Alignment of whole genomes using MUMmer

Presentation in Algorithms in Bioinformatics (TÖ111F autumn 2014)

Hannes Pétur Eggertsson

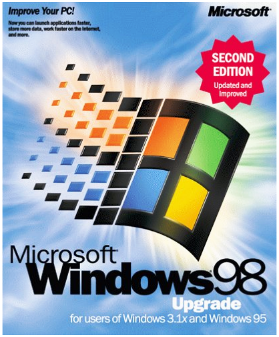
November 18, 2014

Motivation

Let’s go back to 1999... (some things that will remind you of that wonderful time)

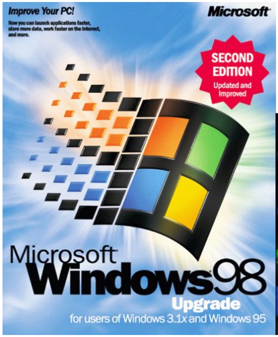
Motivation

Let's go back to 1999... (some things that will remind you of that wonderful time)


Improve Your PC!
Now you can launch applications faster, store more data, work faster on the Internet, and more.
Microsoft
SECOND EDITION
Updated and Improved
Microsoft
Windows98
Upgrade
for users of Windows 3.1x and Windows 95

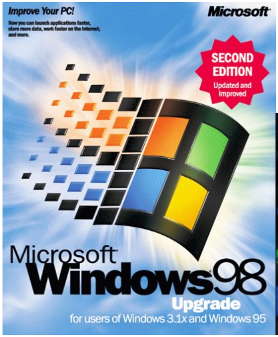
Motivation

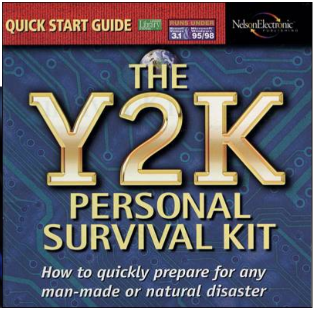
Let's go back to 1999... (some things that will remind you of that wonderful time)


Improve Your PC!
Now you can launch applications faster, store more data, work faster on the Internet, and more.
Microsoft
SECOND EDITION
Updated and Improved
Microsoft Windows98
Upgrade
for users of Windows 3.1x and Windows 95

Motivation

Let’s go back to 1999... (some things that will remind you of that wonderful time)


Improve Your PC!
Now you can launch applications faster, store more data, work faster on the Internet, and more.
Microsoft
SECOND EDITION
Updated and Improved
Microsoft Windows98
Upgrade
for users of Windows 3.1x and Windows 95


QUICK START GUIDE
RUNS UNDER 3.1 95/98
NelsonElectronic
THE Y2K PERSONAL SURVIVAL KIT
How to quickly prepare for any man-made or natural disaster

Motivation

In 1999...

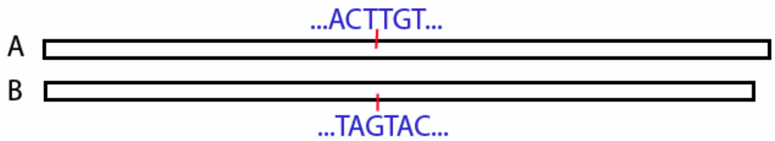
- The number of sequenced genomes was very low but increasing rapidly
- When a new genome is sequenced, one could ask himself:
  - How does this genome align to the other genomes?

Problem:

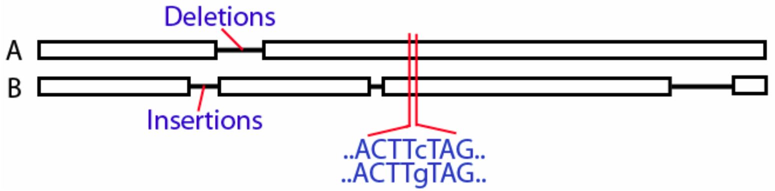
- We had algorithms that were used for single gene sequences (up to 10,000 bp)
- But, they won't work well with whole genomes (can be millions of base pairs or more).
- On the other algorithms they'd either
  - Take up way too much memory or
  - Have unacceptable computational time

Problem description

In Two genomes, A and B. Both could be very large (possibly over 1 Mbp)



Out Align the two genomes using insertions and deletions (or for short, indels) to maximize the matches.



Introduction to MUMmer

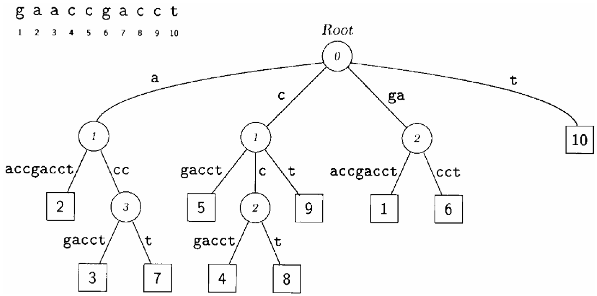
MUMmer:

- Was published in 1999.
- Is a system used to align whole genome sequences.
- Uses suffix trees as a data structure.
- Tries to find large unique chunks of exact matches on both genomes.
  - So it will only work well for two genomes that are similar/related.
- Has been open-source since 2004 (when MUMmer 3 was released).
- Doesn't guarantee the optimal solution, just a good one.
- The algorithm can be split into several steps.

Step 1: Creating a suffix tree

A suffix tree is a compact representation that stores all possible suffixes of an input sequence.

- Square nodes are leaves.
  - Store information about the starting position of the suffix.
- Circular nodes are internal nodes.
  - That means two or more sequences share the same prefix.
  - Store information about the length of the shared prefix.



Creating a suffix tree takes $O(n)$ time and space.

Step 2: MUM decomposition

Let us first define what a MUM is

Definition

A subsequence is a MUM (Maximal Unique Matches) if and only if:

- The subsequence has an exact match on both genomes
- It is not a subsequence of another matched sequence
  - This means the sequence is surrounded be mismatches
- It is unique
  - It appears exactly once in both genomes
- Almost always uses some minimum sequence length.

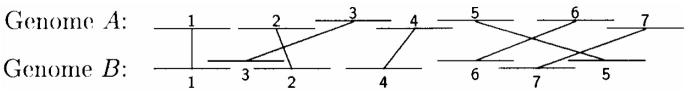
Example

```
tcgatcGACGATCGCGGCCGTAGATCGAATAACGAGAGAGCATAAcgactta
gcattaGACGATCGCGGCCGTAGATCGAATAACGAGAGAGCATAAtccagag
```

We can find these MUMs using the suffix tree (further details later).

Step 3: Sorting the matches found in the MUM alignment

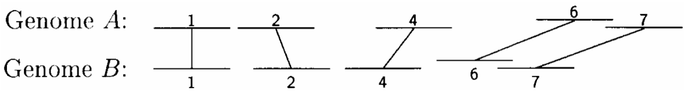
Once we have found the MUMs, we enumerate them like this:



Problem: We cannot align all MUMs because they aren't in the same order in both genomes.

Solution: Align as many MUMs as we can:

- We find the longest increasing sequence for genome B.
  - In this case the sequence is <1,3,2,4,6,7,5>.
- We can do this in $O(K \log K)$ time, where K is the number of MUMs. Since in general $K << N/\log N$ this step takes $O(n)$ time.



Step 4: Closing the gaps

Everything in between the MUMs is called **gaps**.

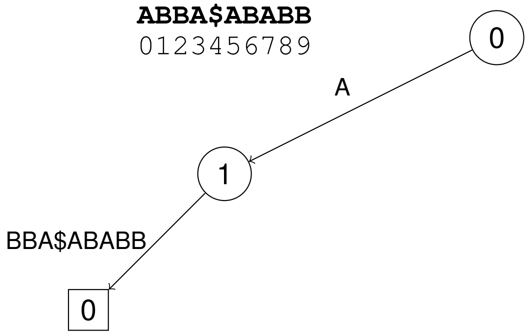
- To find alignment for the gaps we use any alignment algorithm.
- The first version of MUMmer uses the Smith-Waterman alignment algorithm (which takes $O(n^2)$ time).
- If the gaps are too large for the alignment algorithm we need to recursively find new MUMs in that region using a smaller minimum sequence length than before.

Step 5: Output results

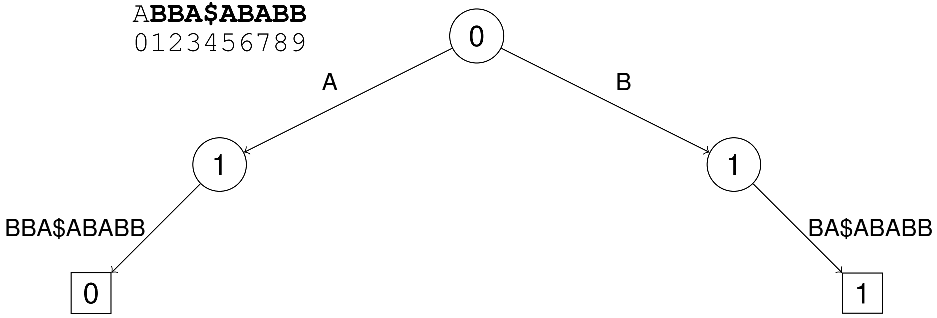
The final alignment is the alignment of MUMs and gaps combined.

- If the two genomes are very similar, then the MUMs sequences will...:
  - ...be long.
  - ...cover most of the genome.
  - ...rarely be a random match. Therefore few or no errors.
  - ...make the algorithm run fast (almost linear time)
- If the genomes are very different, then the MUMs sequences will...:
  - ...be short.
  - ...cover a small part of the genome.
  - ...often be a random match. Therefore many errors.
  - ...make the algorithm run slow.

Creating a suffix tree

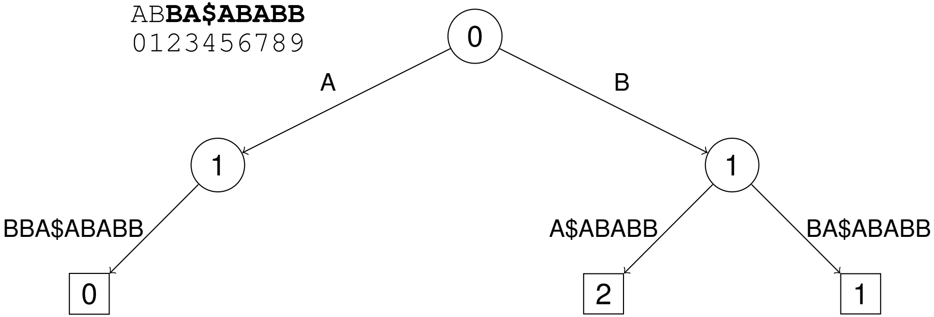


Creating a suffix tree

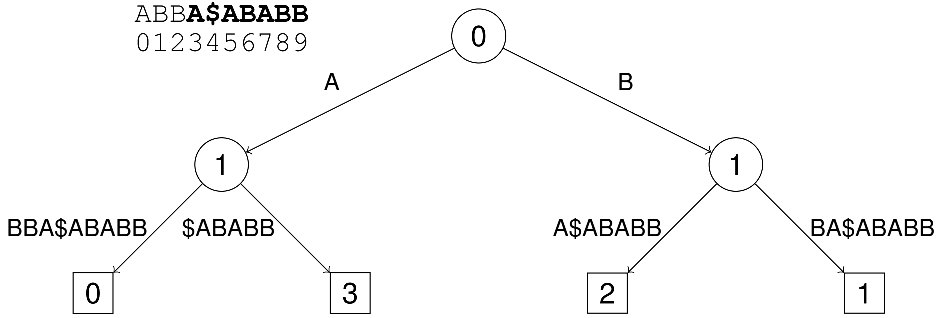


Creating a suffix tree

AB**BA$ABABB**
0123456789

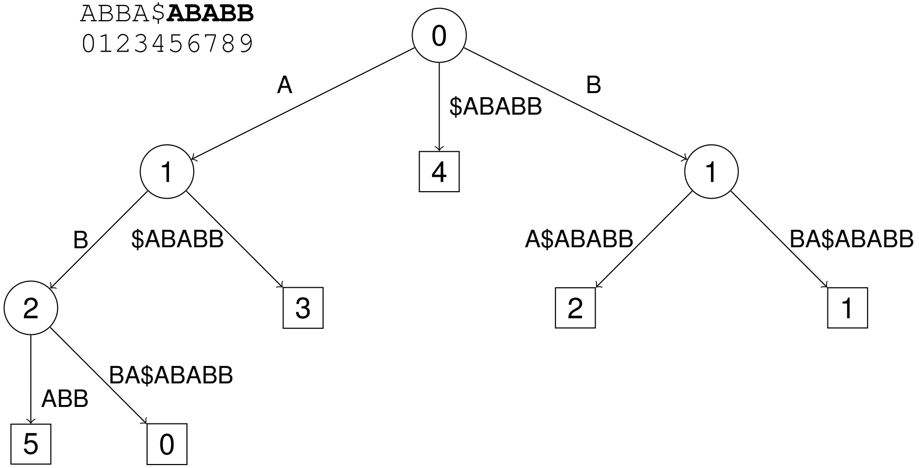


Creating a suffix tree

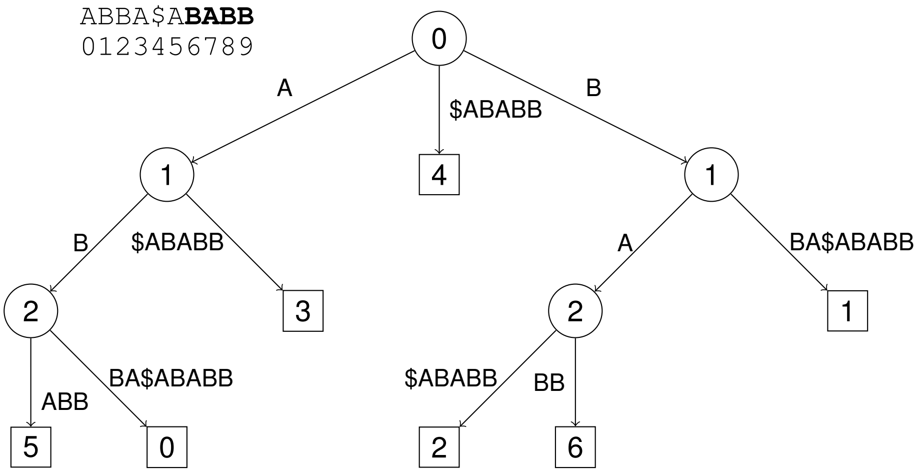


Creating a suffix tree

```
ABBA$**ABABB**
0123456789
```

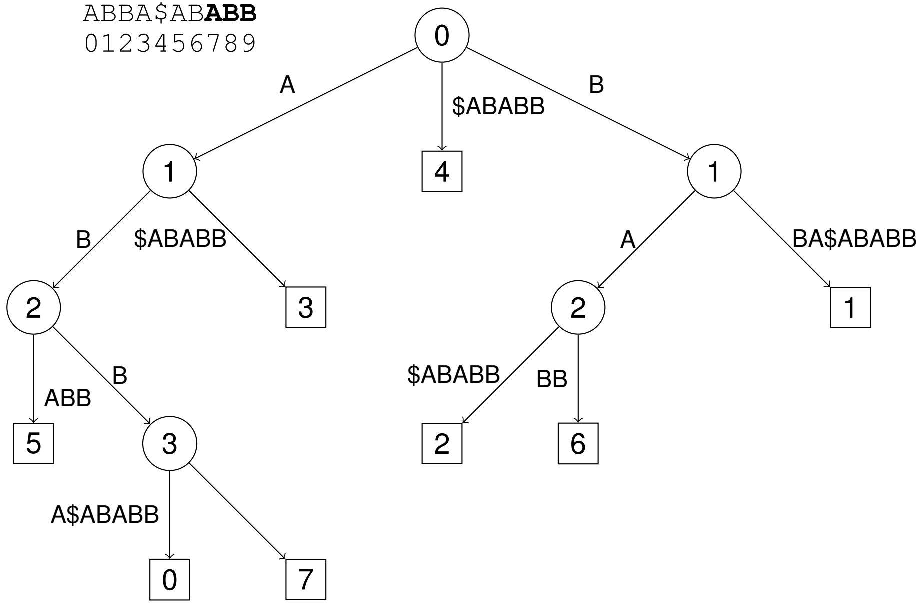


Creating a suffix tree



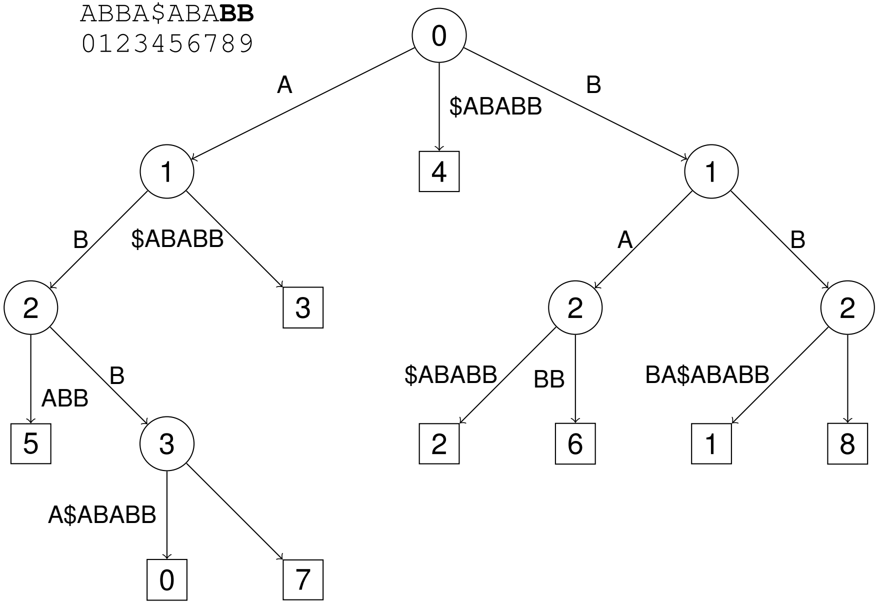
Creating a suffix tree

```
ABBA$AB**ABB**
0123456789
```

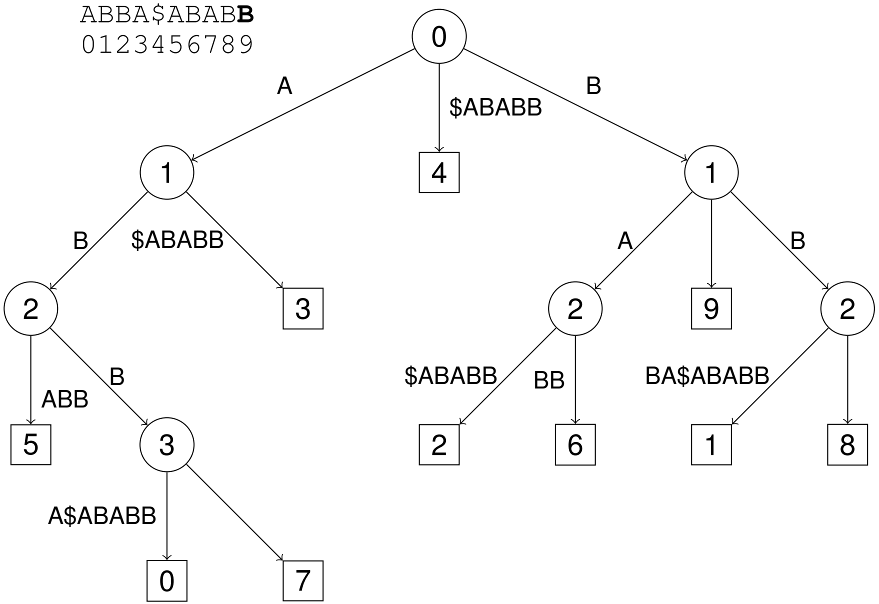


Creating a suffix tree

```
ABBA$ABA**BB**
0123456789
```



Creating a suffix tree



Finding MUMs from suffix tree

Let's recall what condition MUMs had to have:

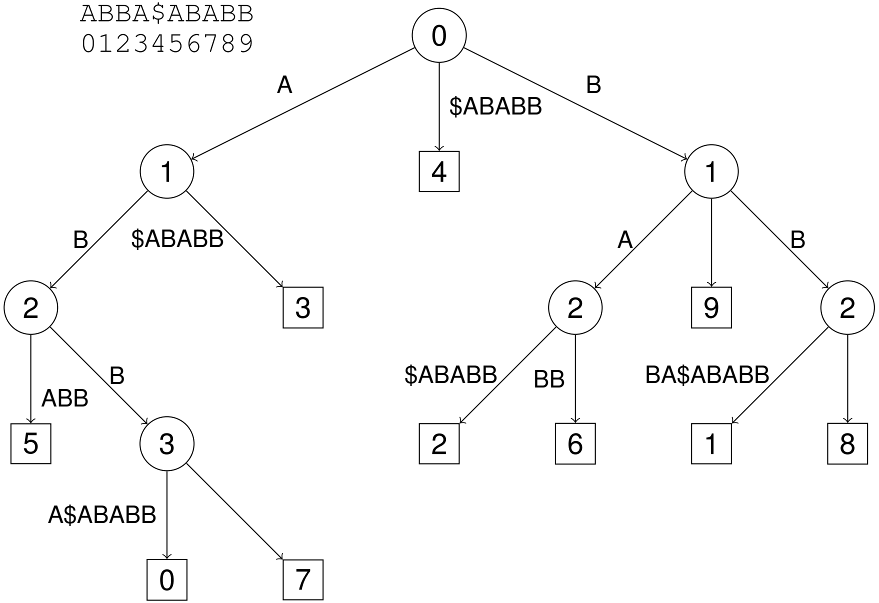
1. Exact matches on both genomes.
2. Surrounded by mismatches.
3. Unique.

We can achieve conditions 1 and 3 by searching the tree for a internal with exactly two leafs which have to start on each side of the dollar sign (on each genome).

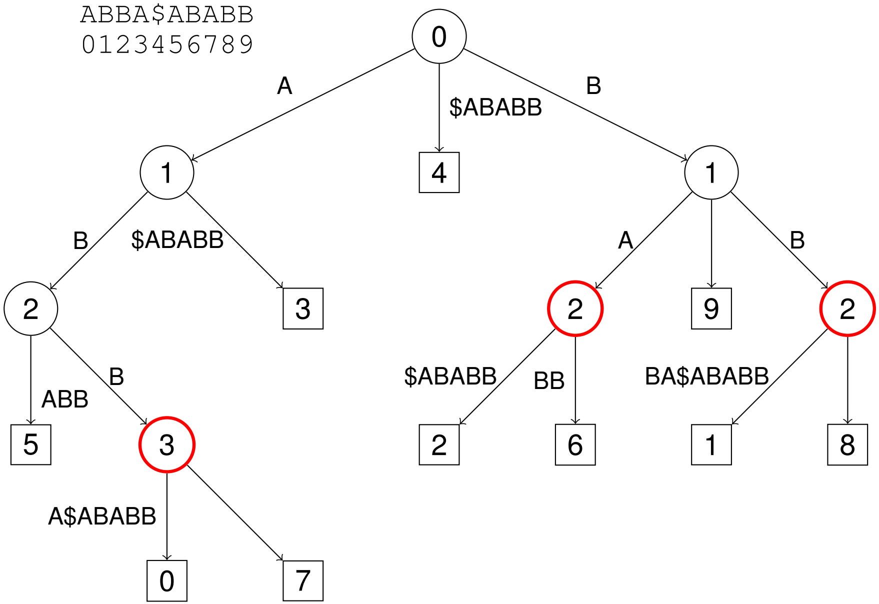
We'll need to check for condition 2 separately.

Finding potential MUMs

```
ABBA$ABABB
0123456789
```



Finding potential MUMs



Potential MUMs to MUMs

We have the following potential MUMs:

- `ABB` at starting positions 0 and 7.
- `BA` at starting positions 2 and 6.
- `BB` at starting positions 1 and 8.

Our string: `ABBA$ABABB`

- `ABB` achieves condition 2 (it's surrounded by mismatches).
- `BA` achieves condition 2 as well.
- `BB` does NOT achieve condition 2. In both cases it is preceded by a `A`.

Resulting MUMs: **`ABB`** and **`BA`**.

Results of using MUMmer

MUMmer was put to the test on various genomes:

- Two strains of tuberculosis (bacteria) that are >99% identical
    - 5 seconds to create the suffix tree.
    - 45 seconds to sort the MUMs.
    - 5 seconds to generate the Smith-Waterman alignments of the gaps.

Results of using MUMmer

MUMmer was put to the test on various genomes:

- Two strains of tuberculosis (bacteria) that are >99% identical
  - 5 seconds to create the suffix tree.
  - 45 seconds to sort the MUMs.
  - 5 seconds to generate the Smith-Waterman alignments of the gaps.
- Two 'cousin' genomes. Genome of *M.genitalium* (580,074 nucleotides) and *M.pneumoniae* (816,394 nucleotides)
  - 6.5 seconds to create the suffix tree.
  - 0.02 seconds to sort the MUMs.
  - 116 seconds to generate the Smith-Waterman alignments of the gaps.

Results of using MUMmer

MUMmer was put to the test on various genomes:

- Two strains of tuberculosis (bacteria) that are >99% identical
    - 5 seconds to create the suffix tree.
    - 45 seconds to sort the MUMs.
    - 5 seconds to generate the Smith-Waterman alignments of the gaps.
- Two ’cousin’ genomes. Genome of *M.genitalium* (580,074 nucleotides) and *M.pneumoniae* (816,394 nucleotides)
    - 6.5 seconds to create the suffix tree.
    - 0.02 seconds to sort the MUMs.
    - 116 seconds to generate the Smith-Waterman alignments of the gaps.
- Subsequence of a human chromosome 12p13 and mouse chromosome 6 (both rougly 230,000 nucleotides)
    - 1.6 seconds to create the suffix tree.
    - $\approx$ 0 seconds to sort the MUMs.
    - 27.4 seconds to generate the Smith-Waterman alignments of the gaps.

MUMmer 3

- MUMmer 3 is the latest version of MUMmer
- It was released in 2004 and is open-source.
- Requires less than half the memory and more than twice as fast than the initial MUMmer.
- Most notable problems fixed and options added since the initial MUMmer:
  - You can allow tolarence for mismatches when finding MUMs.
  - Can handle MUMs that are not necessarily unique.
  - All sorts of visualization tools.
- Could successfully compare all human chromosomes to each other. Tests were performed on a single computer with 950 MHz processor.
  - Computation time: 4.5 days
  - Memory used: 3.9 GB