Rapid Phylogenies

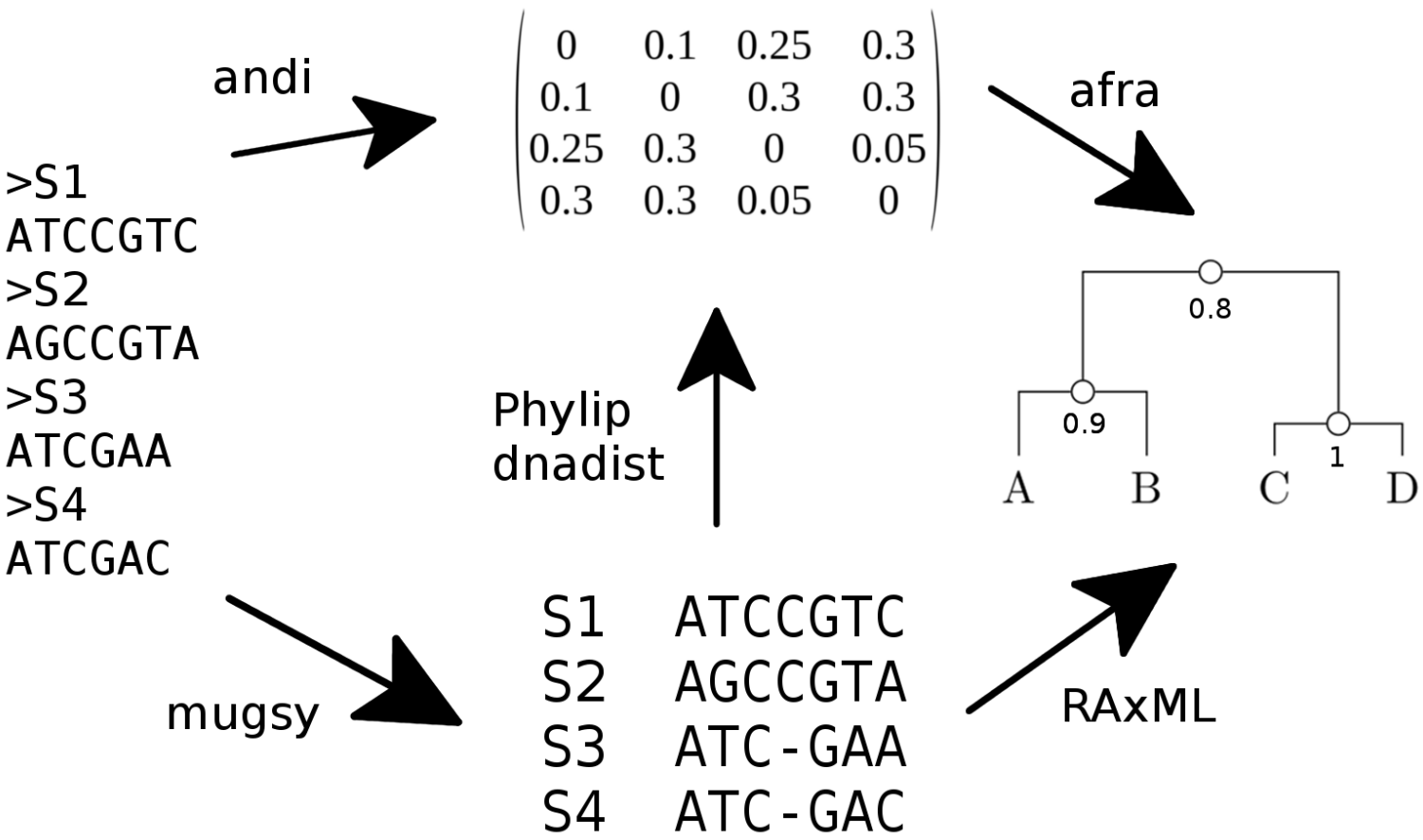
Fabian Klötzl

MPI for Evolutionary Biology, Plön

Hamburg, 2016-05-12

http://kloetzl.info/downloads/slides-hamburg-2016-05-12.pdf

You Had One Job



Our Plan for Today

1. Estimating Evolutionary Distances
2. Implementing an Enhanced Suffix Array
3. Alignment-Free Support Values
4. Fast Multiple Sequence Alignment

Section 1

Estimating Evolutionary Distances

Alignment

S_1 : A A G T T A G T G A A C C

S_2 : A A G T C A T T G A A A C C

Alignment

S_1:	A	A	G	T	T	A	G	T	G	A	A	–	C	C
	|	|	|	|		|		|	|	|	|		|	|
S_2:	A	A	G	T	C	A	T	T	G	A	A	A	C	C

Alignment

$$
\begin{array}{lcccccccccccccc}
S_1: & \mathrm{A} & \mathrm{A} & \mathrm{G} & \mathrm{T} & \mathrm{T} & \mathrm{A} & \mathrm{G} & \mathrm{T} & \mathrm{G} & \mathrm{A} & \mathrm{A} & - & \mathrm{C} & \mathrm{C} \\
 & | & | & | & | & & | & & | & | & | & | & & | & | \\
S_2: & \mathrm{A} & \mathrm{A} & \mathrm{G} & \mathrm{T} & \mathrm{C} & \mathrm{A} & \mathrm{T} & \mathrm{T} & \mathrm{G} & \mathrm{A} & \mathrm{A} & \mathrm{A} & \mathrm{C} & \mathrm{C}
\end{array}
$$

$$d(S_1, S_2) \approx \frac{2}{14}$$

Alignment

$$
\begin{array}{lcccccccccccccc}
S_1: & \mathrm{A} & \mathrm{A} & \mathrm{G} & \mathrm{T} & \mathrm{T} & \mathrm{A} & \mathrm{G} & \mathrm{T} & \mathrm{G} & \mathrm{A} & \mathrm{A} & - & \mathrm{C} & \mathrm{C} \\
 & | & | & | & | & & | & & | & | & | & | & & | & | \\
S_2: & \mathrm{A} & \mathrm{A} & \mathrm{G} & \mathrm{T} & \mathrm{C} & \mathrm{A} & \mathrm{T} & \mathrm{T} & \mathrm{G} & \mathrm{A} & \mathrm{A} & \mathrm{A} & \mathrm{C} & \mathrm{C}
\end{array}
$$

$$d(S_1, S_2) \approx \frac{2}{14}$$

How does one deal with indels?

Alignment

$$
\begin{array}{lcccccccccccccc}
S_1: & \mathrm{A} & \mathrm{A} & \mathrm{G} & \mathrm{T} & \mathrm{T} & \mathrm{A} & \mathrm{G} & \mathrm{T} & \mathrm{G} & \mathrm{A} & \mathrm{A} & - & \mathrm{C} & \mathrm{C} \\
 & | & | & | & | & & | & & | & | & | & | & & | & | \\
S_2: & \mathrm{A} & \mathrm{A} & \mathrm{G} & \mathrm{T} & \mathrm{C} & \mathrm{A} & \mathrm{T} & \mathrm{T} & \mathrm{G} & \mathrm{A} & \mathrm{A} & - & \mathrm{C} & \mathrm{C}
\end{array}
$$

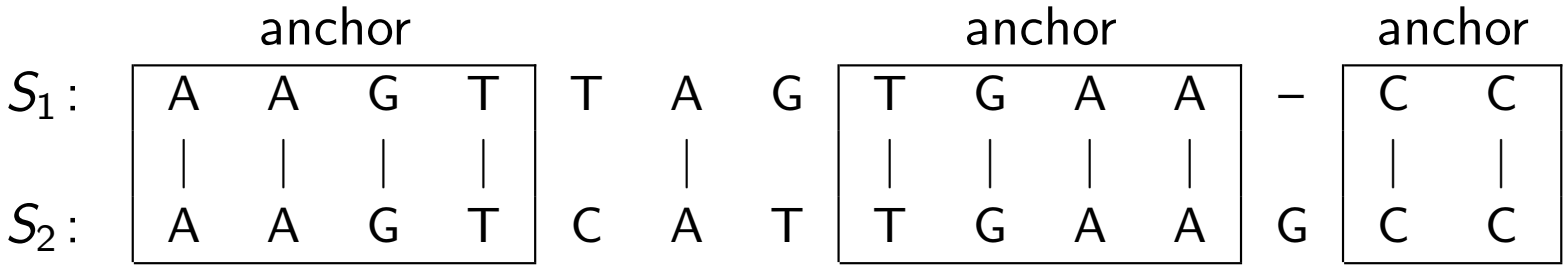
$$d(S_1, S_2) = \frac{2}{13}$$

Indels are ignored for the evolutionary distance in common models (Jukes-Cantor, Kimura 2 parameter)!

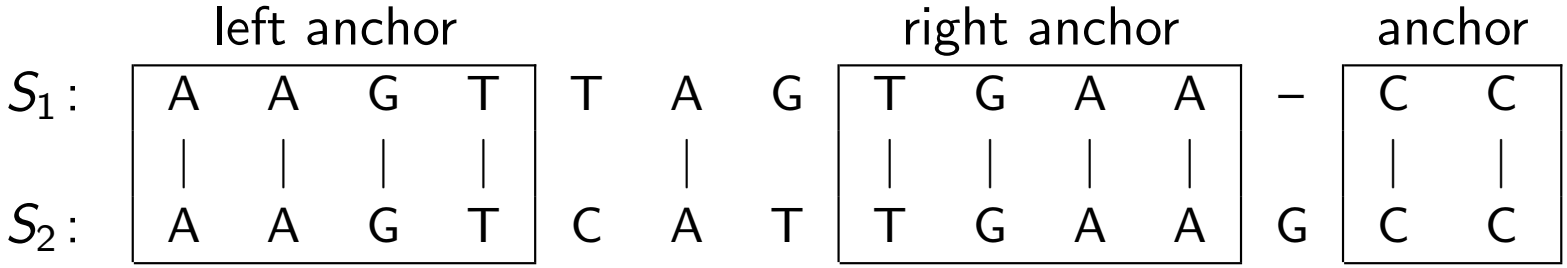
Idea

If indels can be ignored, let's not compute them!

A Blast from the Past



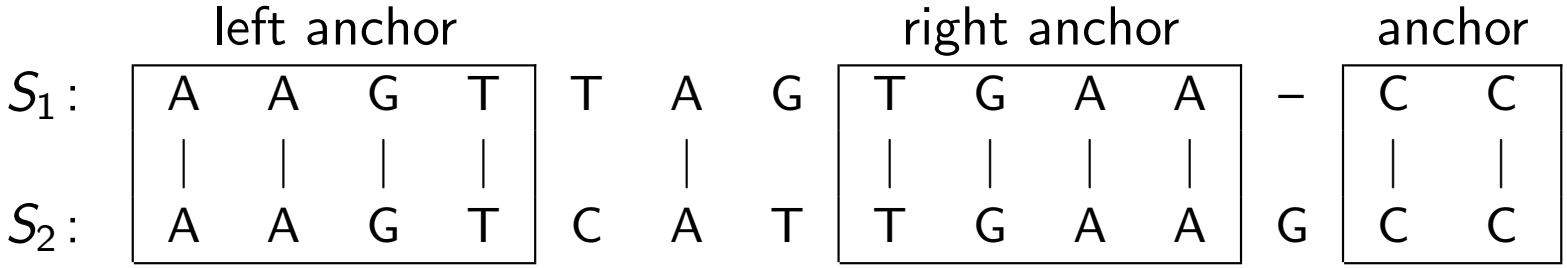
A Blast from the Past



Anchor Pair

Two anchors form a pair if they are *equidistant*.

A Blast from the Past



Anchor Pair

Two anchors form a pair if they are *equidistant*.

Anchor Distance

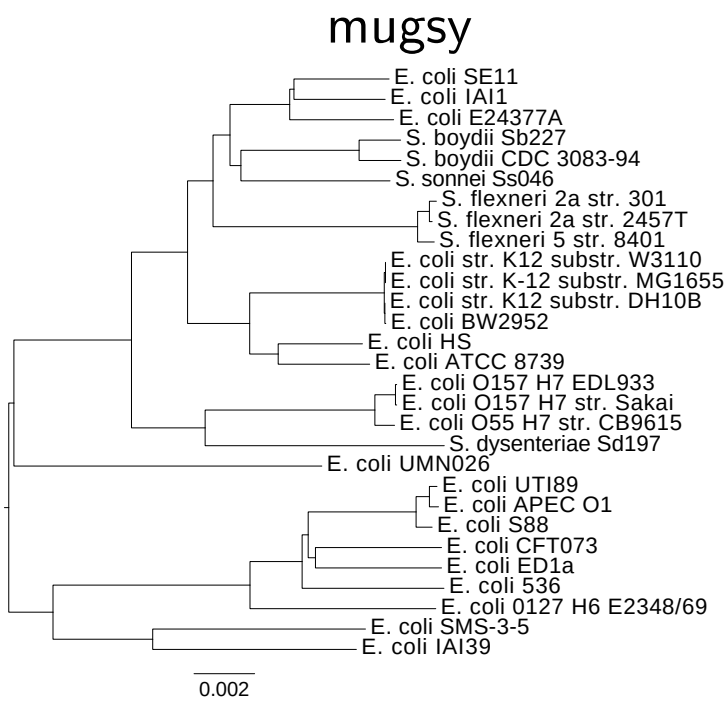
$$d_a = \frac{\#SNP}{\#HomolNucl} = \frac{2}{11}$$

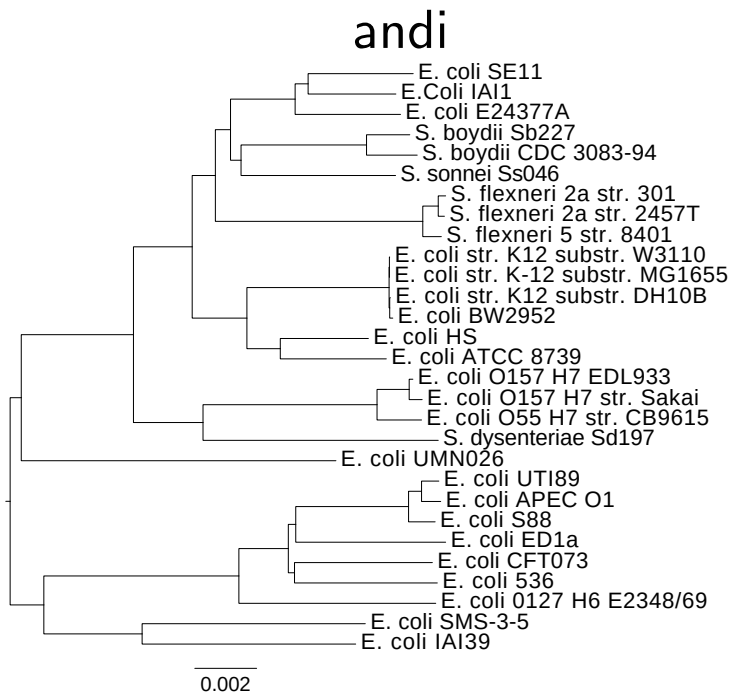
Visualisation

https://evolbioinf.github.io/andi/

Demo

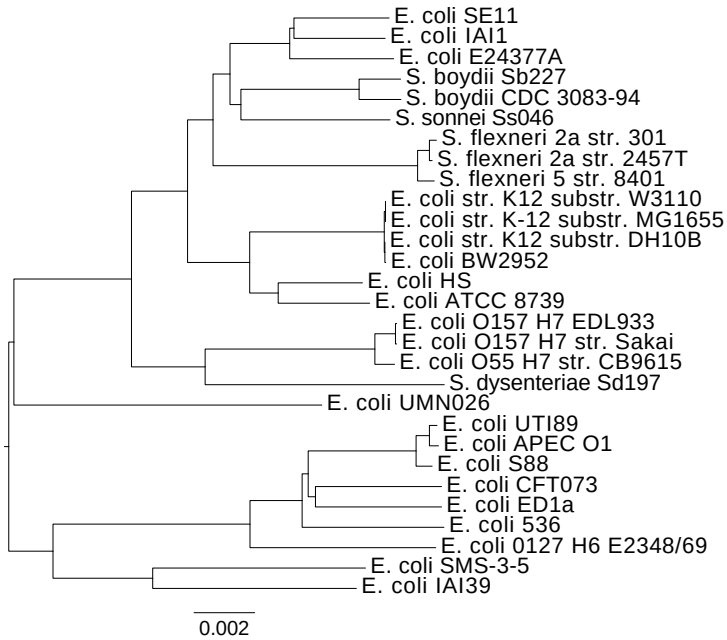
29 whole *E. Coli/ Shigella* Genomes





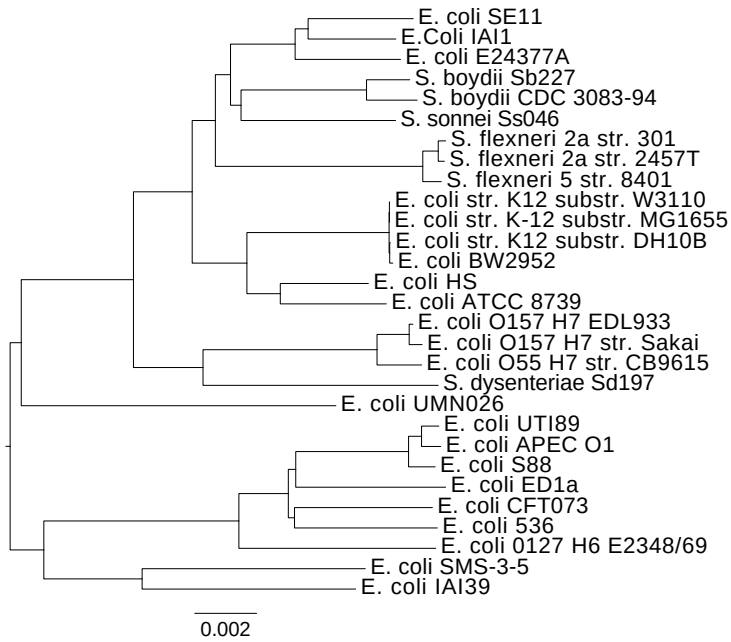
29 whole *E. Coli/ Shigella* Genomes

mugsy



10 164 s

andi

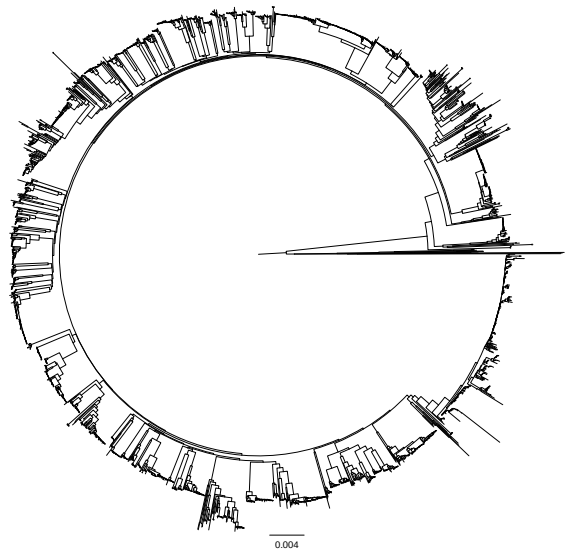


??? s

Size Matters

- 3085 *Streptococcus pneumoniae* genomes
- multiple contigs per genome
- 2 million nucleotides per genome
- a total filesize of 6.8 GB
- over 9 million pairwise comparisions

- cophylog: 36 days, 2.3 GB
- andi v0.10: 4 h 59 min, 9.8 GB
- most tools exceed 256 GB memory



Section 2

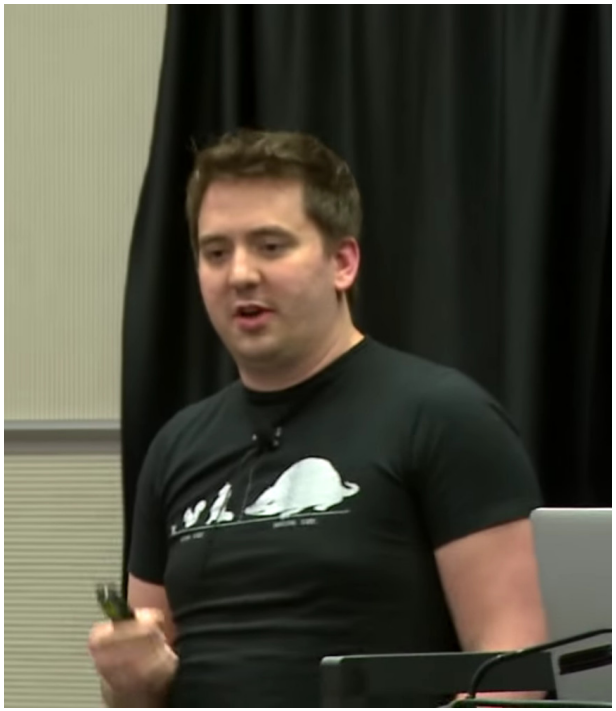
Implementing an Enhanced Suffix Array

getInterval with Child Tables

Not all $O(1)$ are created equal

```
1  input (l–[i..j], m), a
2
3  do
4      if T[ SA[m] +l] = a then
5          j ← m − 1
6          m ← CLD[j+1].L
7          goto line 17
8      end
9
10     if m = j then
11         break
12     end
13
14     m ← CLD[m].R
15 while LCP[m]=l
16 if T[SA[i] + l] = a then
17     l ← LCP[m]
18     output (l–[i..j], m)
19 else
20     output ⊥
21 end
```

Chandler Carruth



Three step recipe to improve performance:

1. Measure!
2. Do less work
3. Do it faster

CppCon 2014: Chandler Carruth "Efficiency with Algorithms, Performance with Data Structures"

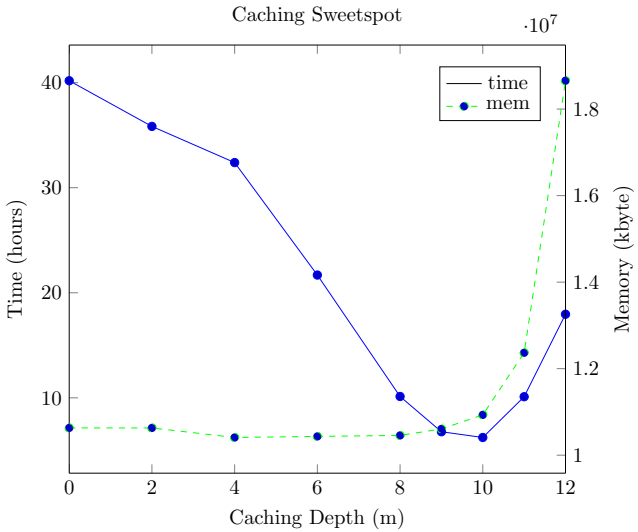
Step 1

Measure

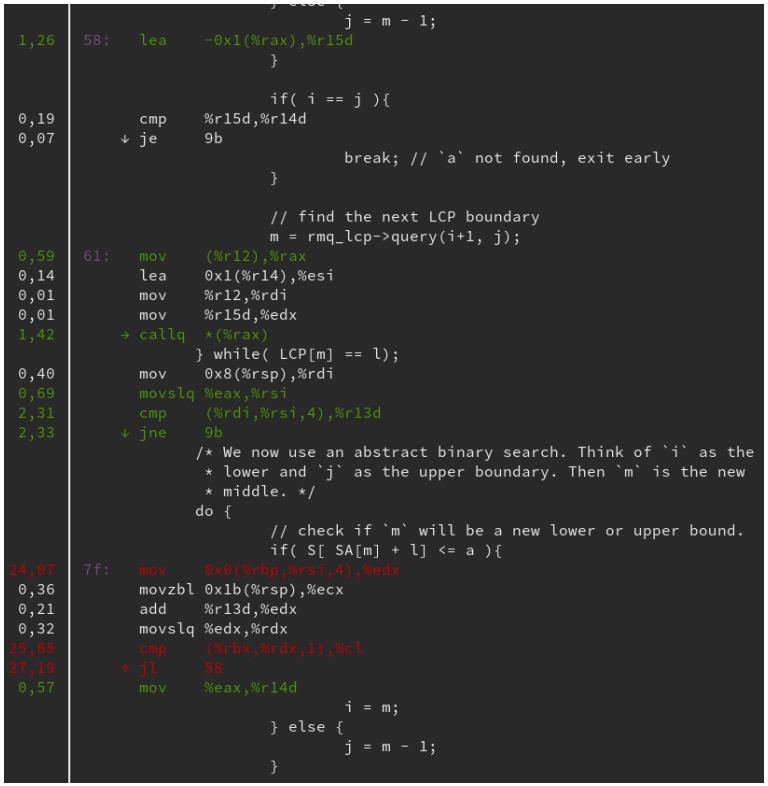
Step 2: Do less work
Less Memory Accesses

```
amples: 705K of event 'cycles:ppp', Event count (approx.): 410864633684
verhead  Command  Shared Object            Symbol
 33,43%  andi     andi                     [.] get_interval
 10,37%  andi     andi                     [.] RMQ_n_1_improved::query
  8,80%  andi     andi                     [.] esa_init
  7,69%  andi     andi                     [.] get_match_cached
  4,97%  andi     libdivsufsort.so.3.0.0   [.] divsufsort
  4,86%  andi     andi                     [.] get_match_from
  4,36%  andi     andi                     [.] RMQ_nlogn_1::query
  1,66%  andi     andi                     [.] RMQ_nlogn_1::RMQ_nlogn_1
  1,45%  andi     andi                     [.] RMQ_n_1_improved::RMQ_n_1_improved
  1,11%  andi     libdivsufsort.so.3.0.0   [.] 0x00000000000061ec
  0,99%  andi     libdivsufsort.so.3.0.0   [.] 0x0000000000006220
  0,78%  andi     andi                     [.] revcomp
  0,67%  andi     andi                     [.] dist_anchor
  0,53%  andi     [kernel.vmlinux]         [k] pageblock_pfn_to_page
  0,49%  andi     [kernel.vmlinux]         [k] clear_page_c_e
  0,39%  andi     libdivsufsort.so.3.0.0   [.] 0x00000000000011a5
  0,36%  andi     [wl]                     [k] osl_readw
  0,36%  andi     libdivsufsort.so.3.0.0   [.] 0x00000000000011ad
```

- Avoid calling *getInterval*
- Flatten the interval tree into buckets
- Lookup prefix of length m in $O(1)$

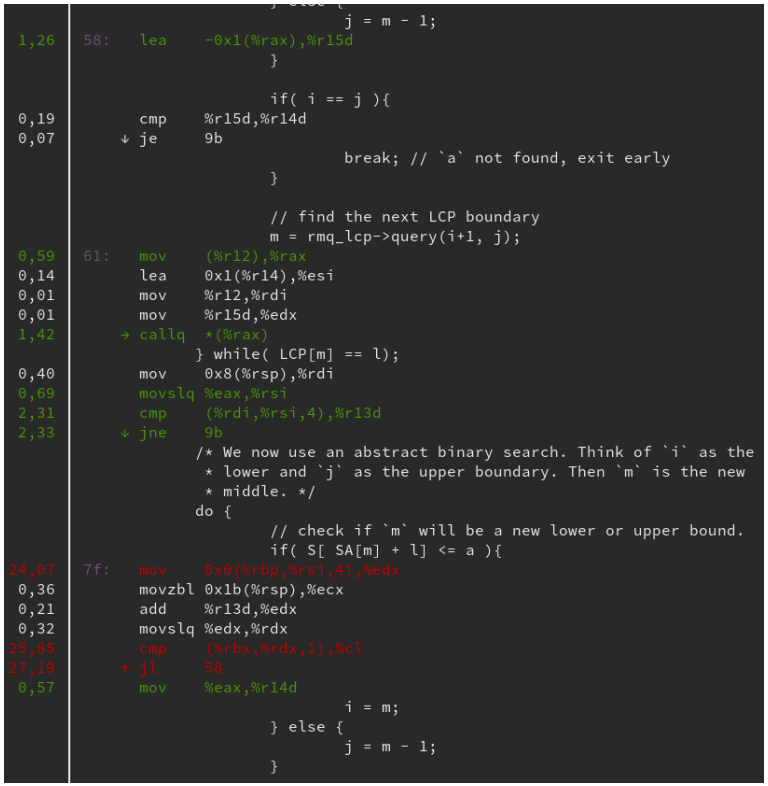


Step 3: Do it faster



- $T[\, SA[m] + l]$
- two memory lookups
- unpredictable
- not in cache
- only uses one byte of a whole cache line

Step 3: Do it faster



- $T[\, SA[m] + l]$
- two memory lookups
- unpredictable
- not in cache
- only uses one byte of a whole cache line

Fortunately,

- l is known ahead
- Precompute $T[\, SA[m] + LCP[m]]$

The First Variant Character

Yet another TLA

i	SA	LCP	$T^{SA[i]}$	lcp − intervals
0	4	−1	AAGG	
1	0	3	AAG**T**AAGG	0
2	5	1	A**G**G	1
3	1	2	AG**T**AAGG	3
4	7	0	**G**	2
5	6	1	G**G**	1
6	2	1	G**T**AAGG	
7	3	0	**T**AAGG	
8		−1		

getInterval with Child Tables and FVC

```
1   ii ← i
2   c ← T[ SA[m] +l]
3   goto line 7
4   do
5       c ← FVC[i]
6
7       if c = a then
8           n ← CLD[j+1].L
9           output (LCP[n] − [i..m−1], n)
10      end
11
12      i ← m
13      if i = j then
14          break
15      end
16
17      m ← CLD[m].R
18  while LCP[m]=l
19  if ii = i then
20      c ← T[SA[i] + l]
21  else
22      c ← FVC[i]
23  end
24  if c = a then
25      l ← LCP[m]
26      output (l−[i..j], m)
27  else
28      output ⊥
29  end
```

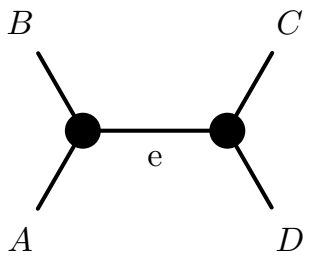
- 20% speed-up
- DC-array by Wu (2016)
- Optimistic algorithm

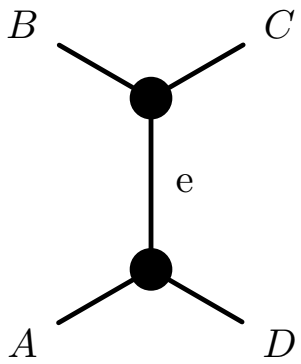
Section 3

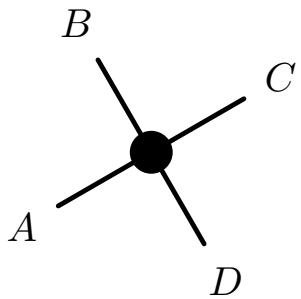
Alignment-Free Support Values

Quiz

4				
A	0	0.3	0.2	0.1
B	0.3	0	0.1	0.3
C	0.2	0.1	0	0.3
D	0.1	0.3	0.3	0







Quartet Support Values

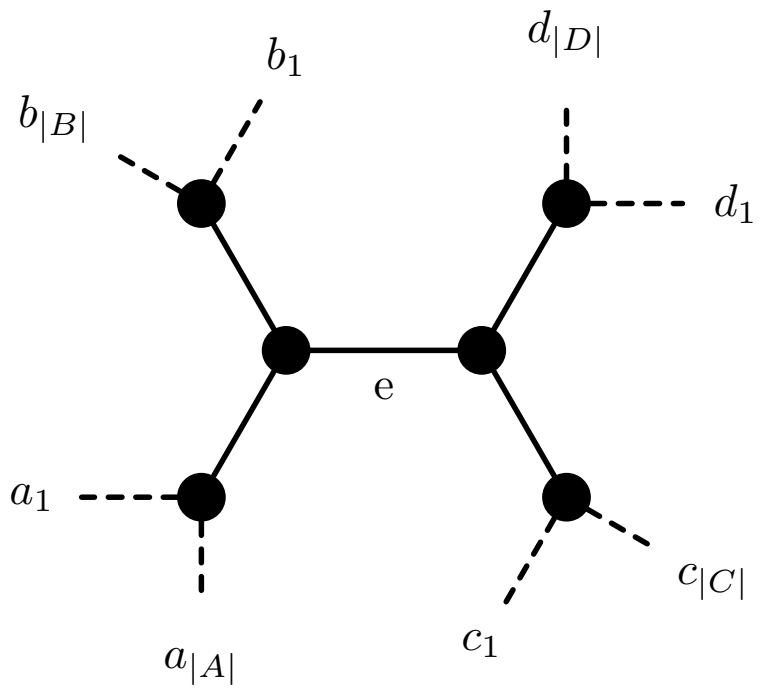
No alignment $\Rightarrow$ no bootstrapping!

Quartets

A quartet a, b, c, d is called supporting an edge e if

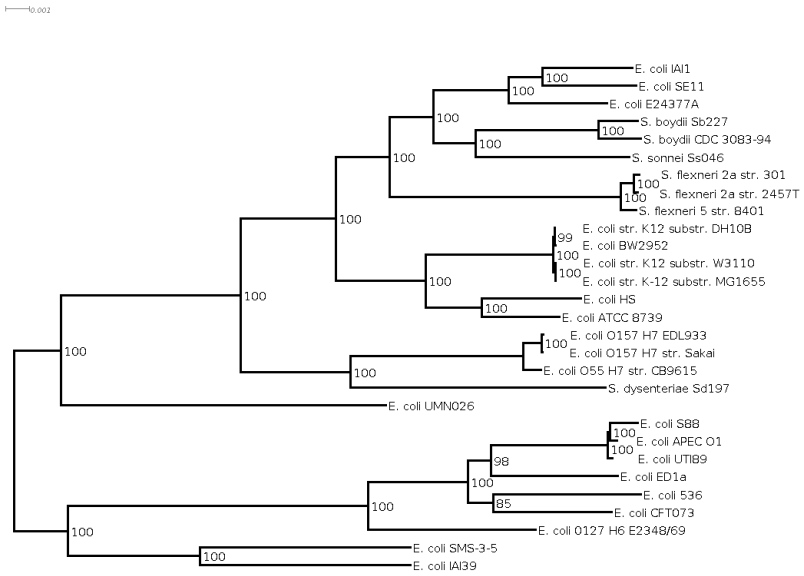
$$D(a,b) + D(c,d) < \min\{D(a,c) + D(b,d), D(b,c) + D(a,d)\}.$$

$$SV_e = \frac{\sum_{a,b,c,d} \mathbf{1}(a,b,c,d \text{ support } e)}{|A||B||C||D|}$$



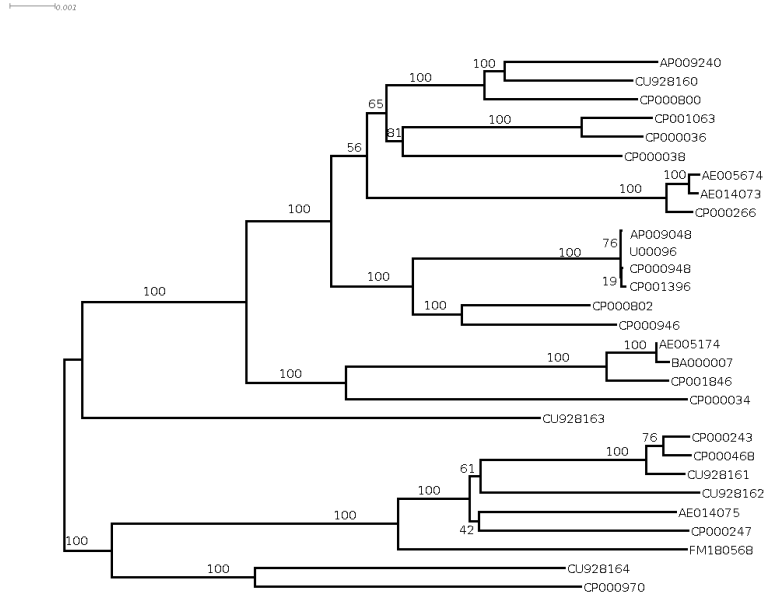
Eco29 again

mugsy + RAxML



2h 49min + 6h

andi + afra

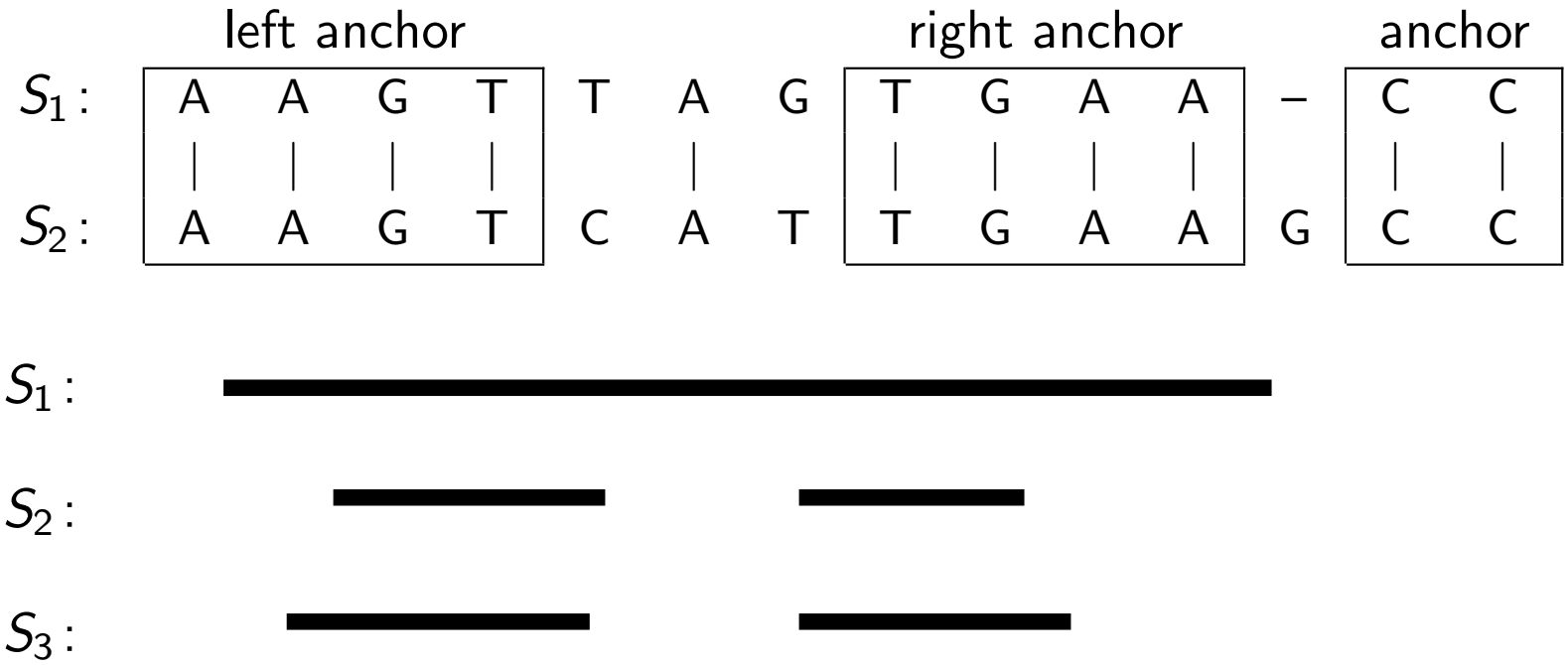


27 s + 1 s

Section 4

Fast Multiple Sequence Alignment

Align to Reference



UGAlign

Demo

Thank you for your attention

Bernhard Haubold

- `apt-get install andi`
- `brew install science/andi`
- `aura -A andi`
- All projects: `github.com/evolbioinf`