Alignment of Long Sequences

BMI/CS 776

www.biostat.wisc.edu/bmi776/

Spring 2019

Colin Dewey

colin.dewey@wisc.edu

Goals for Lecture

Key concepts

- how large-scale alignment differs from the simple case
- the canonical three step approach of large-scale aligners
- using suffix trees to find maximal unique matching subsequences (MUMs)
- The MUMmer system for whole-genome alignment

Pairwise Large-Scale Alignment: Task Definition

Given

- a pair of large-scale sequences (e.g. chromosomes)
- a method for scoring the alignment (e.g. substitution matrices, insertion/deletion parameters)

Do

- construct global alignment: identify all matching positions between the two sequences

Large Scale Alignment Example

Mouse Chr6 vs. Human Chr12

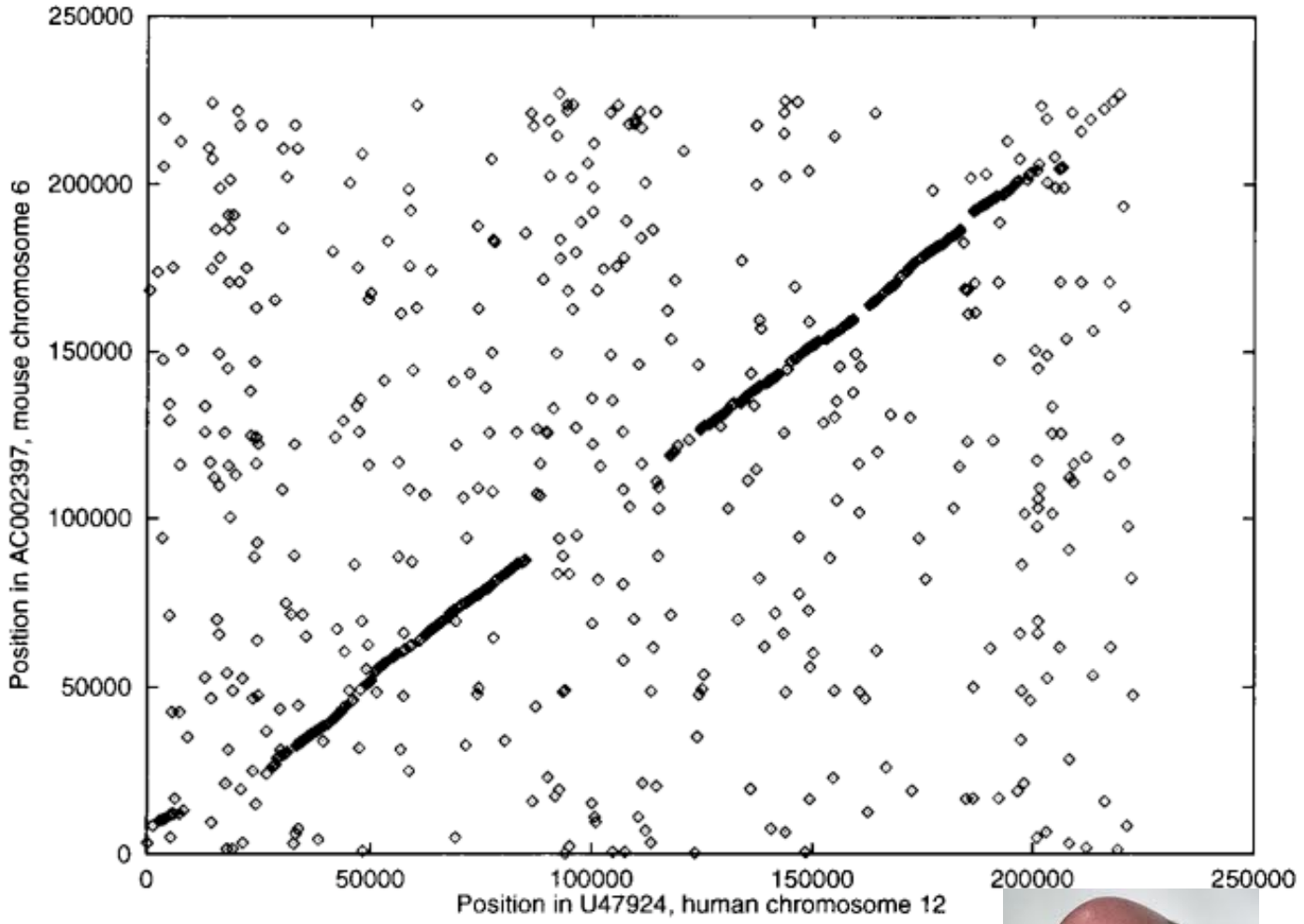


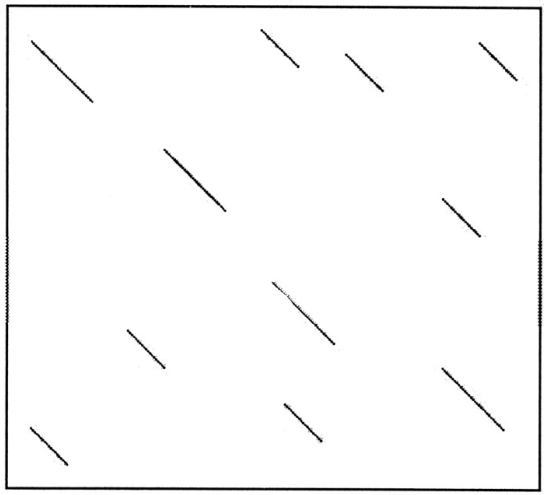
Figure from: Delcher et al., *Nucleic Acids Research* 27, 1999

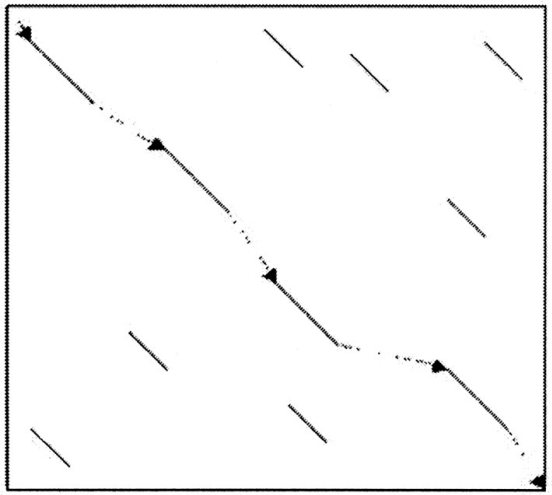
Why the Problem is Challenging

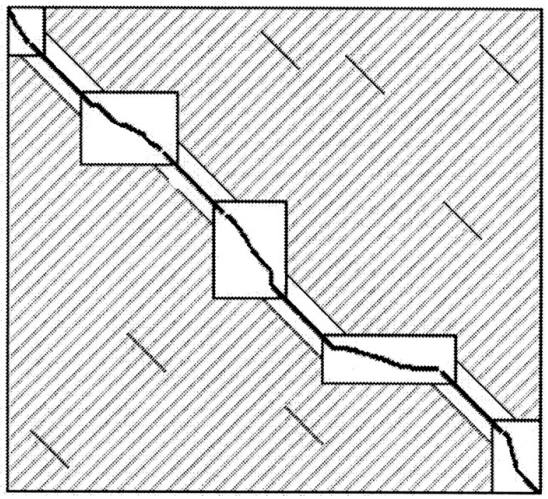
- Sequences too big to make $O(n^2)$ dynamic-programming methods practical
- Long sequences are less likely to be colinear because of *rearrangements*
  - initially we'll assume colinearity
  - we'll consider rearrangements next

General Strategy

Figure from: Brudno et al. *Genome Research*, 2003







1. perform pattern matching to find seeds for global alignment
2. find a good chain of anchors
3. fill in remainder with standard but constrained alignment method

Comparison of Large-Scale Alignment Methods

Method	Pattern matching	Chaining
MUMmer	suffix tree - MUMs	LIS variant
AVID	suffix tree - exact & wobble matches	Smith-Waterman variant
LAGAN	*k*-mer trie, inexact matches	sparse DP

The MUMmer System

Delcher et al., *Nucleic Acids Research*, 1999

Given: genomes A and B

1. find all maximal unique matching subsequences (MUMs)
2. extract the longest possible set of matches that occur in the same order in both genomes
3. close the gaps

Step 1: Finding Seeds in MUMmer

- *Maximal unique match*:
  - occurs exactly once in both genomes A and B
  - not contained in any longer MUM

- Key insight: a significantly long MUM is certain to be part of the global alignment

Suffix Trees

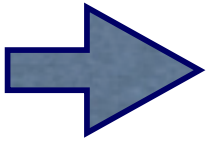
- Substring problem:
  - given text S of length m
  - preprocess S in $O(m)$ time
  - such that, given query string Q of length n, find occurrence (if any) of Q in S in $O(n)$ time

- Suffix trees solve this problem and others

Suffix Tree Definition

- A suffix tree T for a string S of length m is a tree with the following properties:
  - *rooted* and *directed*
  - m leaves, labeled 1 to m
  - each edge labeled by a substring of S
  - concatenation of edge labels on path from root to leaf i is suffix i of S (we will denote this by $S_{i...m}$)
  - each internal non-root node has at least two children
  - edges out of a node must begin with different characters

key property



Suffixes

S = “banana$”

suffixes of S

$ (special character)

a$

na$

ana$

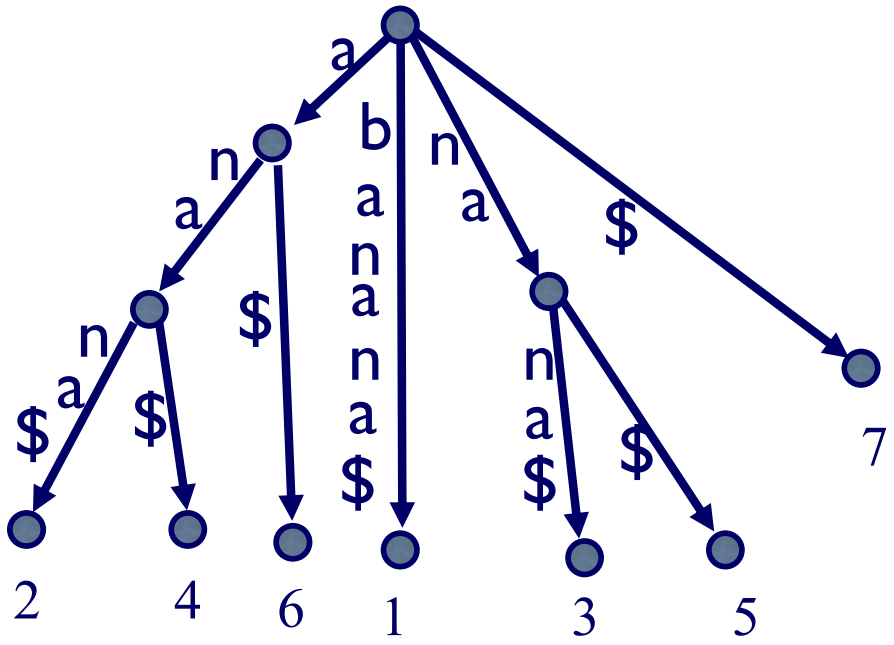
nana$

anana$

banana$

Suffix Tree Example

- S = "banana$"
- Add '$' to end so that suffix tree exists (no suffix is a prefix of another suffix)



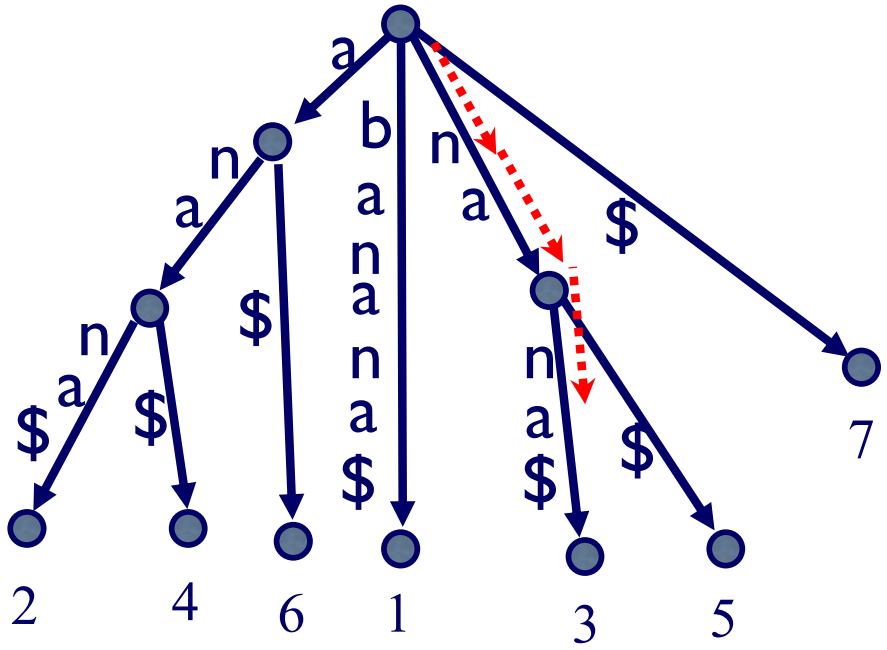
Solving the Substring Problem

- Assume we have suffix tree T and query string Q
- FindMatch(Q, T):
  - follow (unique) path down from root of T according to characters in Q
  - if all of Q is found to be a prefix of such a path

    return label of some leaf below this path
  - else, return no match found

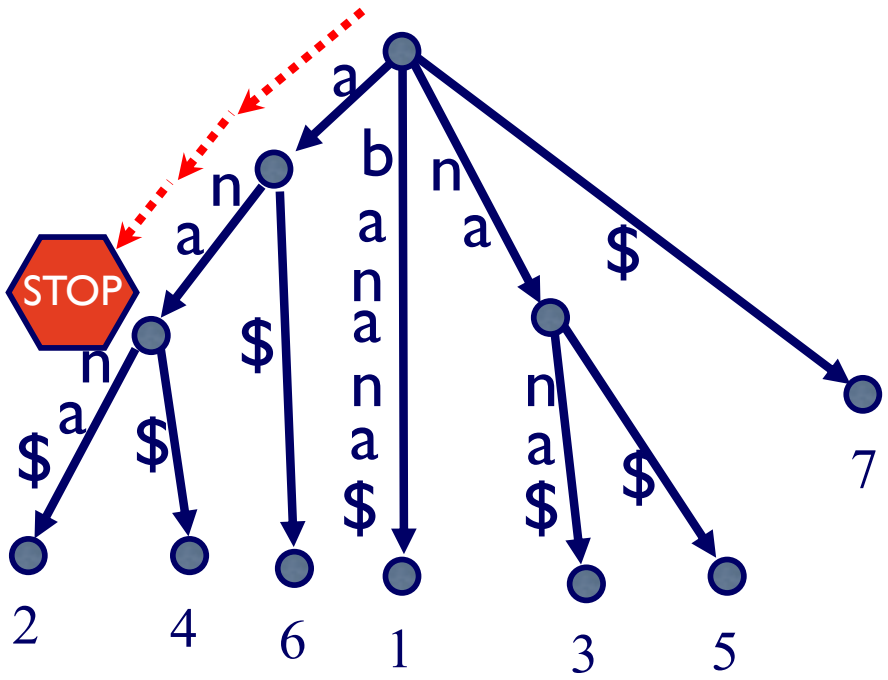
Solving the Substring Problem

Q = nan



return 3

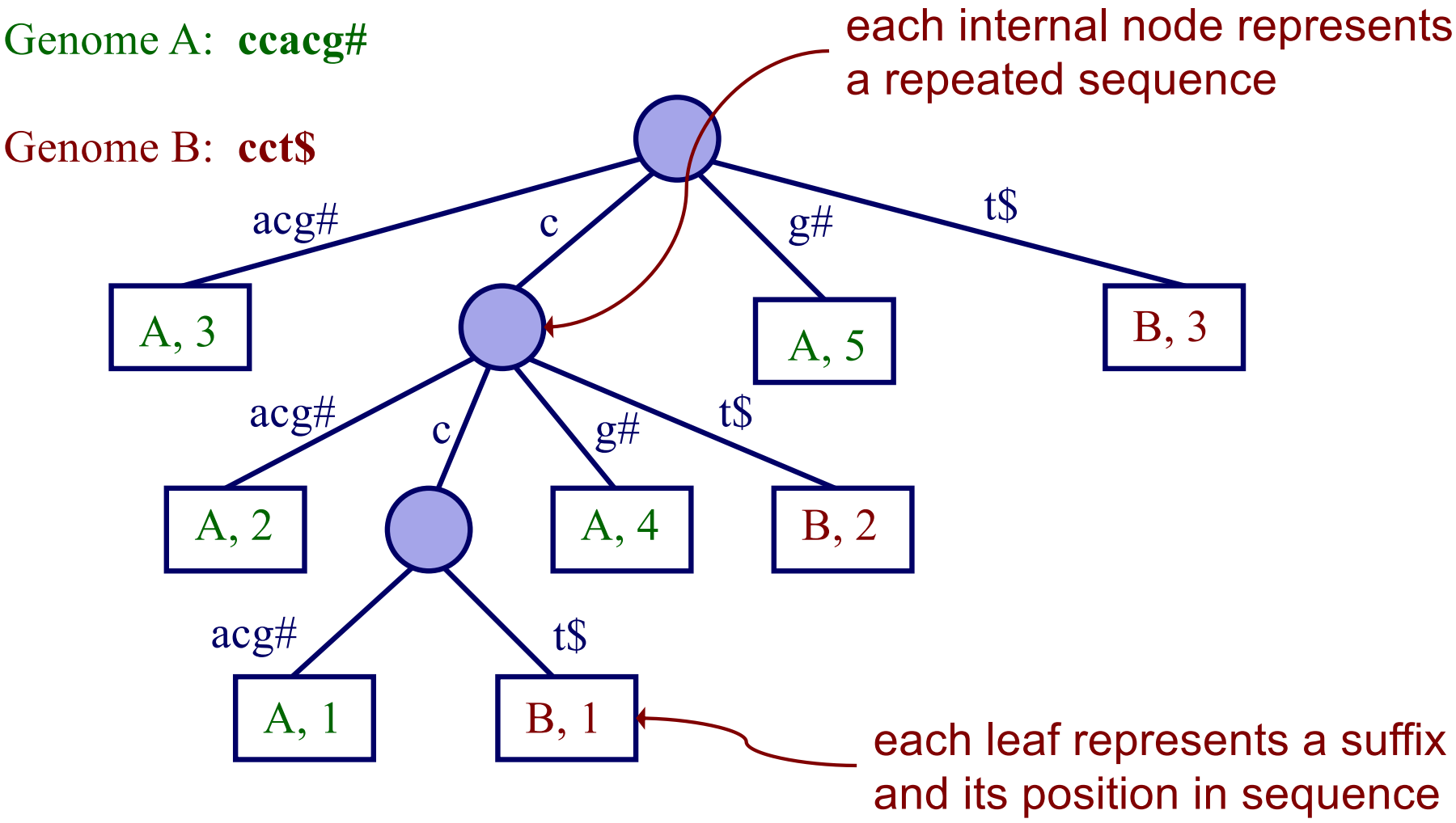
Q = anab



return no match found

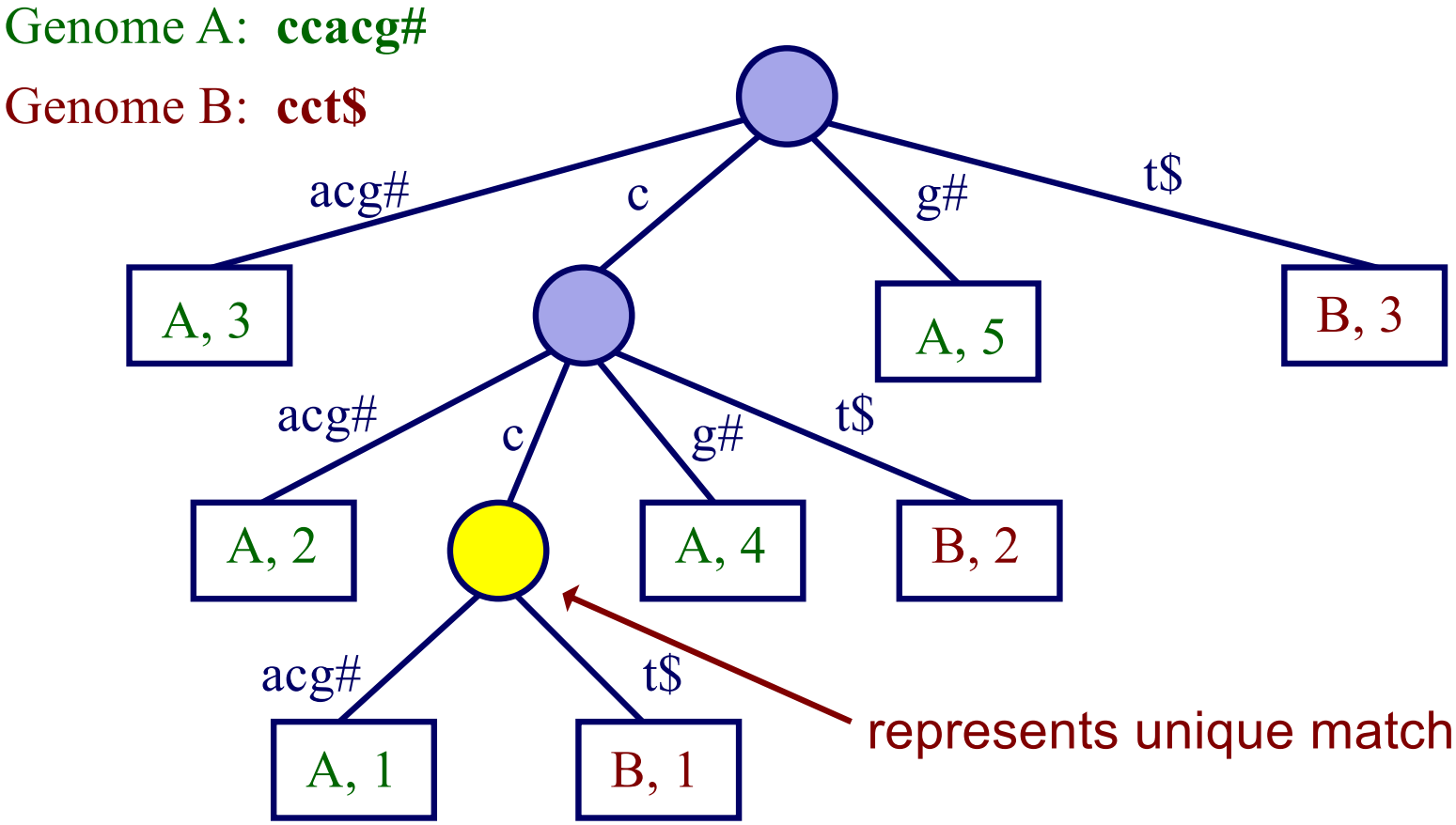
MUMs and *Generalized* Suffix Trees

- Build one suffix tree for both genomes A and B
- Label each leaf node with genome it represents



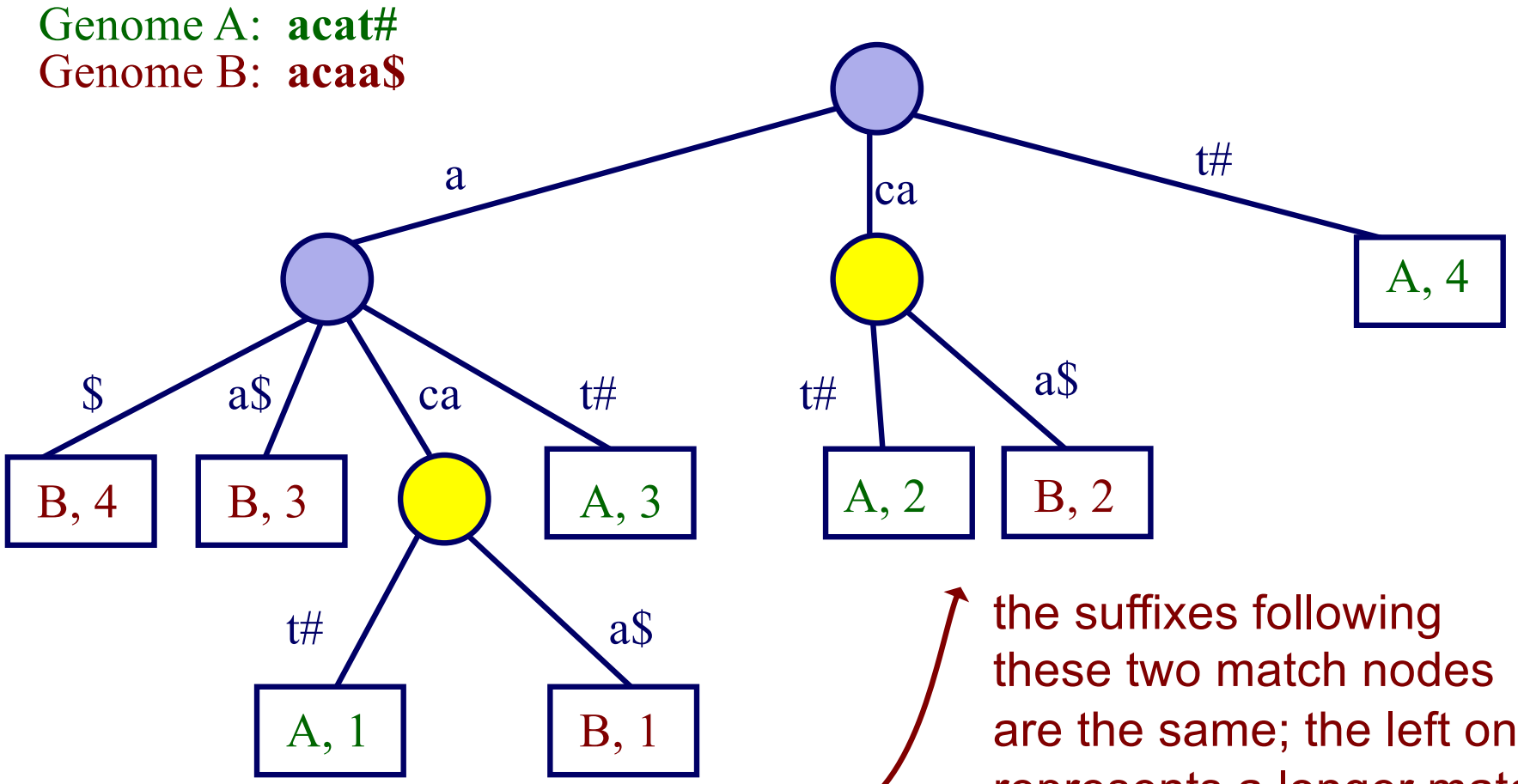
MUMs and Suffix Trees

- Unique match: internal node with 2 children, leaf nodes from different genomes
- But these matches are not necessarily maximal



MUMs and Suffix Trees

- To identify maximal matches, can compare suffixes following unique match nodes



Using Suffix Trees to Find MUMs

- O(n) time to construct suffix tree for both sequences (of lengths ≤ n)
- O(n) time to find MUMs - one scan of the tree (which is O(n) in size)
- O(n) possible MUMs in contrast to O(n^2) possible exact matches

- Main parameter of approach: length of shortest MUM that should be identified (20 – 50 bases)

Step 2: Chaining in MUMmer

- Sort MUMs according to position in genome A
- Solve variation of *Longest Increasing Subsequence* (LIS) problem to find sequences in ascending order in both genomes

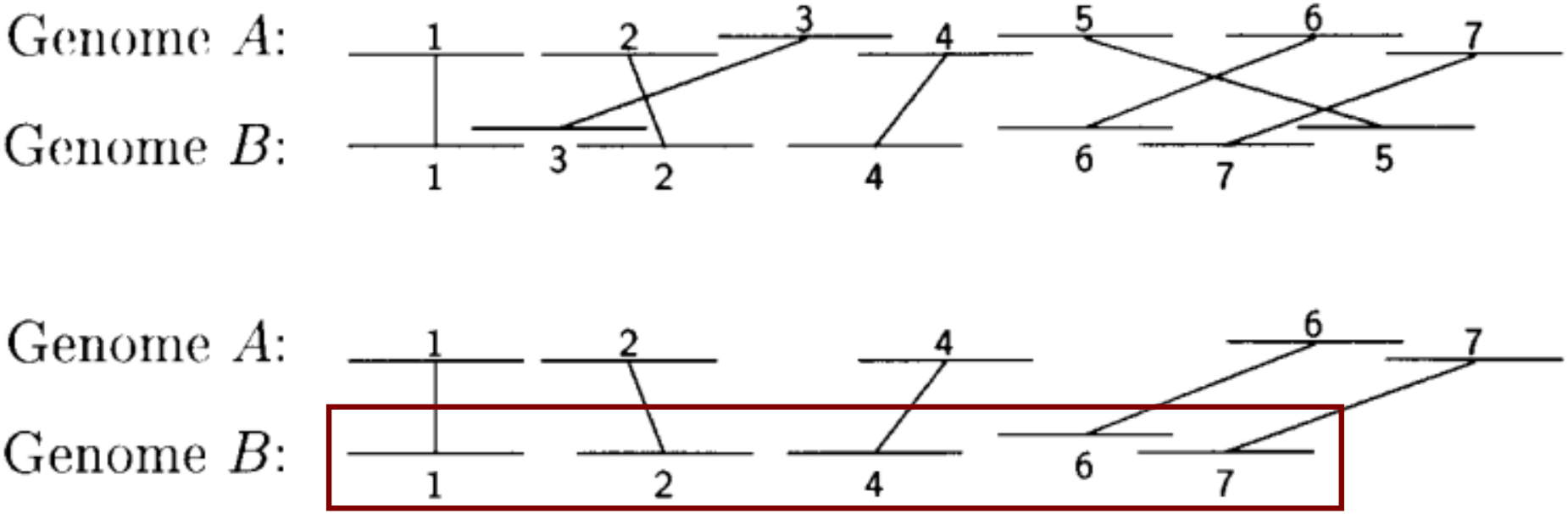
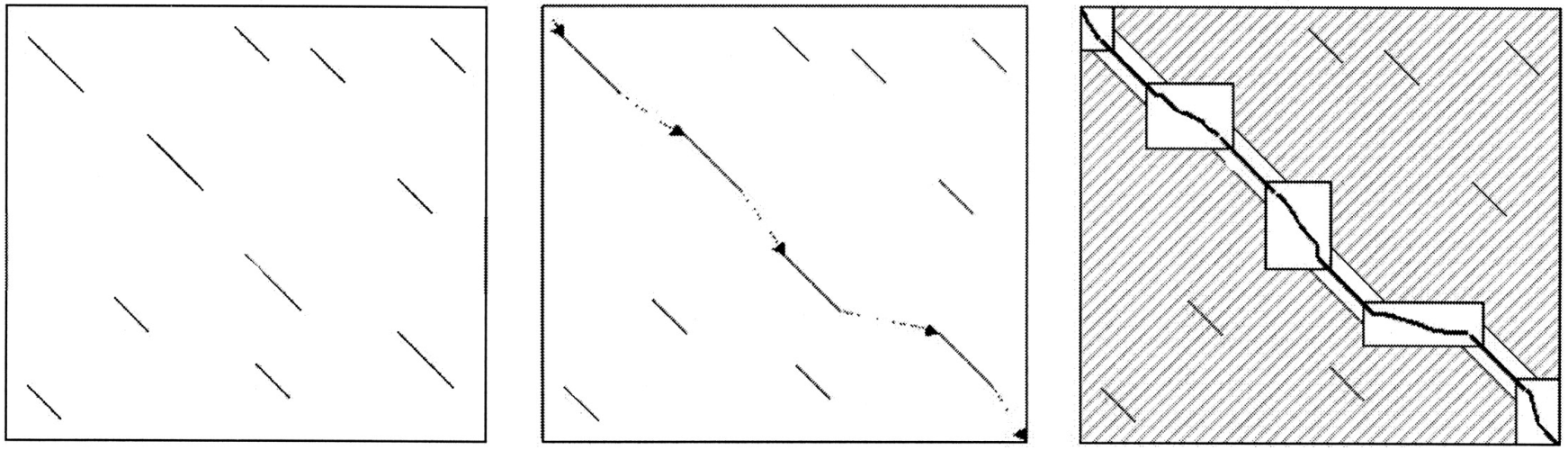


Figure from: Delcher et al., *Nucleic Acids Research* 27, 1999

Finding Longest Subsequence

- Unlike ordinary LIS problems, MUMmer takes into account
  - lengths of sequences represented by MUMs
  - overlaps
- Requires $O(k \log k)$ time where k is number of MUMs

Recall: Three Main Steps of Large-Scale Alignment



Brudno et al. *Genome Research*, 2003

General

1. Pattern matching to find seeds for global alignment
2. Find a good chain of anchors
3. Fill in with standard but constrained alignment

MUMmer

1. Suffix trees to obtain MUMs
2. LIS to find colinear MUMs
3. Smith-Waterman and recursive MUMmer for gap filling

Types of Gaps in a MUMmer Alignment

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
           ^          ^                          ^
```

Figure from: Delcher et al., *Nucleic Acids Research* 27, 1999

Step 3: Close the Gaps

- SNPs:
  - between MUMs: trivial to detect
  - otherwise: handle like repeats
- Insertions
  - simple insertions: trivial to detect
  - transpositions (subsequences that were deleted from one location and inserted elsewhere): look for out-of-sequence MUMs

Step 3: Close the Gaps

- Polymorphic regions
  - short ones: align them with dynamic programming method
  - long ones: call MUMmer recursively with reduced minimum MUM length
- Repeats
  - detected by overlapping MUMs

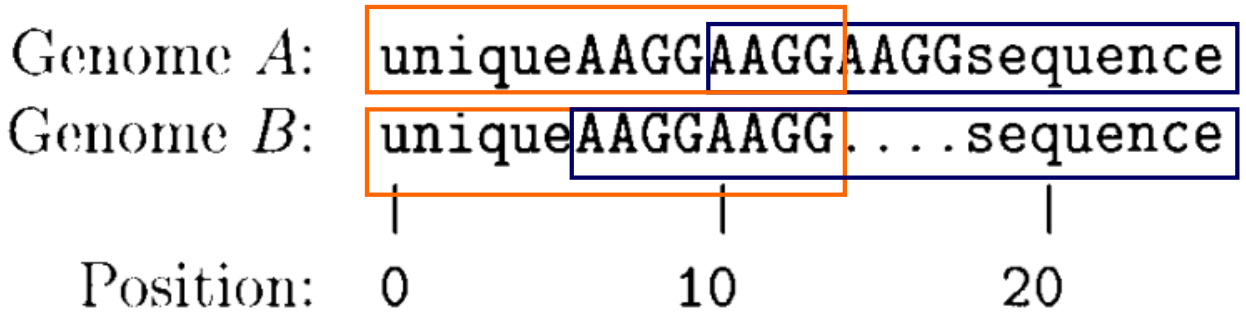


Figure from: Delcher et al. Nucleic Acids Research 27, 1999

MUMmer Performance

FASTA on 1000 base pair segments

MUMmer

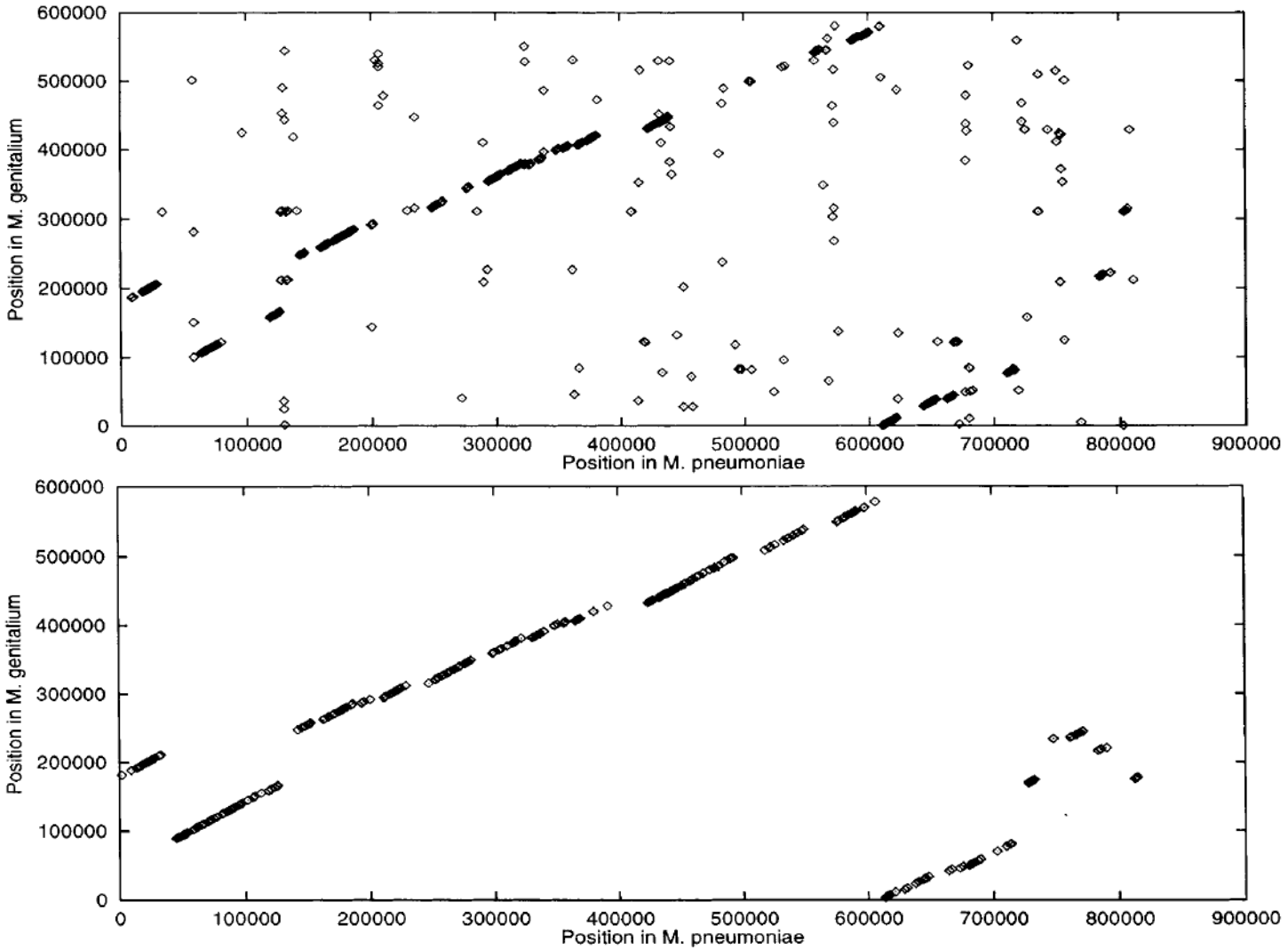


Figure from: Delcher et al. Nucleic Acids Research 27, 1999

MUMmer Performance

- *Mycoplasma* test case
- Suffix tree: 6.5s
- LIS: 0.02s
- Smith-Waterman: 116s

- FASTA baseline: many hours

DEC Alpha 4100

Centre for Computing History

Longevity of MUMmer

- Antimicrobial Resistance Identification By Assembly (ARIBA)
- Identify antimicrobial resistance genes from Illumina reads

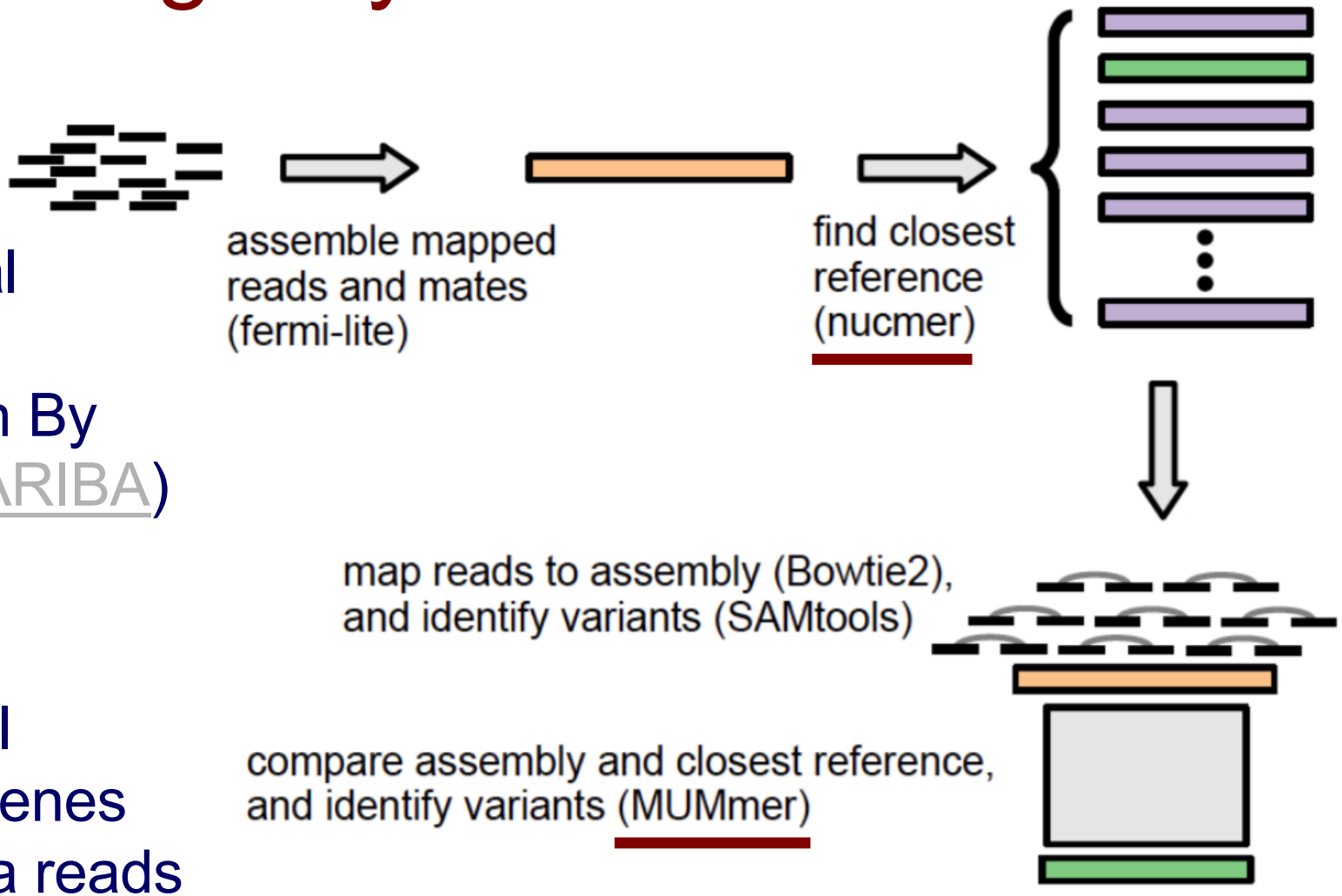


Figure from: Hunt et al. bioRxiv 2017

Longevity of MUMmer

- Whole genome alignment still an active area of research
  - Jain et al. 2018 (Mashmap2): “we were able to map an error-corrected whole-genome NA12878 human assembly to the hg38 human reference genome in about **one minute total execution time** and **< 4 GB memory** using 8 CPU threads”
  - Uses MUMmer as ground truth in evaluation

Limitations of MUMmer

- MUMs are perfect matches, typically ≥ 20-50 base pairs
- Evolutionarily distant may not have sufficient MUMs to anchor global alignment
- How can we tolerate minor variation in the seeds?

More recent developments

- MUMmer4 uses a suffix *array* data structure instead
  - requires less space than a suffix tree (constant factors are smaller)
- Compressed data structures related to suffix arrays are in widespread use
  - Burrows-Wheeler transform (BWT)
  - Ferragina–Manzini index (FM-index)
  - Most commonly used in read mapping applications (e.g., Bowtie, BWA)