

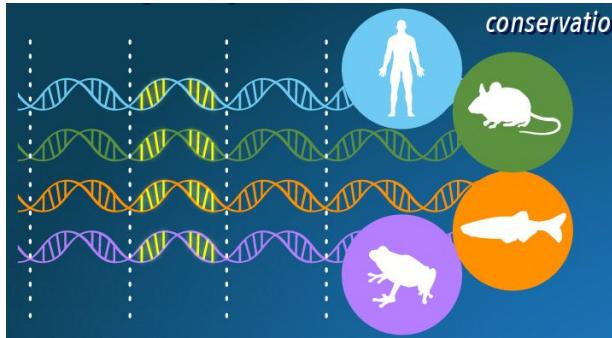
Couple of things...

1. **The 8th ISCB Wikipedia Competition** <https://en.wikipedia.org/wiki/Wikipedia:ISCB-WP8>
 - a. Logan DW, Sandal M, Gardner PP, Manske M, Bateman A (2010). "Ten simple rules for editing Wikipedia". PLoS Comput. Biol. 6 (9). doi:10.1371/journal.pcbi.1000941.
 - b. <https://en.wikipedia.org/wiki/User:Rockpocket/Training>
2. **Bioinformatics Contest 2019** <https://bioinf.me/en/contest>
3. **DREAM Challenges** <http://dreamchallenges.org/>
 - a. Drug-kinase binding prediction
 - b. Tumor deconvolution
4. **ENCODE Imputation Challenge** <https://www.synapse.org/#!Synapse:syn17083203/wiki/587192>

Lecture 8-9: Comparative genomics

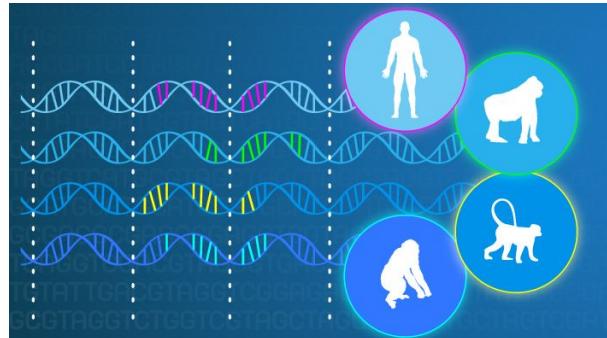
- Whole genome alignment
 - MUMmer & Suffix trees
- Gene/species trees
 - Phylogenetic trees
 - Gene orthology & functional analysis

Comparative genomics



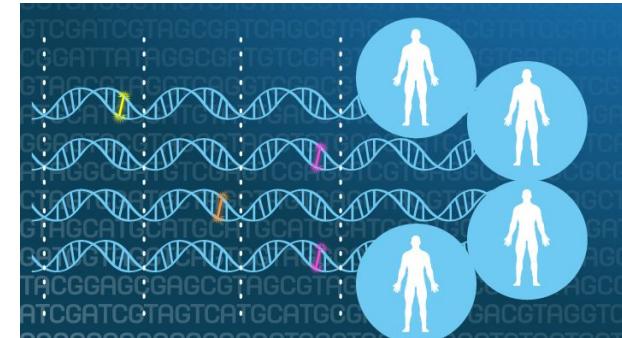
Common features of different organisms

- Conserved DNA.
- Very large distances: gene order & regulatory seqs not conserved.
- Moderate distances: Functional vs. nonfunctional based on negative/purifying selection.



Distinctive features of closely-related species

- Unique genomic elements.



Genetic differences within one species

- Genetic variants with a role in a trait/disease.

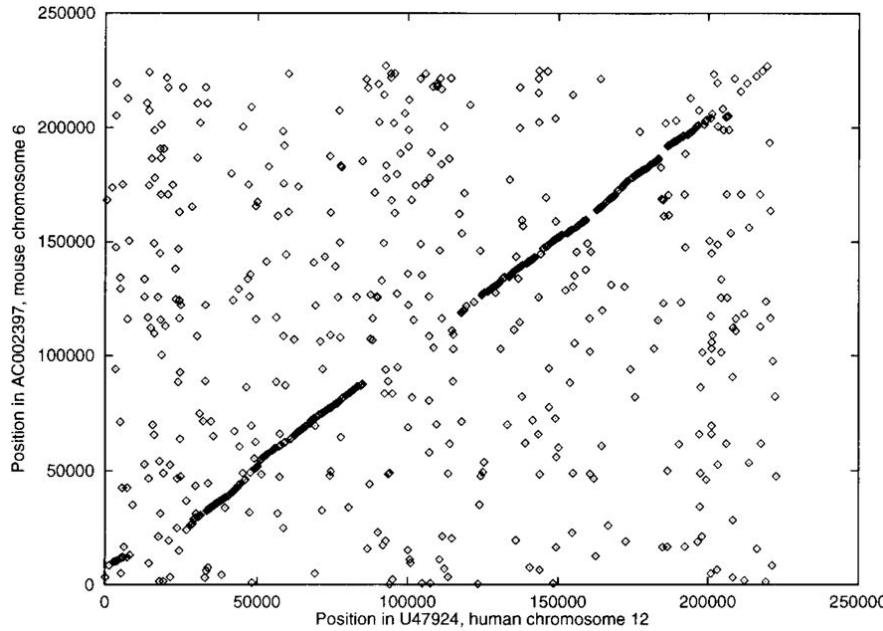
Whole-genome alignment

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.



Whole-genome alignment

Time & space constraints:

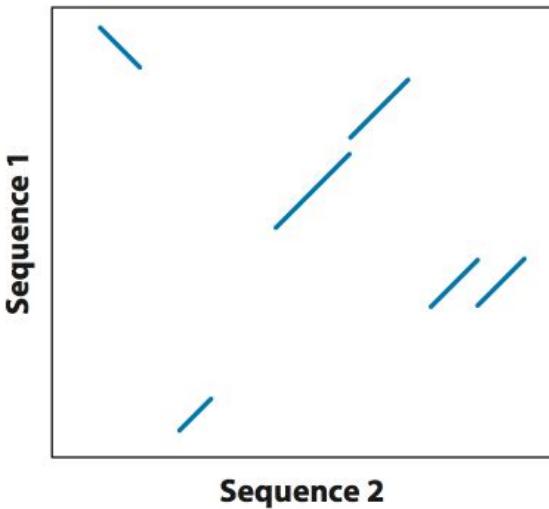
- Traditional DP algorithms require $O(nm)$ time and space (n and m are the lengths of the two sequences).
- As n and m grow to genome scale, too expensive to solve in practice.

Genomic rearrangements:

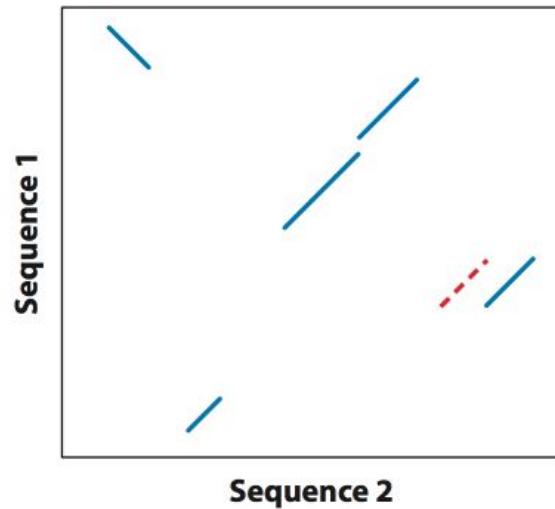
- SW & NW → alignments that have fixed order & orientation.
 - Insertions, deletions, and substitutions are the only allowed edit operations (OK for short or well-conserved sequences, like genes).
- Over large evolutionary distances and within a sufficiently large window, genomes almost always contain more complex rearrangements with respect to each other
 - Inversions, transpositions, and duplications all cause breaks in order and orientation that cannot be captured under constant order and orientation.

Whole-genome alignment

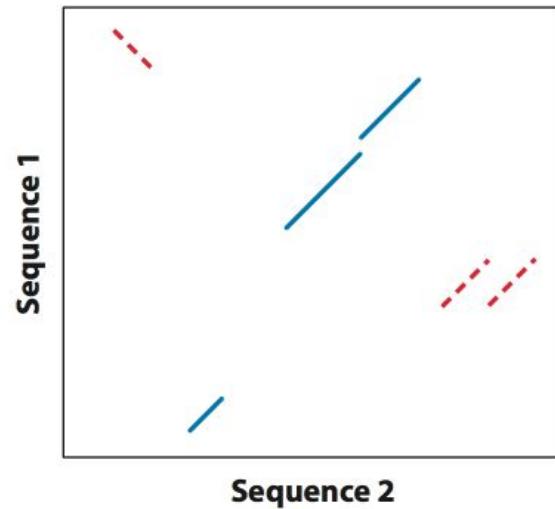
a True genome alignment



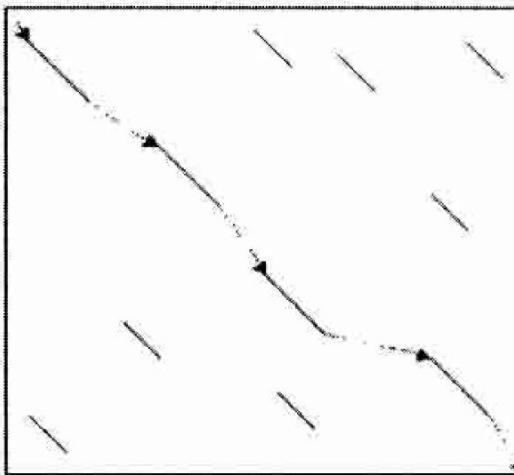
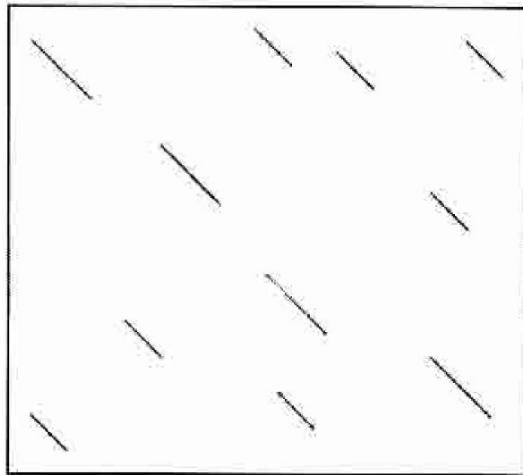
b Single-copy



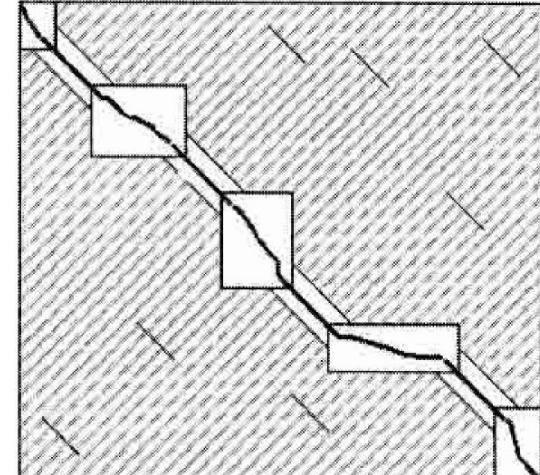
c Single order-and-orientation (global alignment)



Whole-genome alignment



1. Perform pattern matching to find seeds for global alignment.



2. Find a good chain of anchors.

3. Fill in the gaps with traditional but constrained alignment methods.

Alignment of Long Sequences

BMI/CS 776

www.biostat.wisc.edu/bmi776/

Spring 2018

Anthony Gitter

gitter@biostat.wisc.edu

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

Genome A : tcgatcGACGATCGCGGCCGTAGATCGAATAACGAGAGAGCATAAcgactta

Genome B : gcattaGACGATCGCGGCCGTAGATCGAATAACGAGAGAGCATAAtccagag



- 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

Suffixes

$S = \text{"banana\$"}$

suffixes of S

$\$$ (special character)

a\$

na\$

ana\$

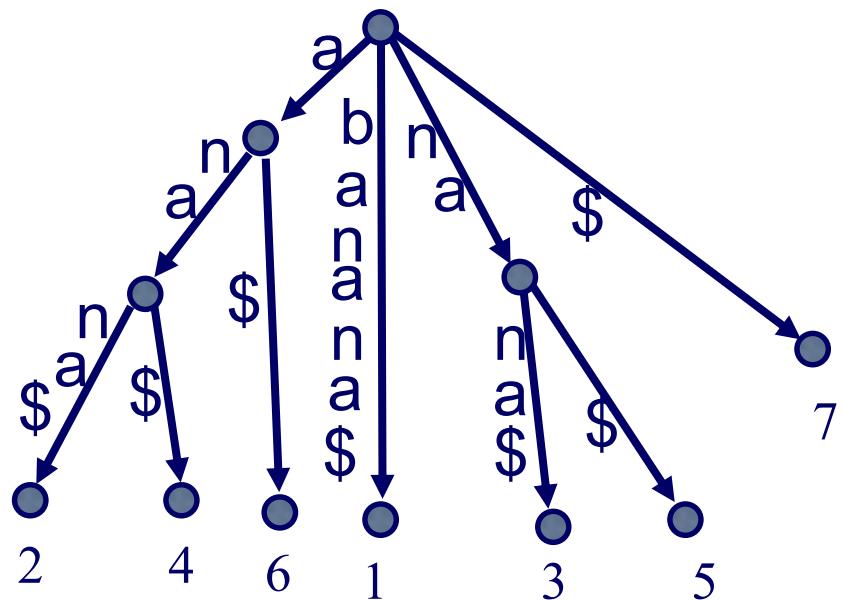
nana\$

anana\$

banana\$

Suffix Tree Example

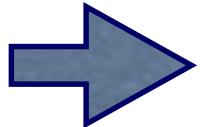
- $S = \text{"banana\$"}$
- Add ‘\$’ to end so that suffix tree exists (no suffix is a prefix of another suffix)



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 $S_{i\dots m}$)
 - each internal non-root node has at least two children
 - edges out of a node must begin with different characters

key property

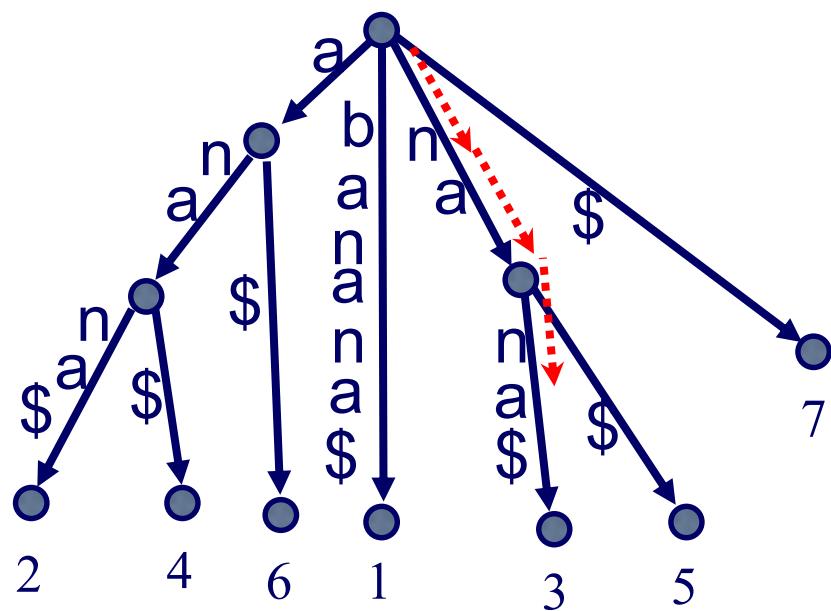


Solving the Substring Problem

- Assume we have suffix tree T and query string Q
- $\text{FindMatch}(Q, T)$:
 - 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

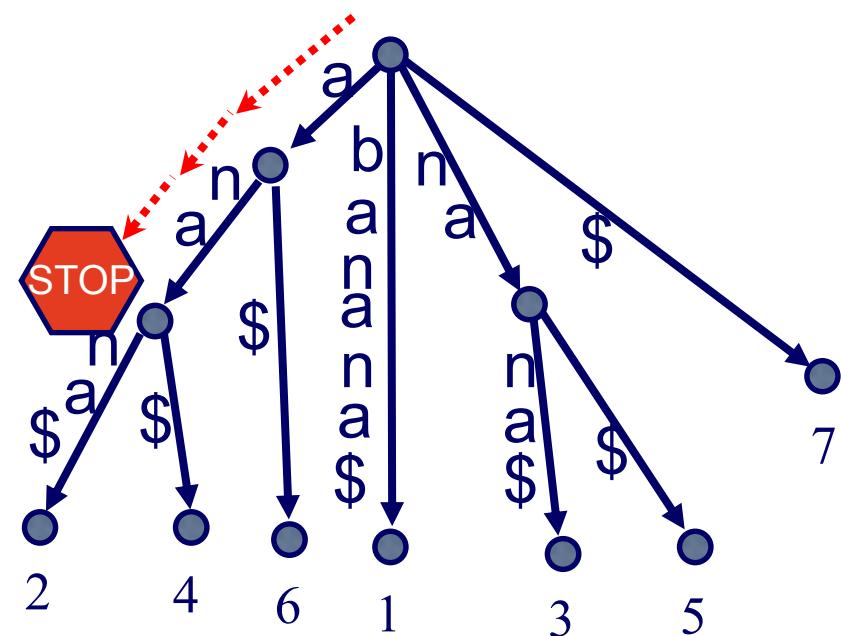
Solving the Substring Problem

$Q = \text{nan}$



return 3

$Q = \text{anab}$



return no match found

Notes

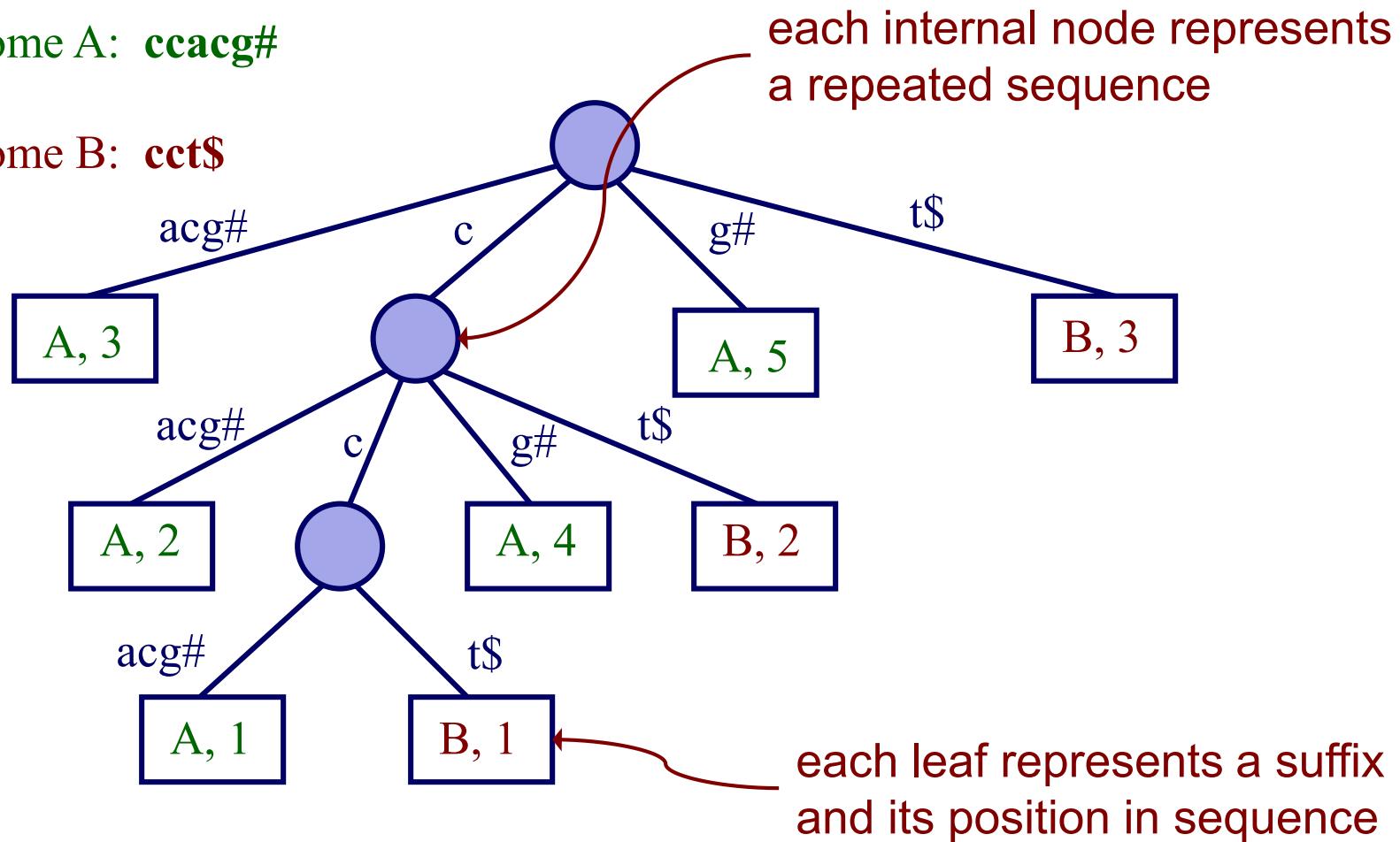
- Example: Build a suffix tree for "aba" & "abaaba"
- Each path from root to leaf represents a suffix; each suffix is represented by some path from root to leaf;
Would this still be the case if we hadn't added \$?
 - Is string Q a substring of S? (Every substring is a prefix of some suffix in the tree!)
 - Is Q a suffix of S?
 - How many times does Q occur in S? (Leaves can be counted with DFS)
 - What is the longest repeated substring of S?
- Make them smaller
 - Coalesce non-branching paths
 - Store T itself in addition to the tree (as (offset, length) pairs)

MUMs and Generalized Suffix Trees

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

Genome A: **ccacg#**

Genome B: **cct\$**

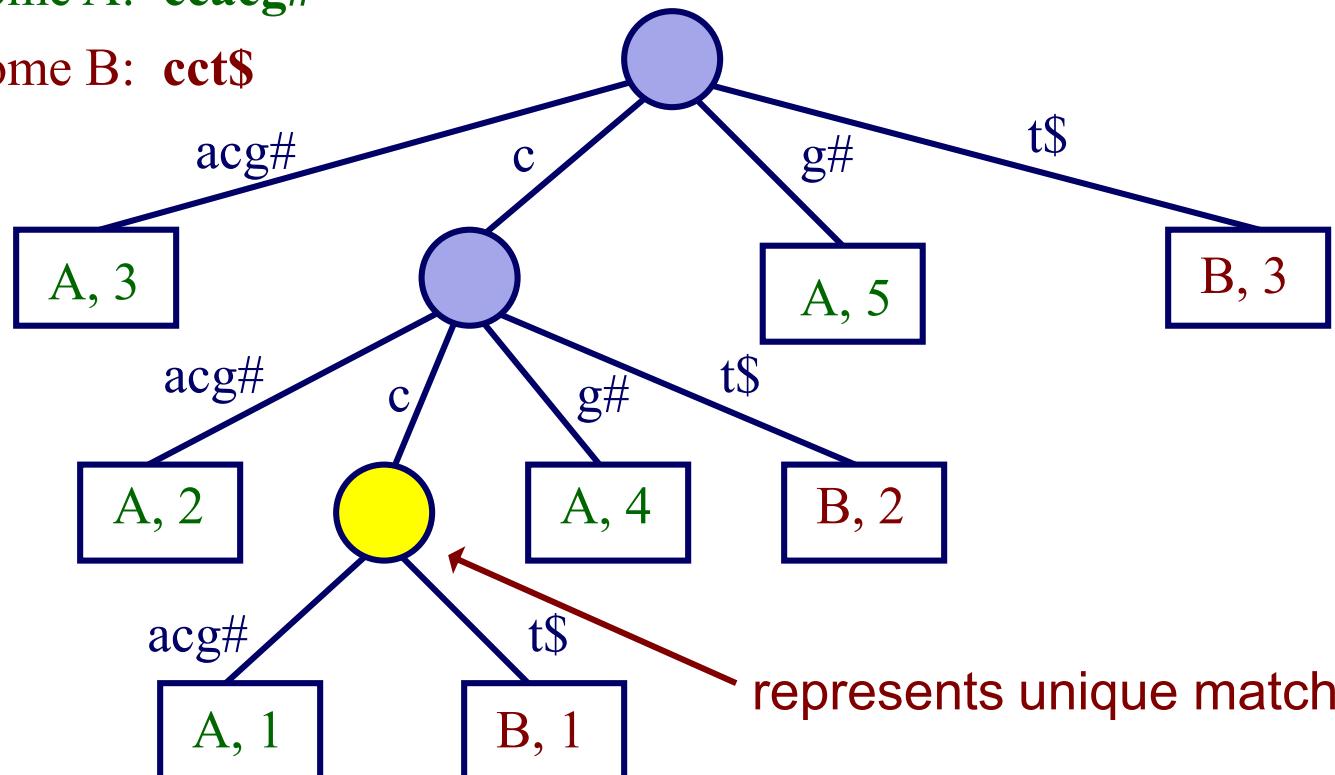


MUMs and Suffix Trees

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

Genome A: **ccacg#**

Genome B: **cct\$**

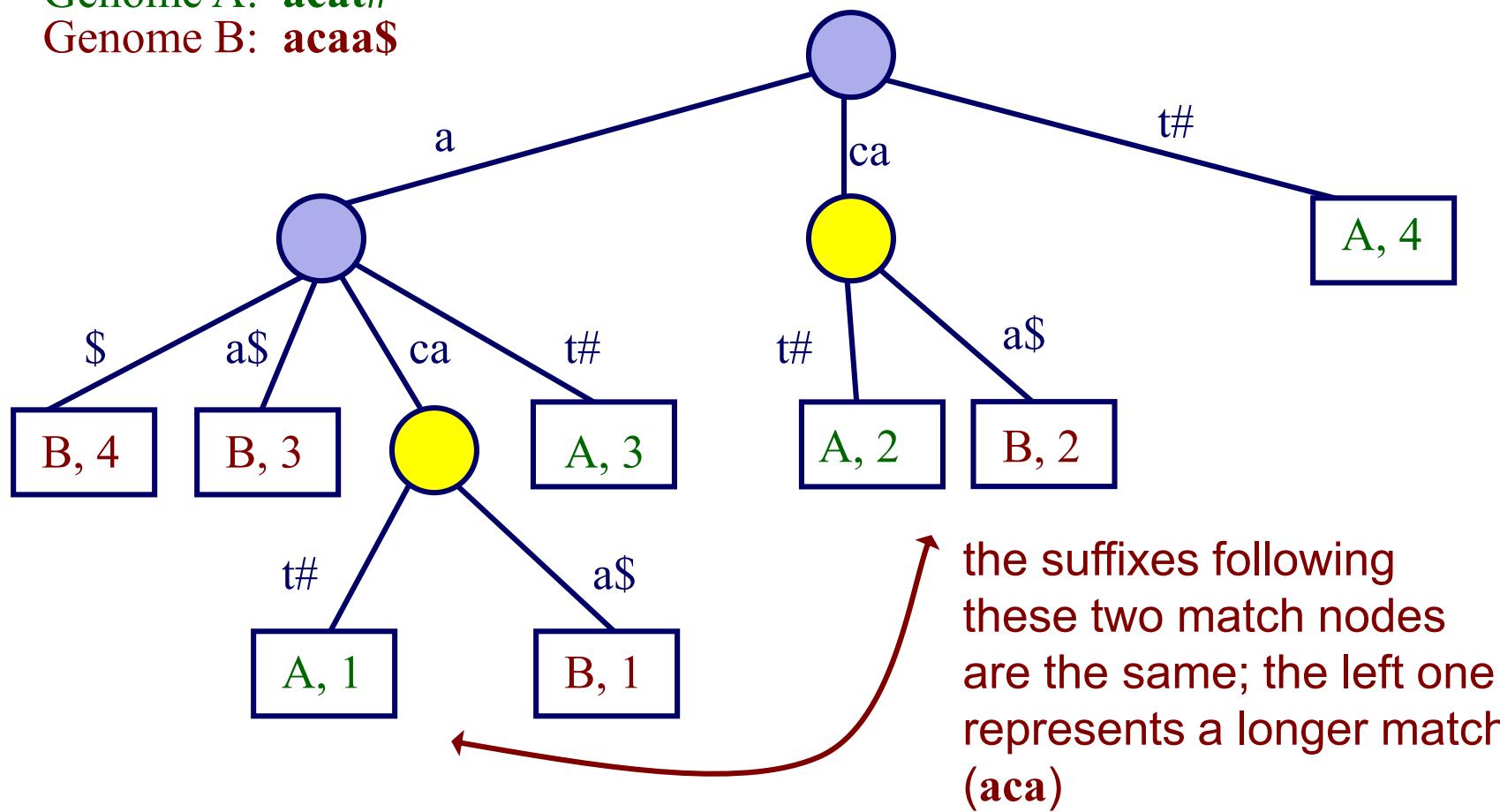


MUMs and Suffix Trees

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

Genome A: acat#

Genome B: acaa\$



Using Suffix Trees to Find MUMs

- $O(n)$ time to construct suffix tree for both sequences (of lengths $\leq n$)
- $O(n)$ 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

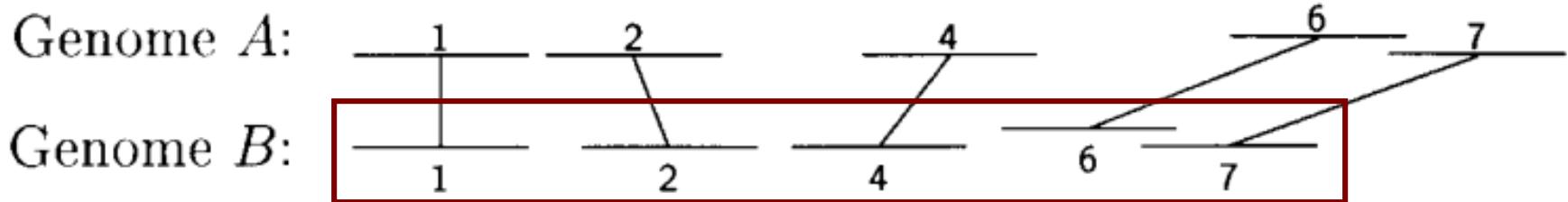
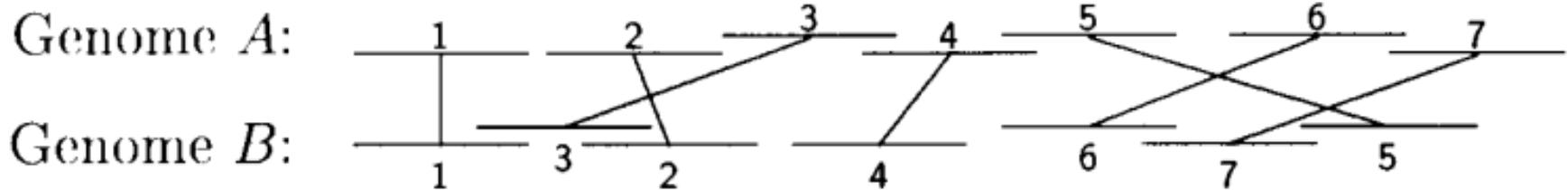


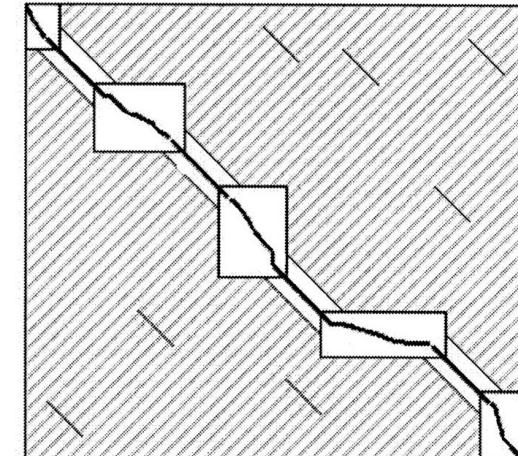
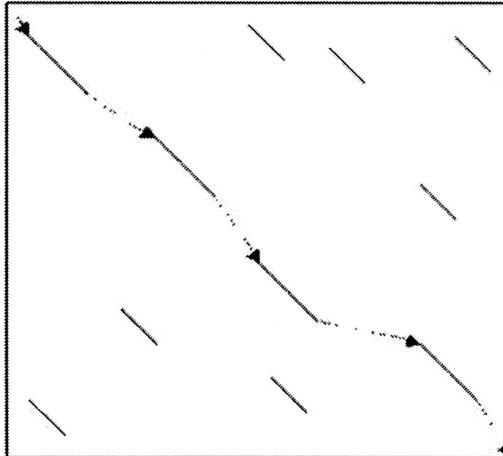
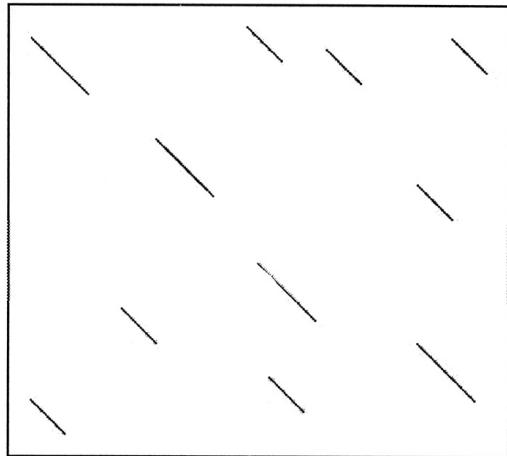
Figure from: Delcher et al., *Nucleic Acids Research* 27, 1999

genome A: 1, 2, 3, 4, 5, 6, 7: [1, 3, 4, 6, 7]
genome B: 1, 3, 2, 4, 7, 5, 6: [1, 2, 4, 5, 6]

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



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: cgtcatggcg^ttcgtcg^ttg

Genome B: cgtcatggcattcg^ttcgtcg^ttg

^

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

Genome A: cggggtaaccgc.....cctggtcggg

Genome B: cggggtaaccgc^tttgc^tcgggtaaccgc^cc^tgg^ttcggg

.....

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

Genome A: ccgcctcgcc^tgg.gctggcgccc^ctc

Genome B: ccgcctcgcc^ag^ttgc^cccg^cccc^ctc

^ ~ ~ ~

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: cTGGGTGGGACAA^ACGTa^aaaaaaaa^aTGGGTGGGACAA^ACGTc

Genome B: aTGGGTGGGGC^aCGT^tggggggggggTGGGTGGGACAA^ACGT^a

^

^

^

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

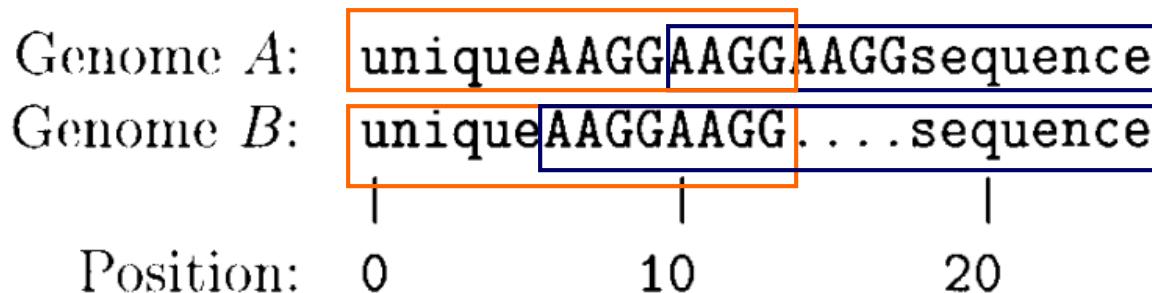
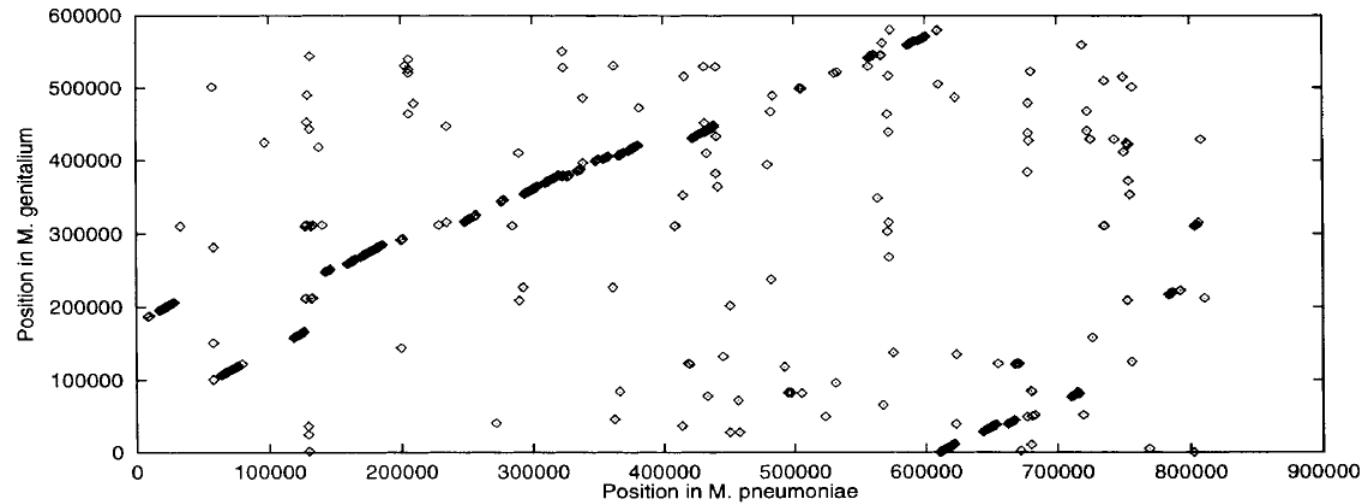


Figure from: Delcher et al. Nucleic Acids Research 27, 1999

MUMmer Performance

FASTA on
1000 base
pair segments



MUMmer

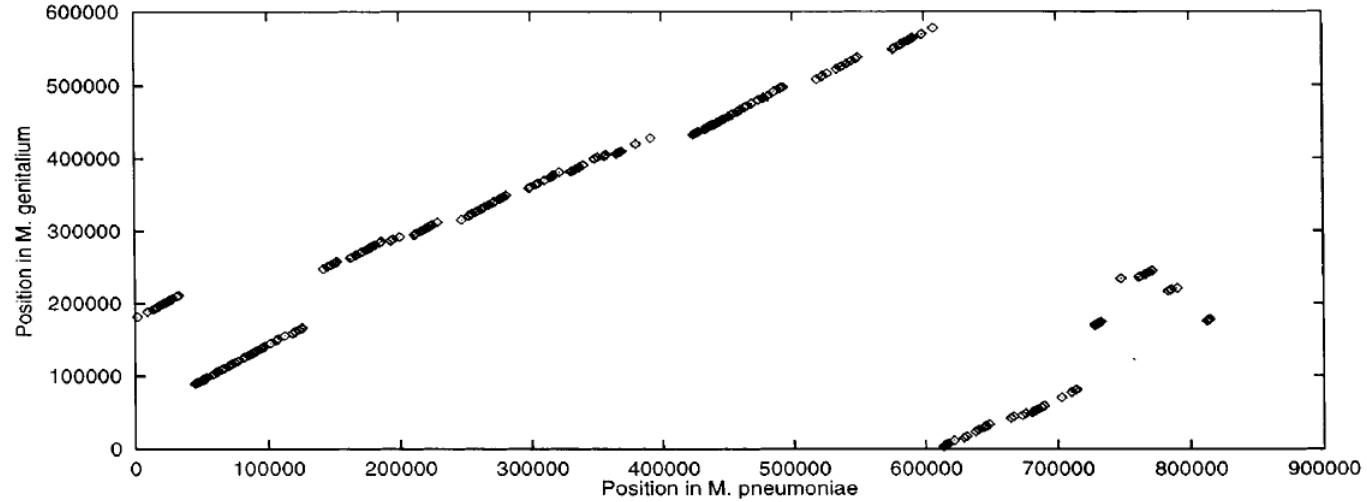


Figure from: Delcher et al. Nucleic Acids Research 27, 1999

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

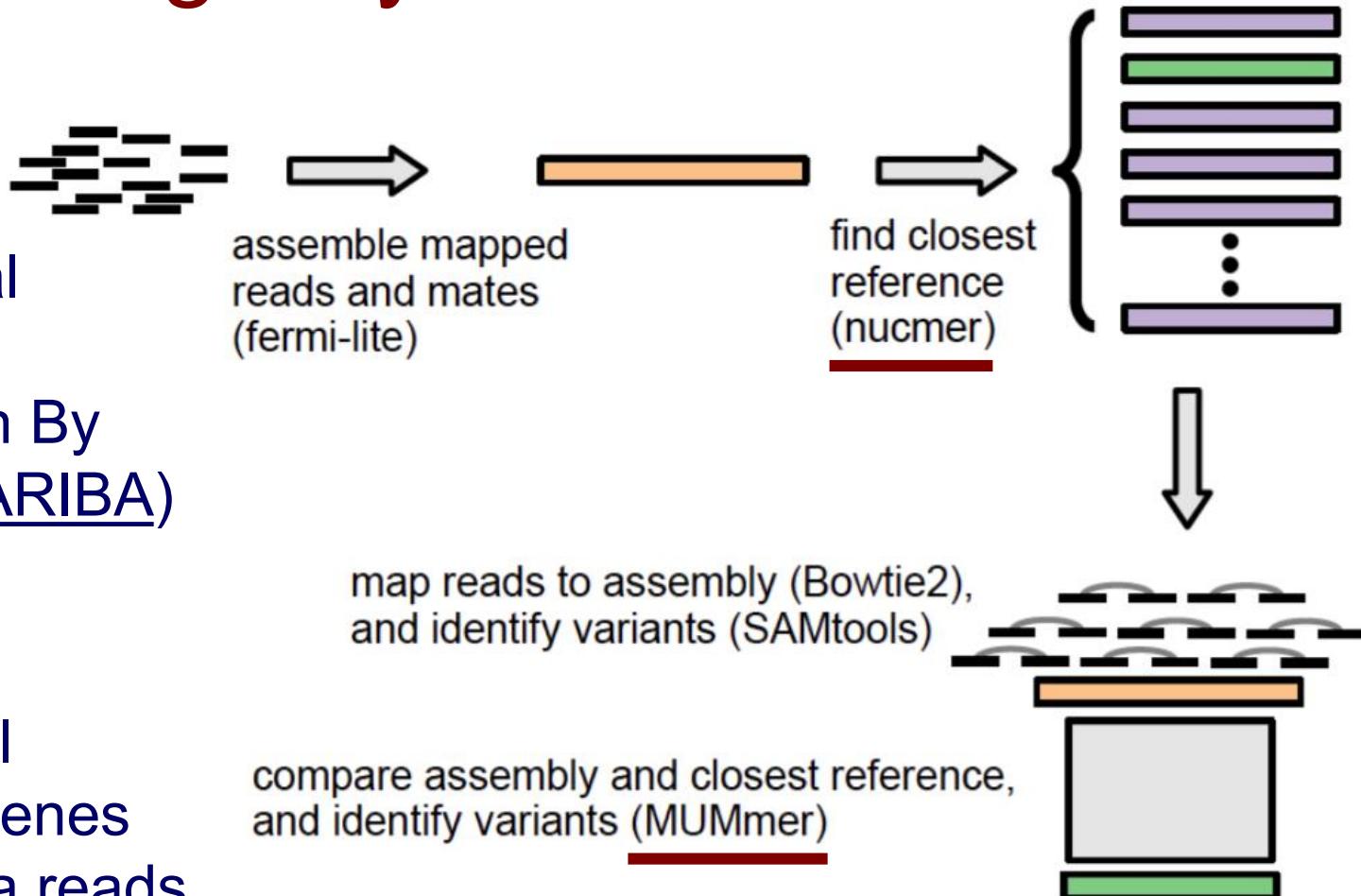


Figure from: Hunt et al. bioRxiv 2017

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 $\geq 20\text{-}50$ base pairs
- Evolutionarily distant may not have sufficient MUMs to anchor global alignment
- How can we tolerate minor variation in the seeds?

Paper report & discussion

	Biological / conceptual	Technological / experimental	Algorithmic / methodological	Hardware / software
MUMer v1				
MUMer v2				
MUMer v3				
MUMer v4				