a,b) = 1$ if $a=b$ and -5 otherwise.

Also, assume that d = -1

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

		A	G	C	C	T	T
	0	-1	-2	-3	-4	-5	-6
A	-1	1	0	-1	-2	-3	-4
C	-2	0	-1				
C	-3	-1					
A	-4	-2					
T	-5	-3					
T	-6	-4					

Example

Assume a simple model where *S(a,b) = 1* if *a=b* and -5 otherwise.

Also, assume that d = -1

		A	G	C	C	T	T
	0	-1	-2	-3	-4	-5	-6
A	-1	1	0	-1	-2	-3	-4
C	-2	0	-1	1	0	-1	-2
C	-3	-1	-2	0	2	1	0
A	-4	-2	-3	-1	1	0	-1
T	-5	-3	-4	-2	0	2	1
T	-6	-4	-5	-3	-1	1	3

Example

Assume a simple model where *S(a,b) = 1* if *a=b* and -5 otherwise.

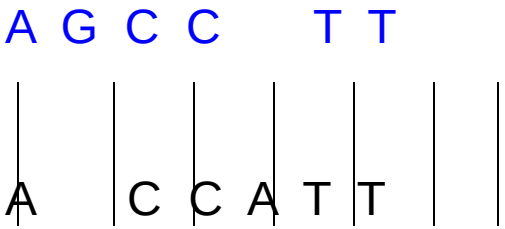
Also, assume that d = -1

		A	G	C	C	T	T
	0	-1	-2	-3	-4	-5	-6
A	-1	1	0	-1	-2	-3	-4
C	-2	0	-1	1	0	-1	-2
C	-3	-1	-2	0	2	1	0
A	-4	-2	-3	-1	1	0	-1
T	-5	-3	-4	-2	0	2	1
T	-6	-4	-5	-3	-1	1	**3**

Example

Assume a simple model where *S(a,b) = 1* if *a=b* and -5 otherwise.

Also, assume that d = -1



		A	G	C	C	T	T
	0	-1	-2	-3	-4	-5	-6
A	-1	1	0	-1	-2	-3	-4
C	-2	0	-1	1	0	-1	-2
C	-3	-1	-2	0	2	1	0
A	-4	-2	-3	-1	1	0	-1
T	-5	-3	-4	-2	0	2	1
T	-6	-4	-5	-3	-1	1	3