

**splitMEM:
graphical pan-genome analysis
with suffix skips**

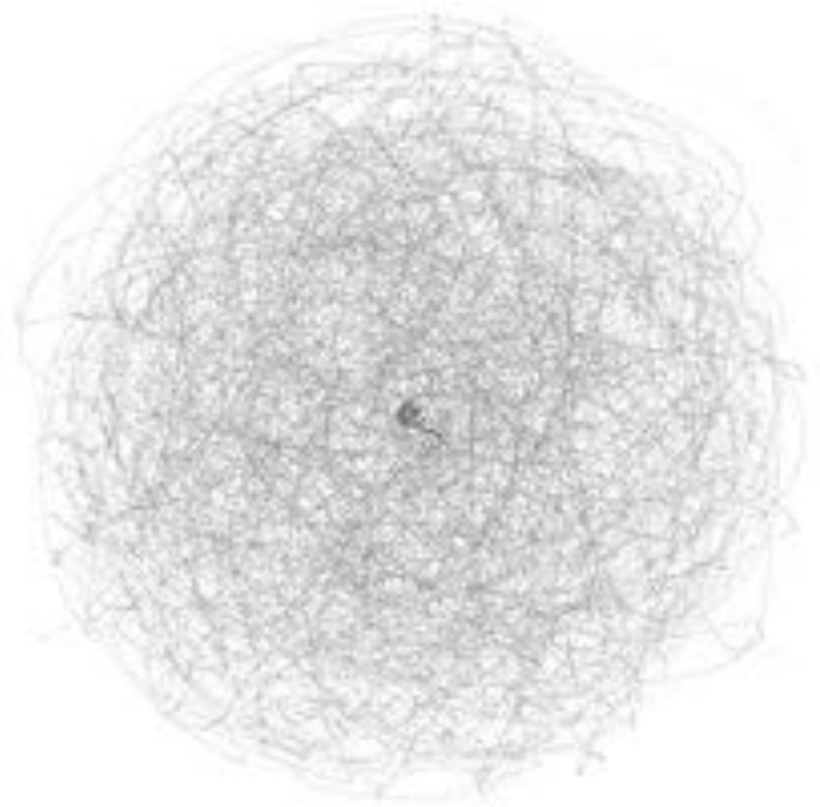
Shoshana Marcus

May 7, 2014



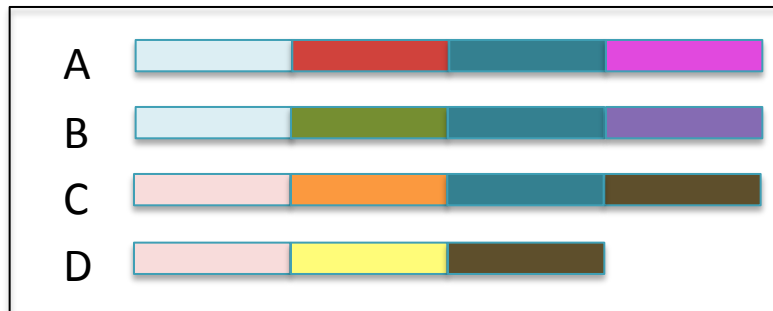
Outline

- 1 Overview
- 2 Data Structures
- 3 splitMEM Algorithm
- 4 Pan-genome Analysis

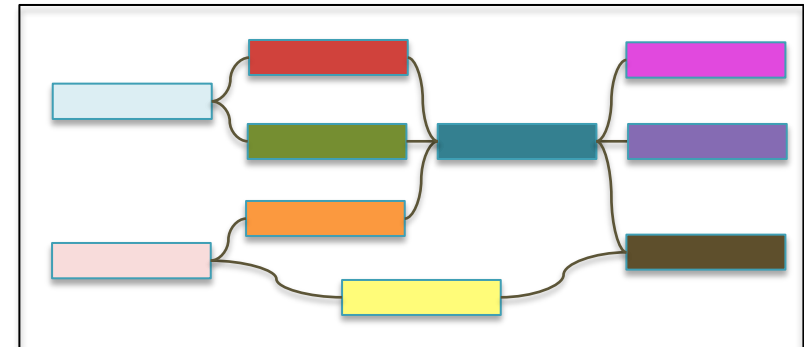


Objective

Input



Output



- Several complete genomes
- Available today for many microbial species, near future for higher eukaryotes
- Pan-genome: analyze multiple genomes of species together

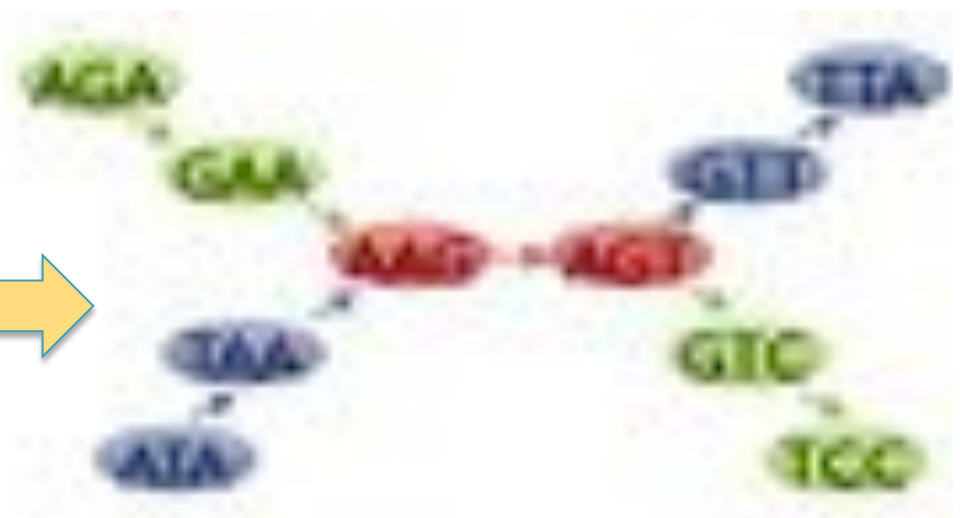
Compressed de Bruijn graph

- Graphical representation depicts how population variants relate to each other, especially where they diverge at branch points
- How well conserved is a sequence?
- What are network properties?

de Bruijn graph

- Node for each distinct kmer
- Directed edge connects consecutive kmers
- Nodes overlap by $k-1$ bp
- Self-loops, multi-edges

AGAAGTCC
ATAAGTTA



Reconstruct original sequence:

Eulerian path through graph, visit each edge once

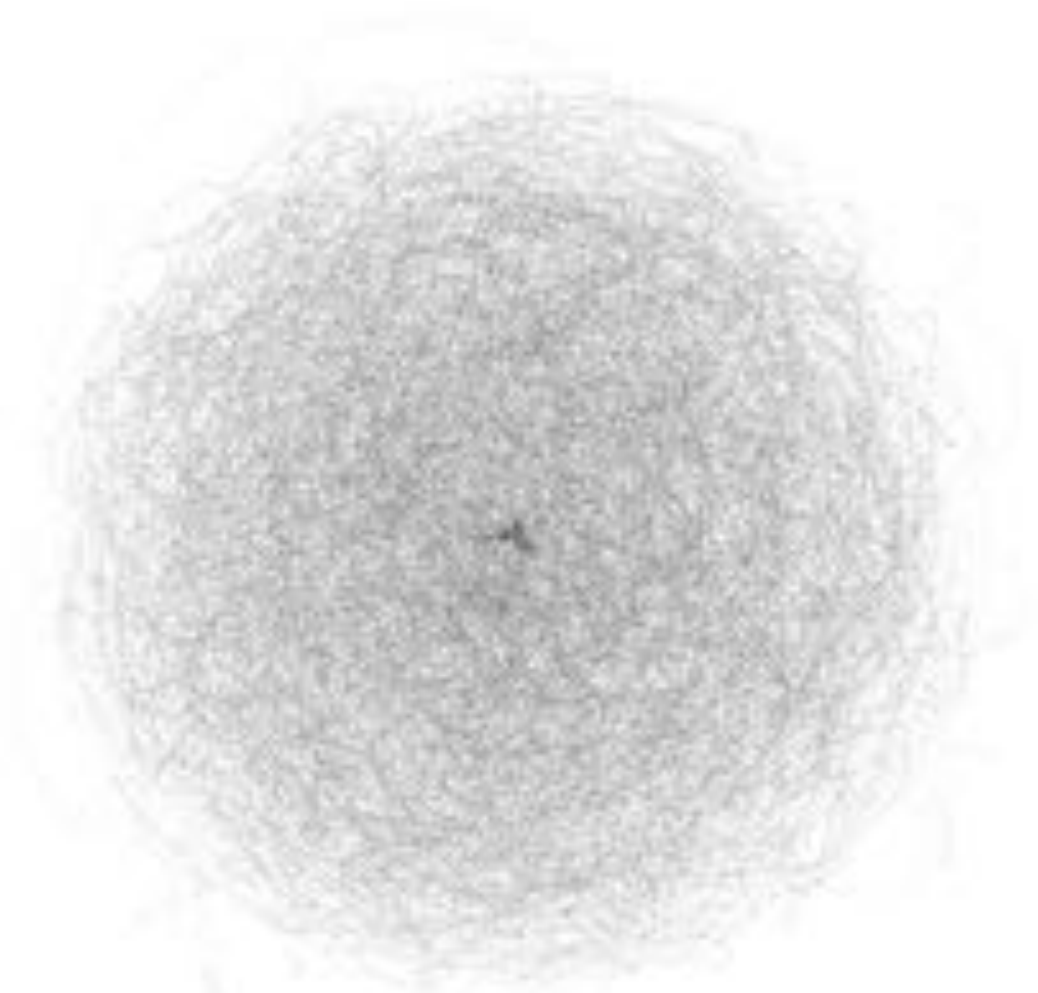
Compressed de Bruijn graph

- Merge non-branching chains of nodes
- Min. number of nodes that preserve path labels



- ✧ Usually built from uncompressed graph
- ✧ We build directly in $O(n \log n)$ time and space

Compressed de Bruijn graph



9 strains of *Bacillus anthracis* $k=25$

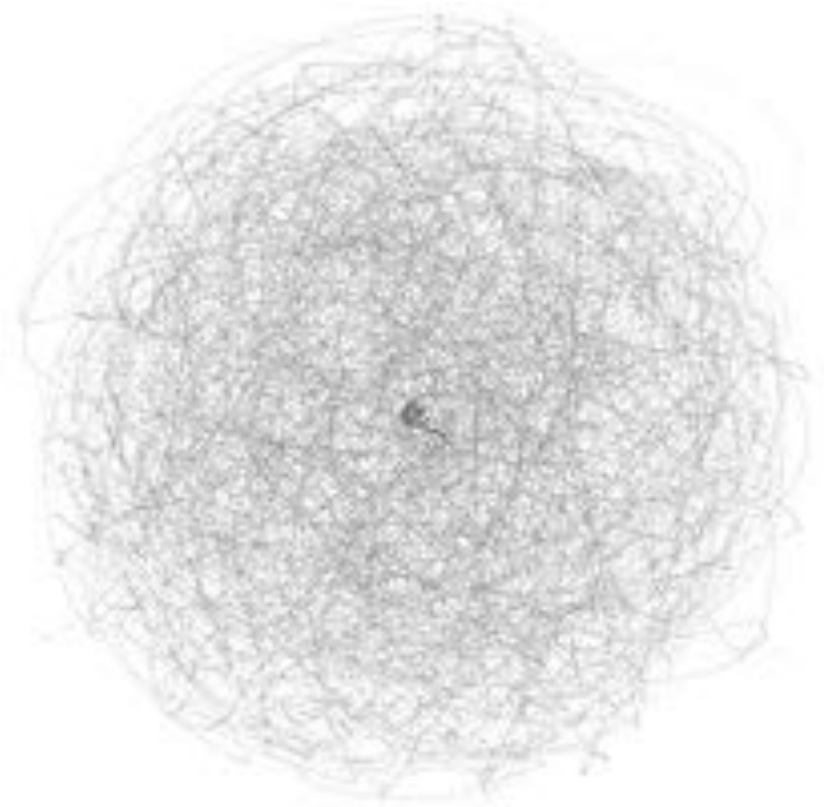
Compressed de Bruijn graph



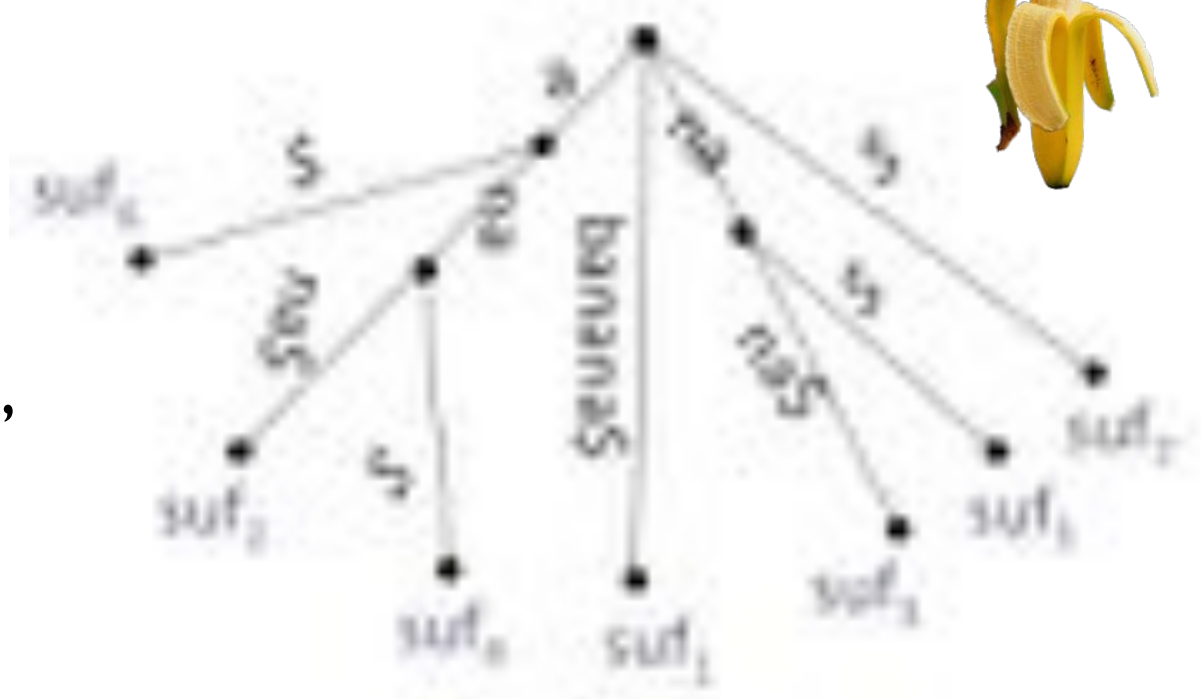
9 strains of *Bacillus anthracis* $k=1000$

Outline

- 1 Overview
- 2 Data Structures
- 3 splitMEM Algorithm
- 4 Pan-genome Analysis



Suffix Tree



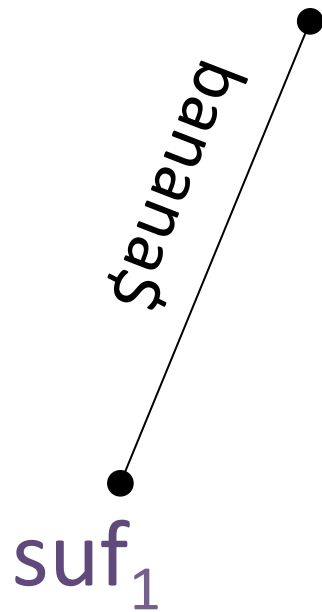
- Rooted, directed tree with leaf for each suffix.
- Each internal node, except the root, has at least two children.
- Each edge is labeled with nonempty substring.
- No two siblings begin with the same character.
- Path from root to leaf i spells suffix $S[i \dots n]$.
- Append special character $\$$ to guarantee each suffix ends at leaf.



Constructing Suffix Tree

Naïve Algorithm

$S = \text{banana}\$$



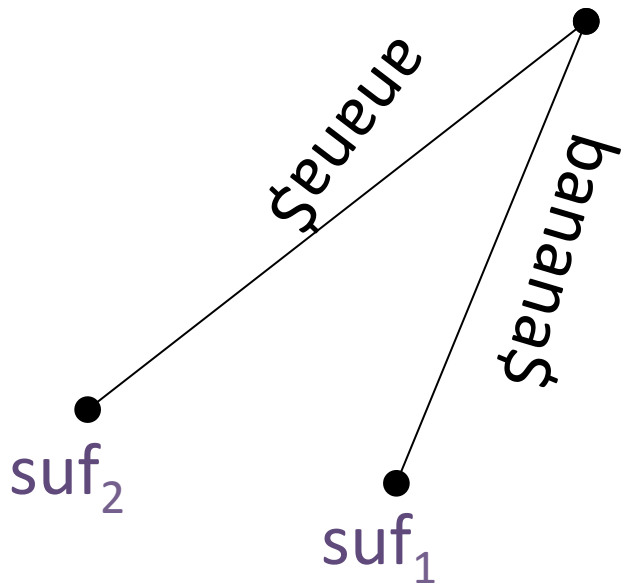
suf_1
 $\text{banana}\$$



Constructing Suffix Tree

Naïve Algorithm

$S = \text{banana}\$$



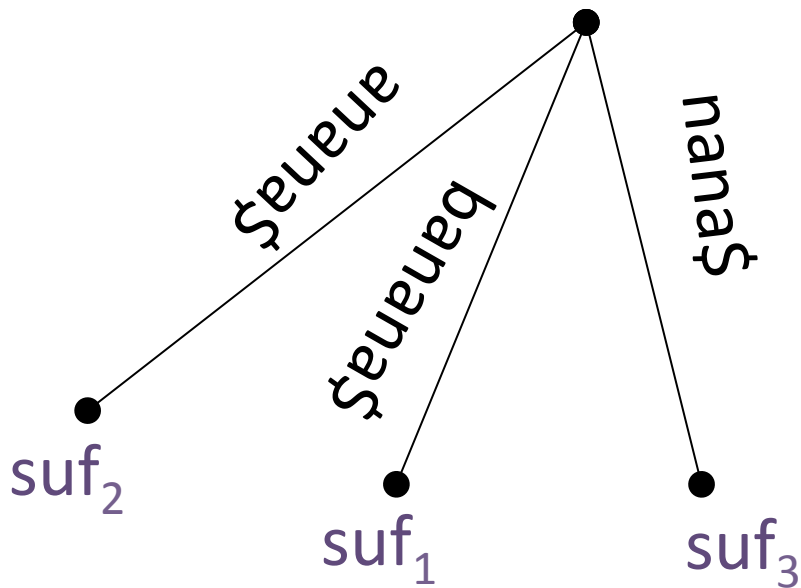
suf₂
anana\$



Constructing Suffix Tree

Naïve Algorithm

$S = \text{banana}\$$



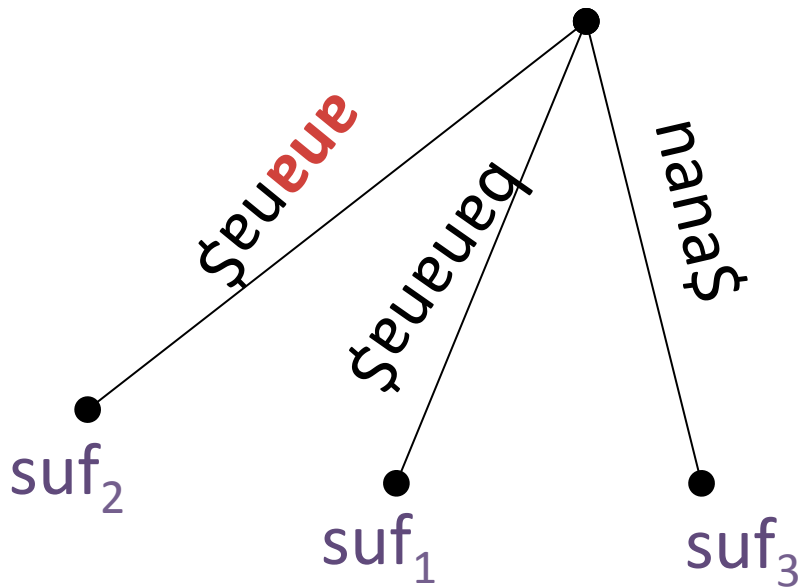
suf₃
nana\$



Constructing Suffix Tree

Naïve Algorithm

S = banana\$



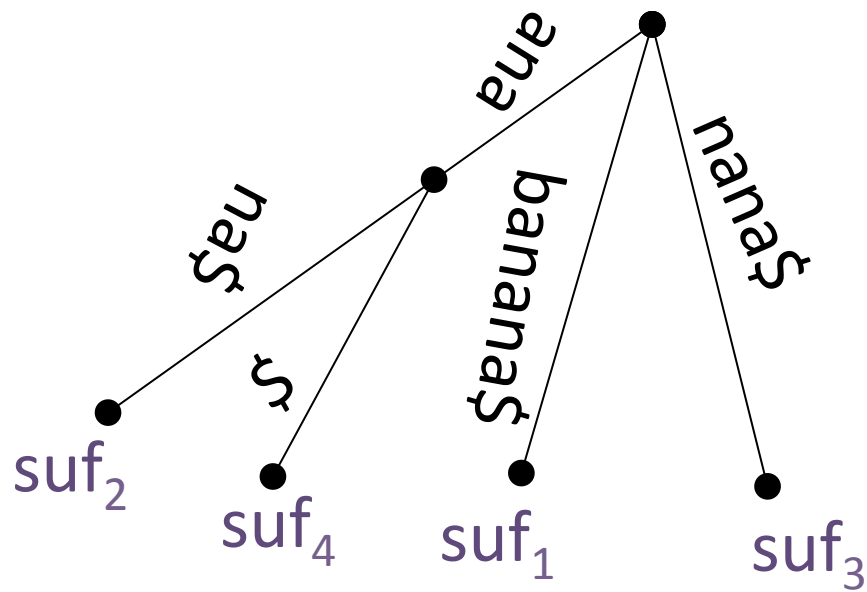
suf₄
ana\$



Constructing Suffix Tree

Naïve Algorithm

S = banana\$

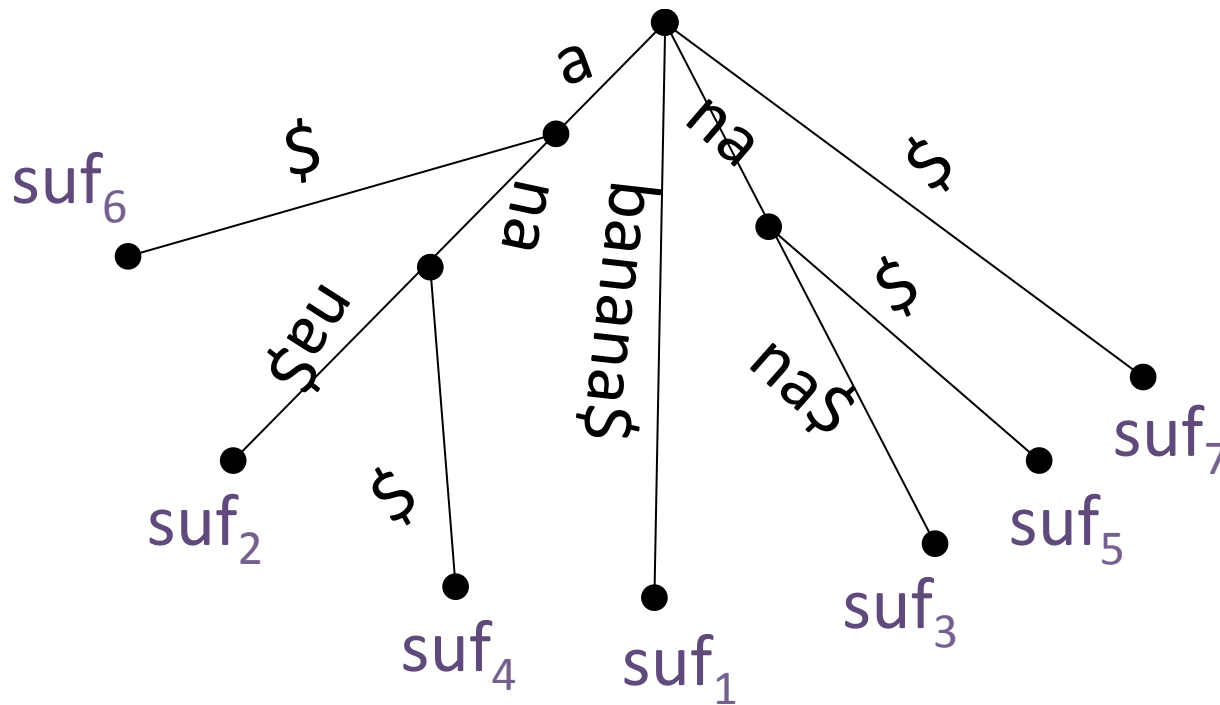


suf₄
ana\$



Constructing Suffix Tree

Naïve Algorithm



S = banana\$

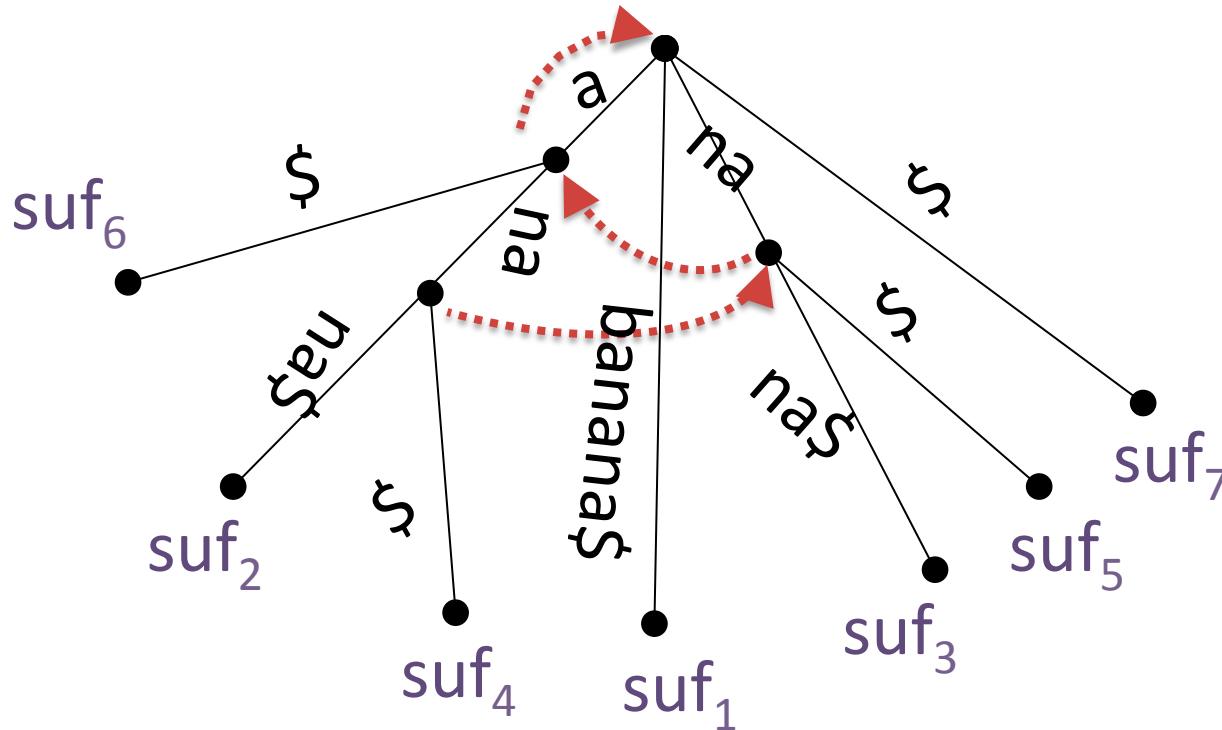
banana\$	suf ₁
anana\$	suf ₂
nana\$	suf ₃
ana\$	suf ₄
na\$	suf ₅
a\$	suf ₆
\$	suf ₇

$O(n^2)$ time

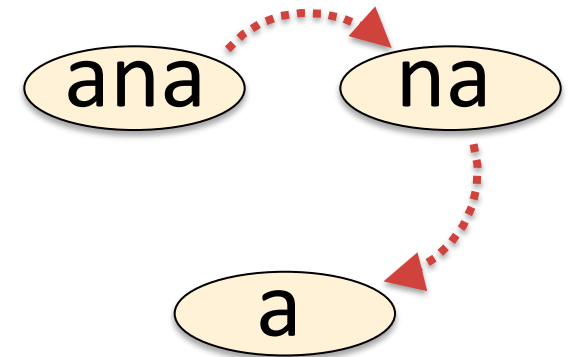


Constructing Suffix Tree

$O(n)$ time



Suffix Links

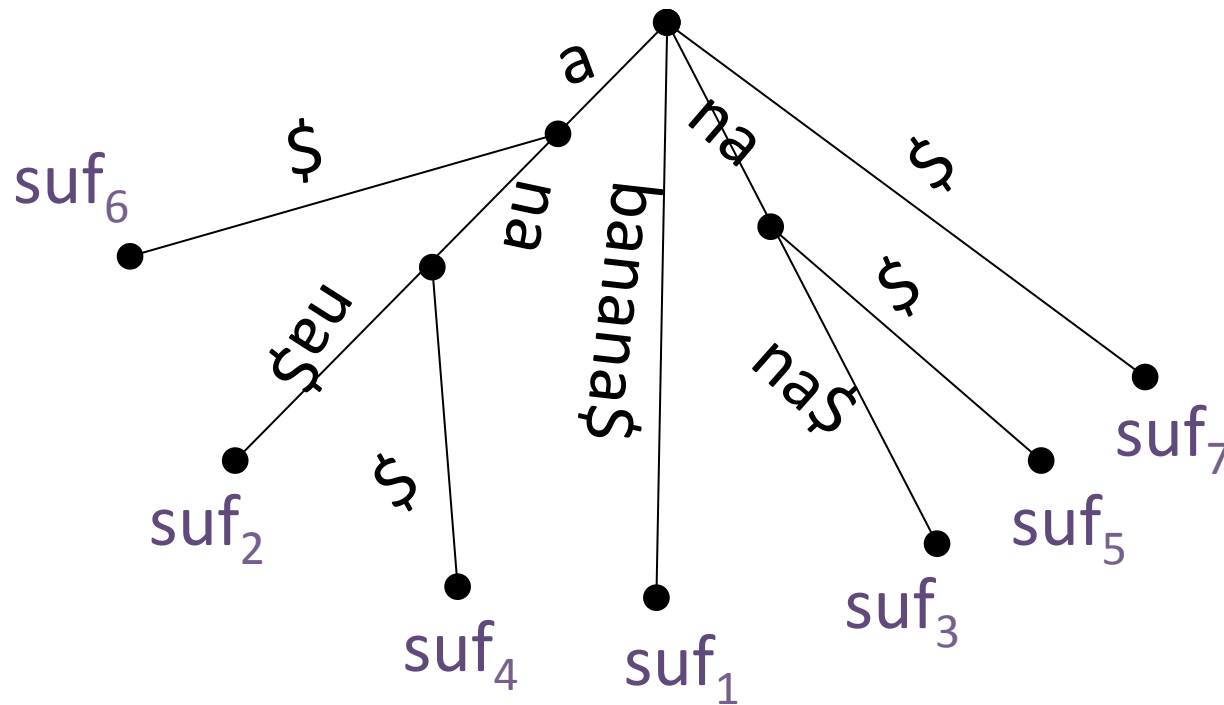


On-line Constructin of Suffix Trees, E. Ukkonen
Algorithmica (1995)



Suffix Tree Query

S = banana\$

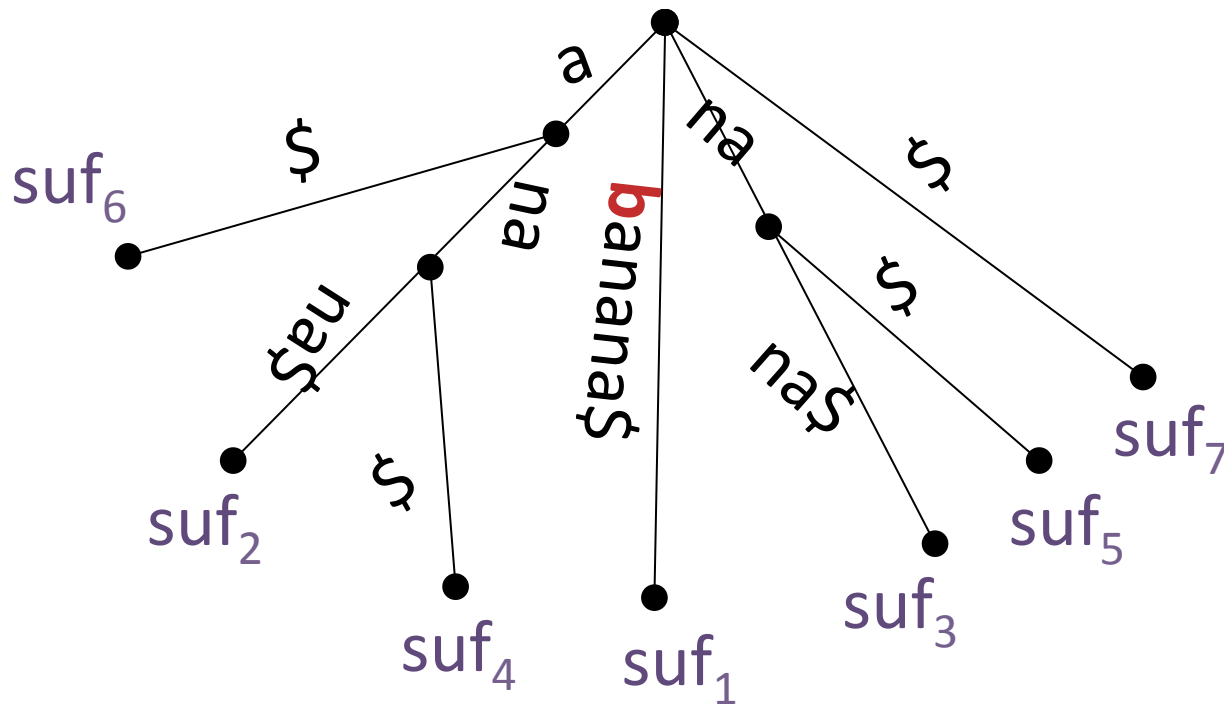


Search for **ban**



Suffix Tree Query

S = banana\$

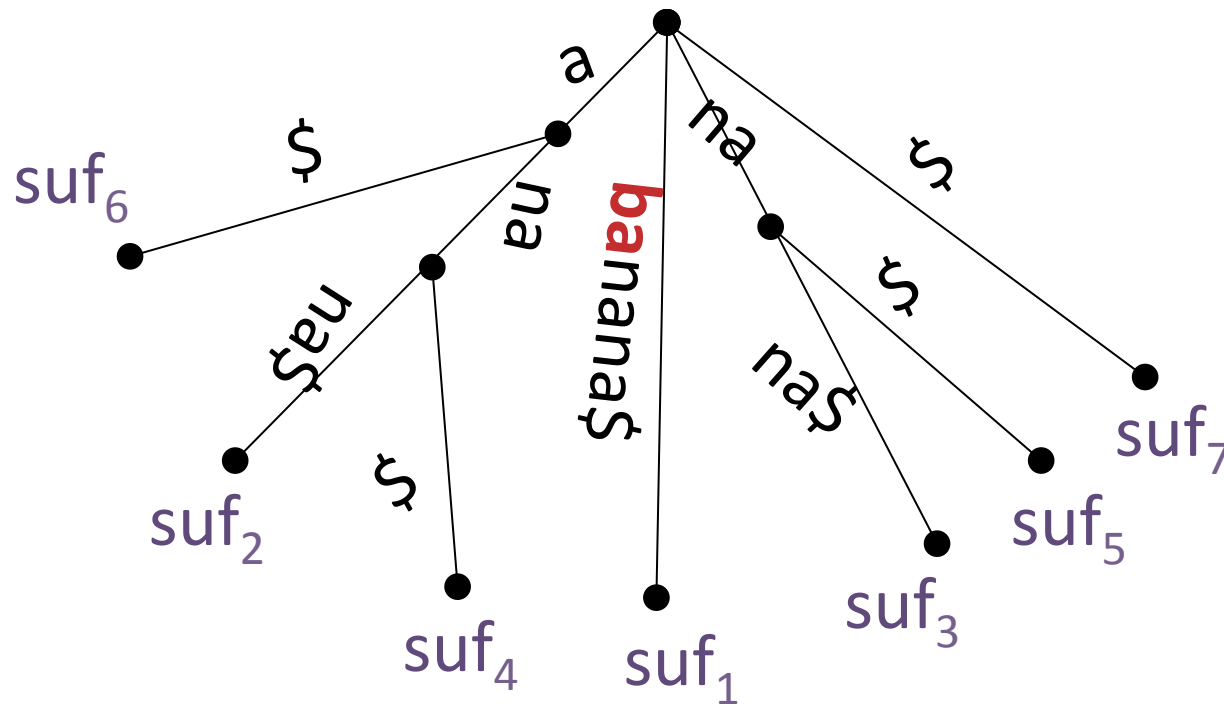


Search for **ban**



Suffix Tree Query

S = banana\$

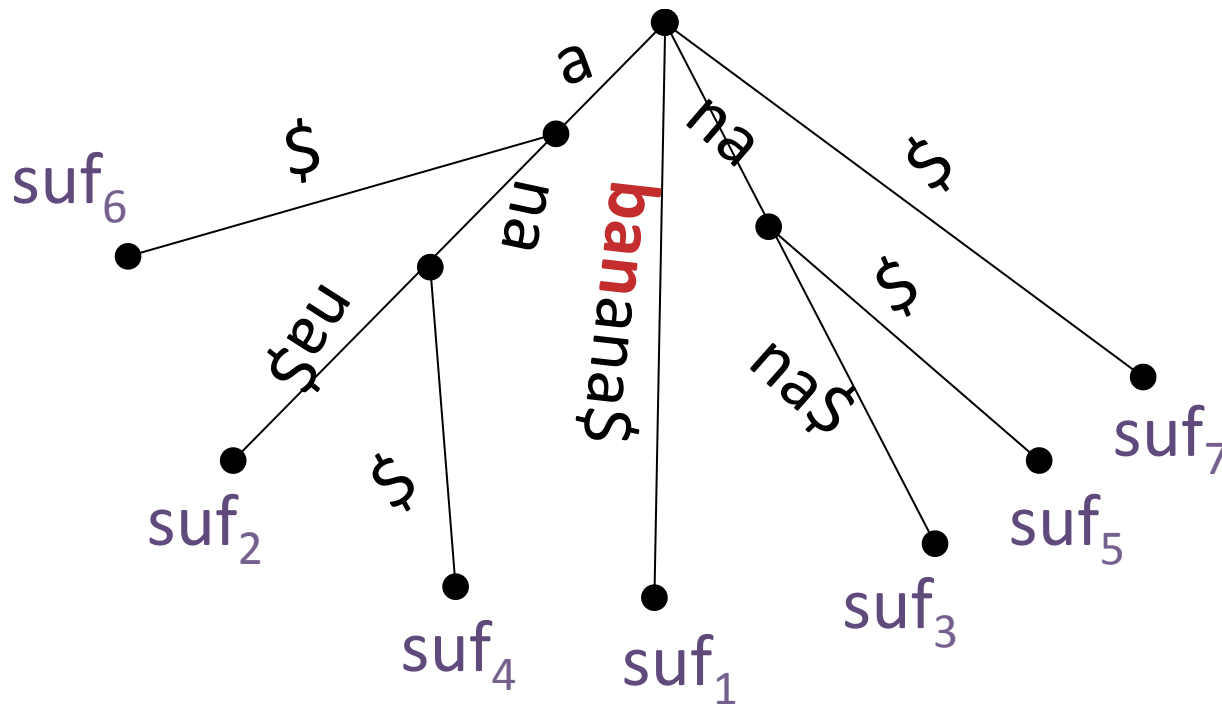


Search for **ban**



Suffix Tree Query

S = banana\$



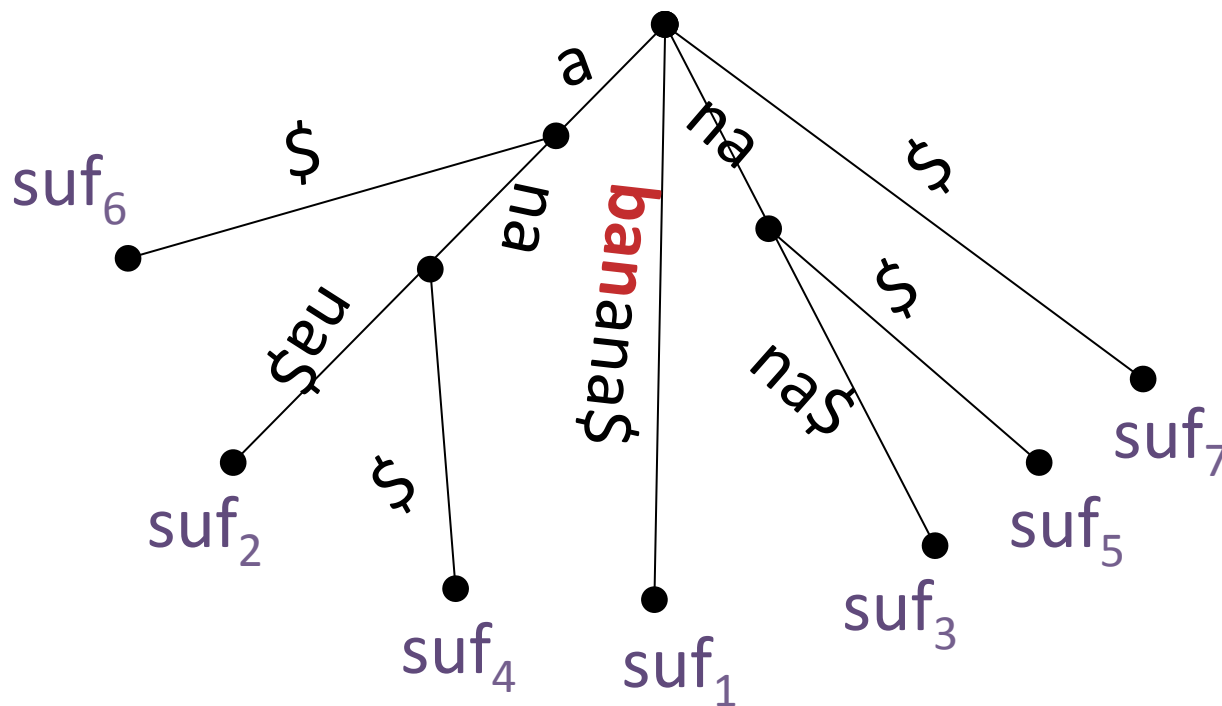
Search for **ban**

Found 1
occurrence



Suffix Tree Query

S = banana\$



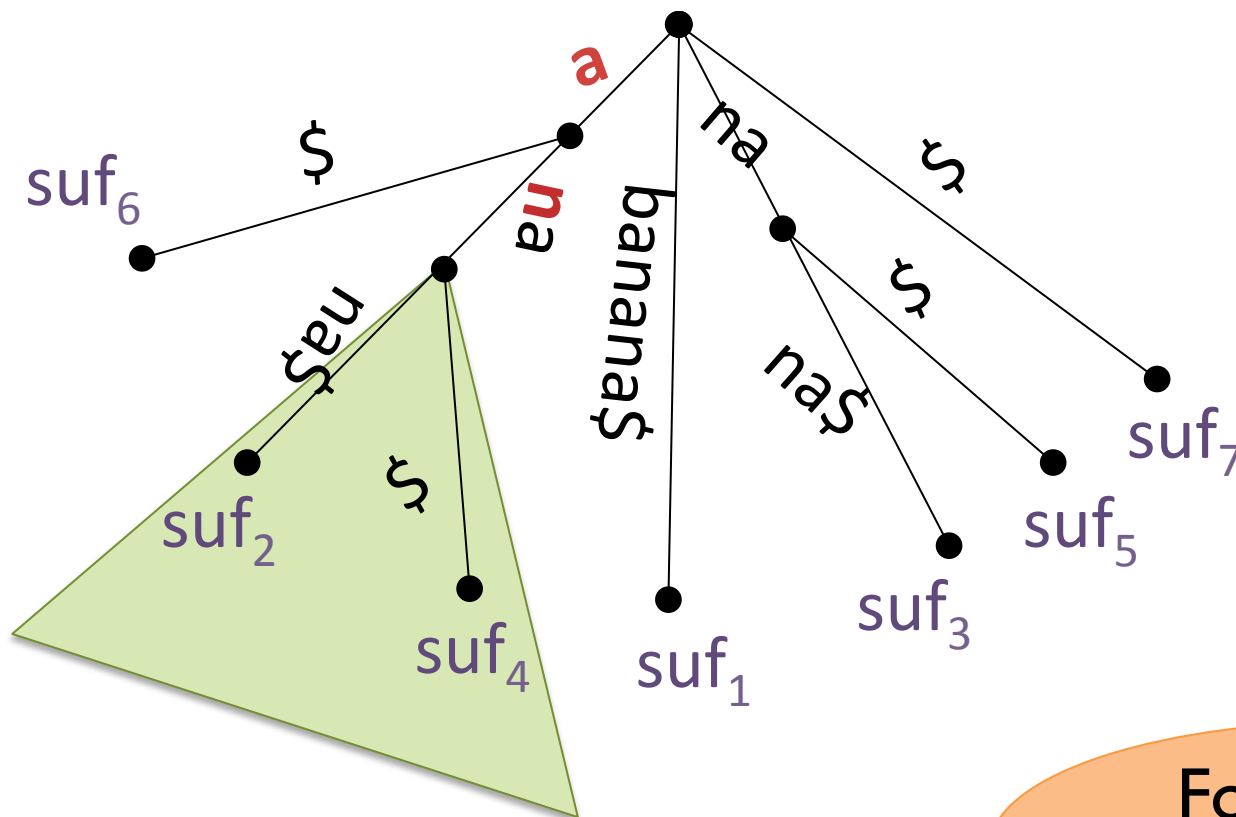
Search for **band**

Not found



Suffix Tree Query

S = banana\$



Search for **an**

Found 2
occurrences

Suffix Tree

- ✧ Many applications in computational biology
- ✧ Linear time construction algorithms

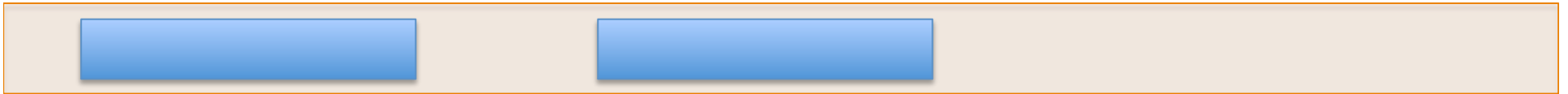
Linear time solutions to

- Genome alignment
- Finding longest common substring
- All-pairs suffix-prefix matching
- **Locating all maximal repetitions**
- And many more...

MEMs

Maximal Exact Match (MEM)

Exact match within sequence that cannot be extended left or right without introducing mismatch.



T **G C A C** **G C A A**

We are interested
in MEMs length $\geq k$

MEMs

Maximal Exact Match (MEM)

Exact match within sequence that cannot be extended left or right without introducing mismatch.

MEMs are **internal nodes** in the suffix tree that have **left-diverse descendants**.

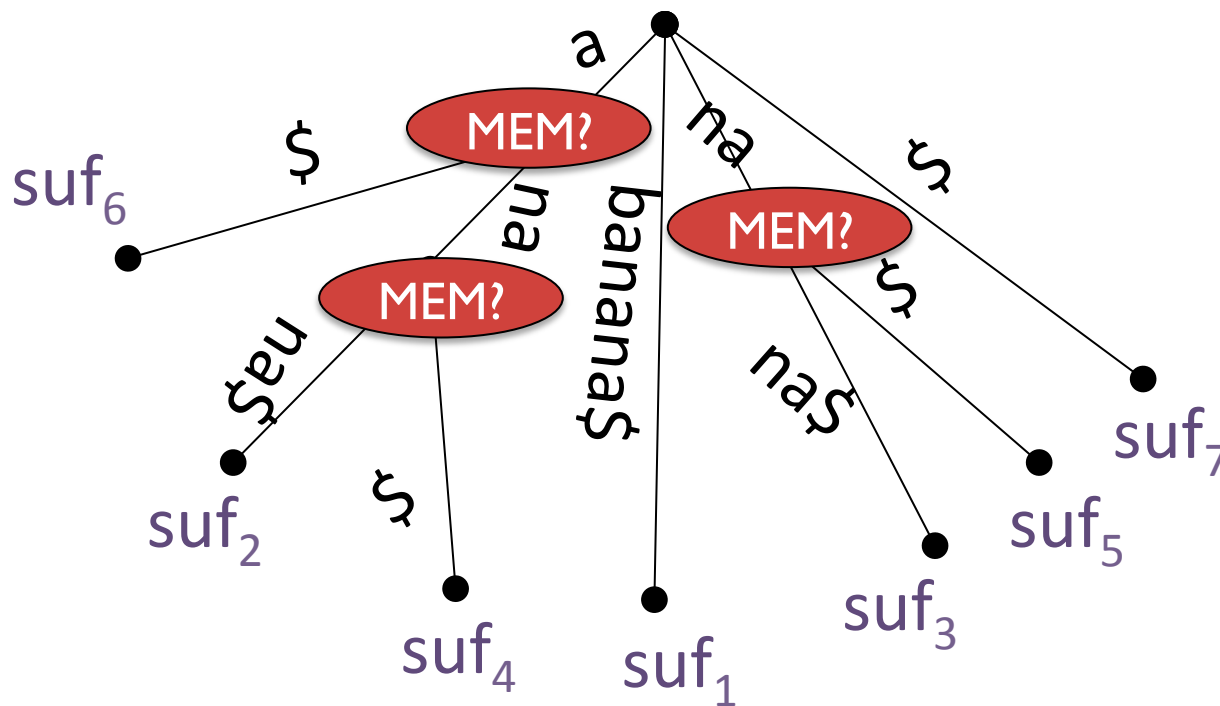
(have descendant leaves that represent suffixes with different characters preceding them)

✧ Linear-time suffix tree traversal to locate MEMs.



MEMs in Suffix Tree

Possible MEMs: a, ana, na



S = banana\$

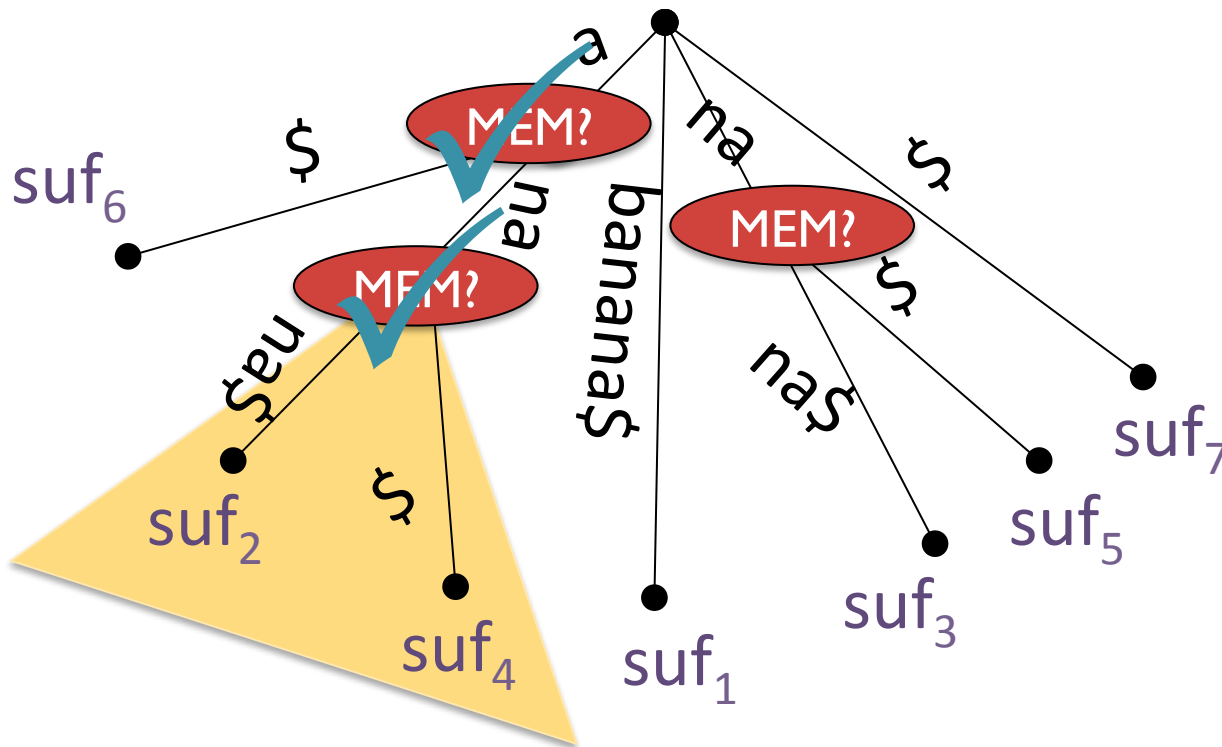
banana\$	suf ₁
anana\$	suf ₂
nana\$	suf ₃
ana\$	suf ₄
na\$	suf ₅
a\$	suf ₆
\$	suf ₇

MEMs are internal nodes in suffix tree with left-diverse descendants



MEMs in Suffix Tree

MEMs: a, ana



S = banana\$

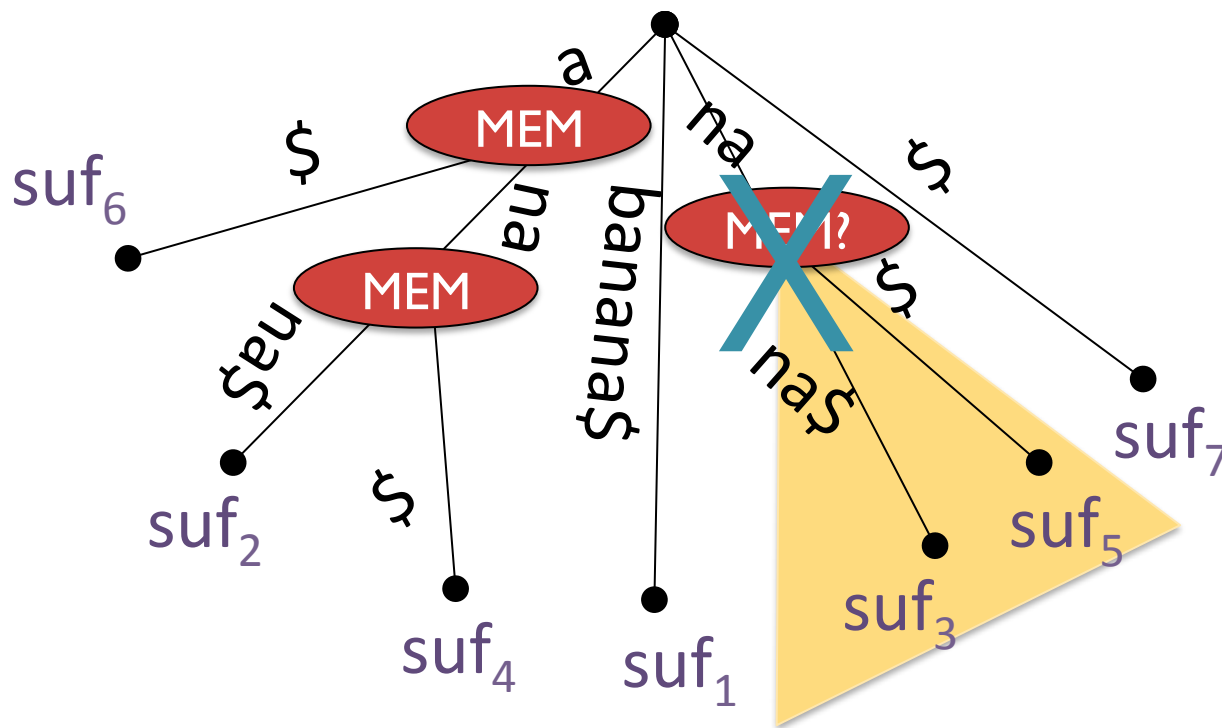
banana\$ suf₂
nana\$ suf₄

MEMs are internal nodes in suffix tree with left-diverse descendants



MEMs in Suffix Tree

MEMs: a, ana



S = banana\$

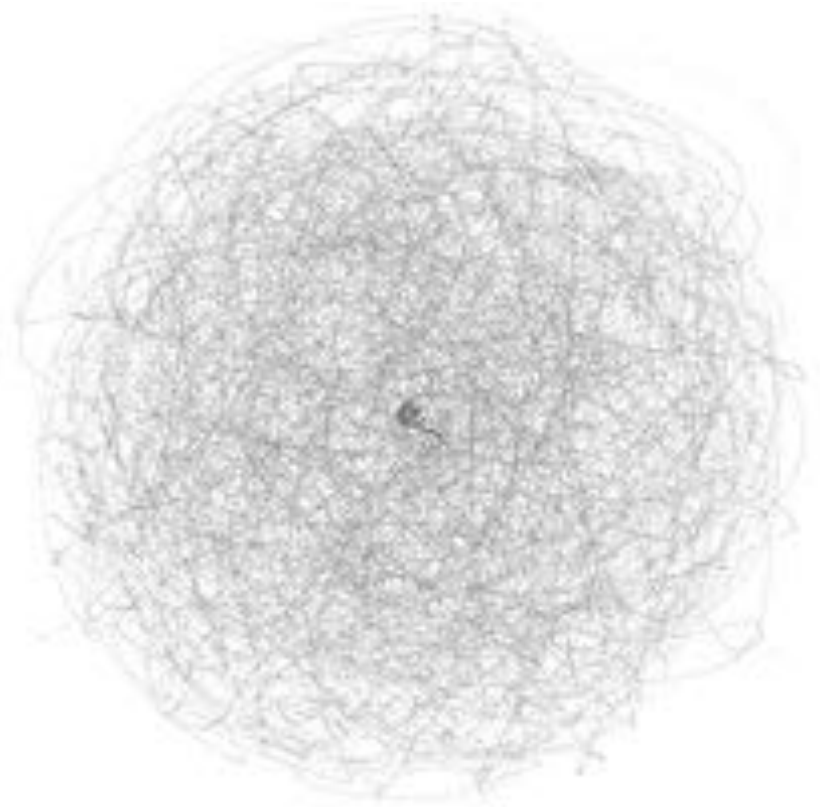
anana\$ suf₃

ana\$ suf₅

MEMs are internal nodes in suffix tree with left-diverse descendants

Outline

- 1 Overview
- 2 Data Structures
- 3 splitMEM Algorithm
- 4 Pan-genome Analysis



Compressed de Bruijn graph



- Types of nodes:
- i. repeatNodes
 - ii. uniqueNodes

Input:

AGAAGTCC\$ATAAGTTA

splitMEM

Nodes in compressed de Bruijn graph classified as

- i. repeatNodes
- ii. uniqueNodes

Algorithm:

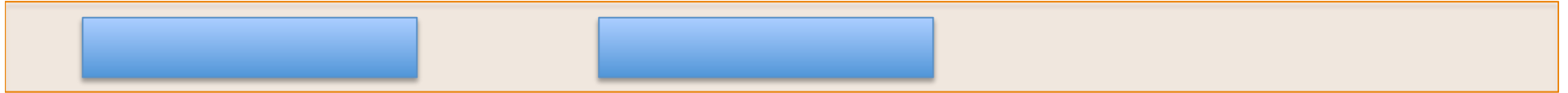
- 1 Construct set of repeatNodes
- 2 Sort start positions of repeatNodes
- 3 Create edges and uniqueNodes to link non-contiguous repeatNodes

repeatNodes

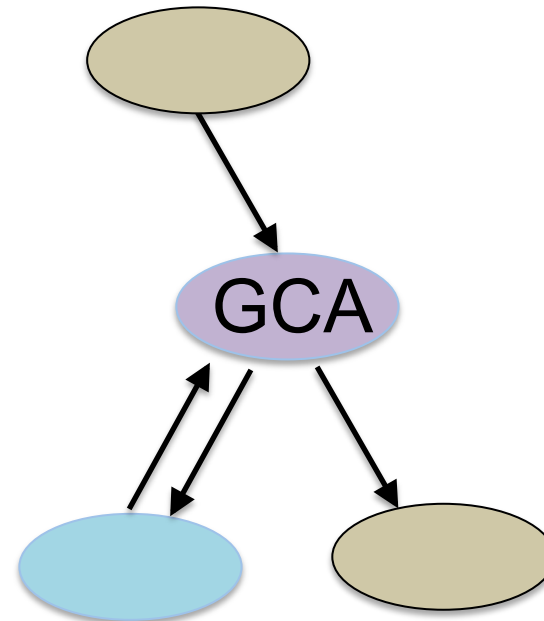
I Construct set of repeatNodes

1. Build suffix tree of genome
2. Mark internal nodes that are MEMs, length $\geq k$
3. Preprocess suffix tree for LMA queries
4. Compute repeatNodes in compressed de Bruijn graph by decomposing MEMs and extracting overlapping components, length $\geq k$

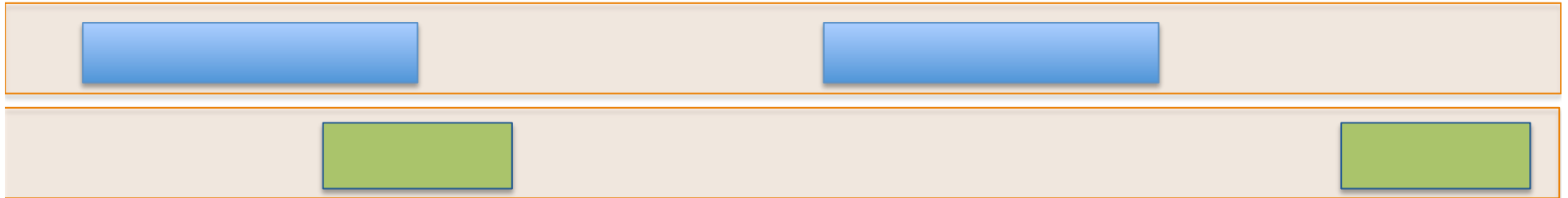
1 MEM occurs twice



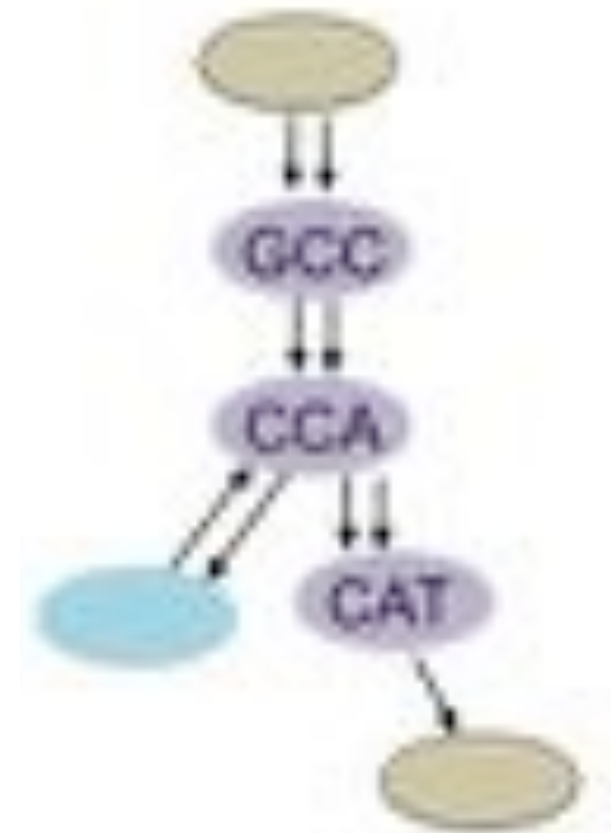
TGCAC...GGCAA



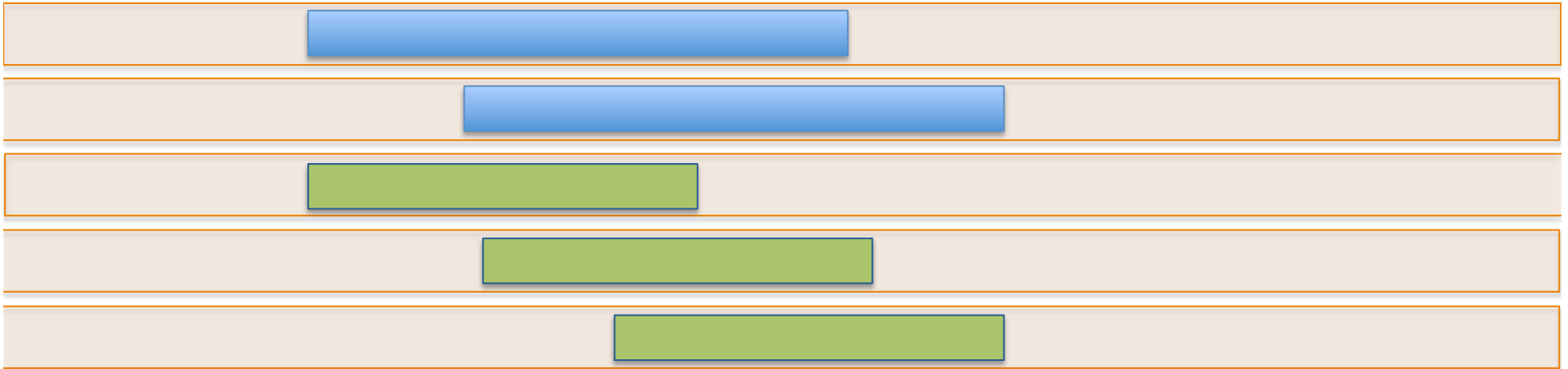
Overlapping MEMs



TGCCATCGCCAACCAT
TGCCATCGCCAACCAT



Tandem Repeat



AGGCTTGGCTTGGCTTGGCTA
AGGCTTGGCTTGGCTTGGCTA
AGGCTTGGCTTGGCTTGGCTA
AGGCTTGGCTTGGCTTGGCTA
AGGCTTGGCTTGGCTTGGCTA



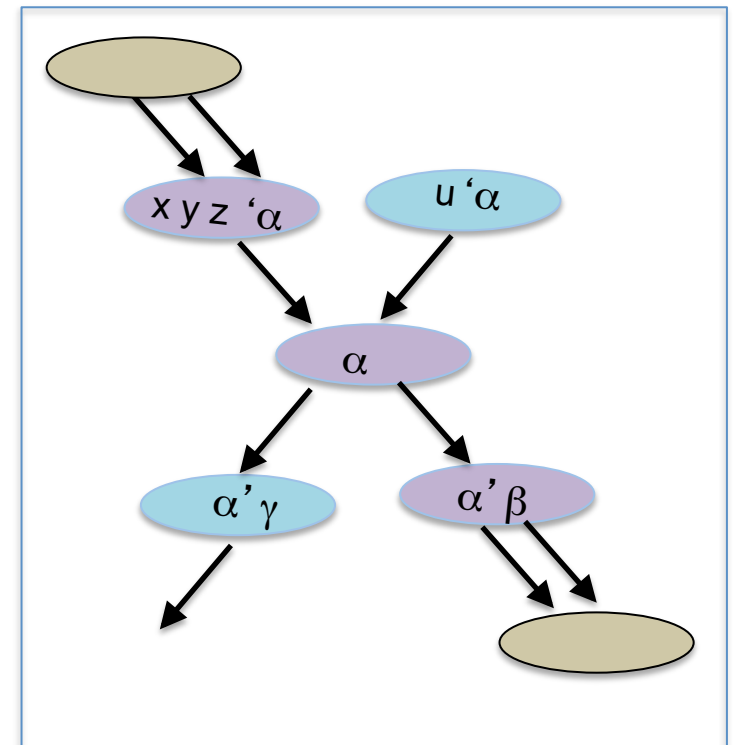
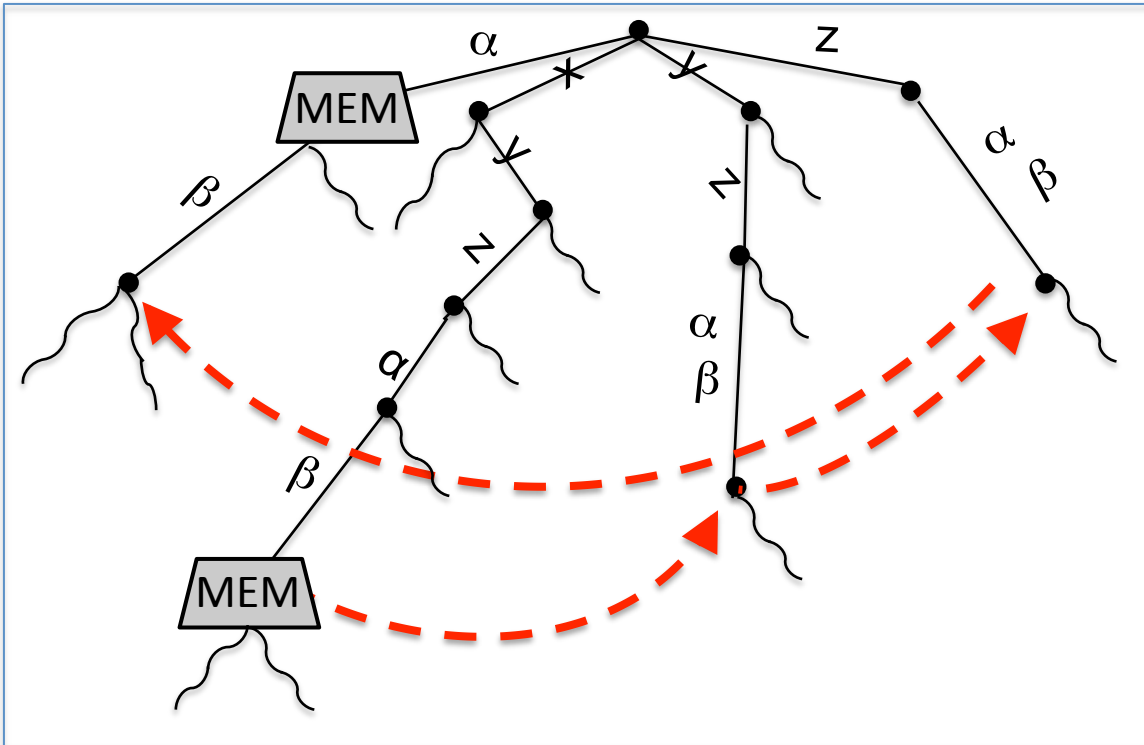
repeatNodes

I Construct set of repeatNodes

1. Build suffix tree of genome
2. Mark internal nodes that are MEMs, length $\geq k$
3. Preprocess suffix tree for LMA queries
4. Compute repeatNodes in compressed de Bruijn graph by decomposing MEMs and extracting overlapping components, length $\geq k$

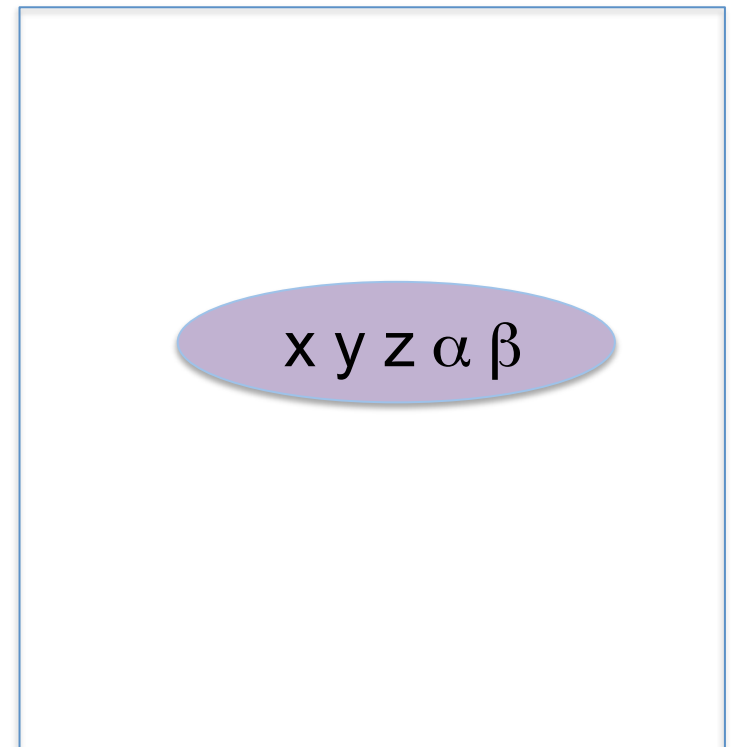
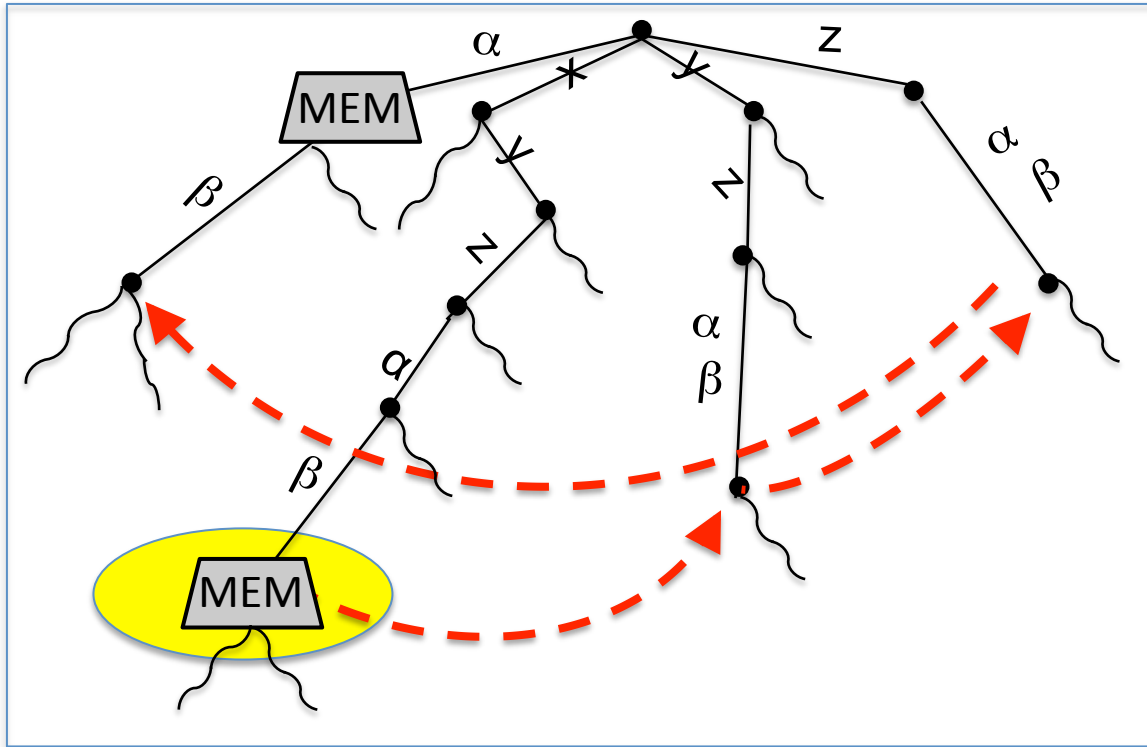
Split MEM to repeatNodes

... x **x y z** **α** **β** ... y **x y z** **α** **β** ... u **α** **γ** ...



Split MEM to repeatNodes

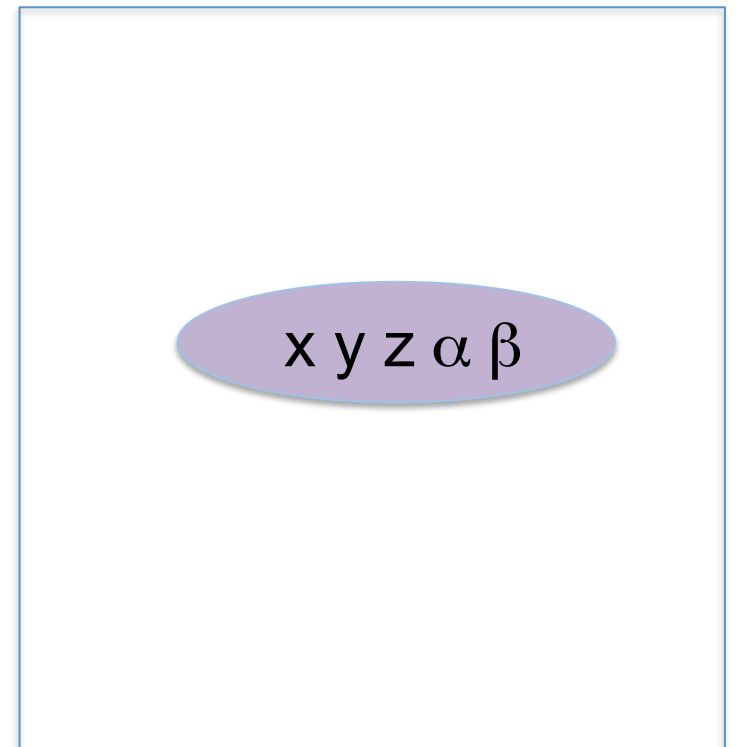
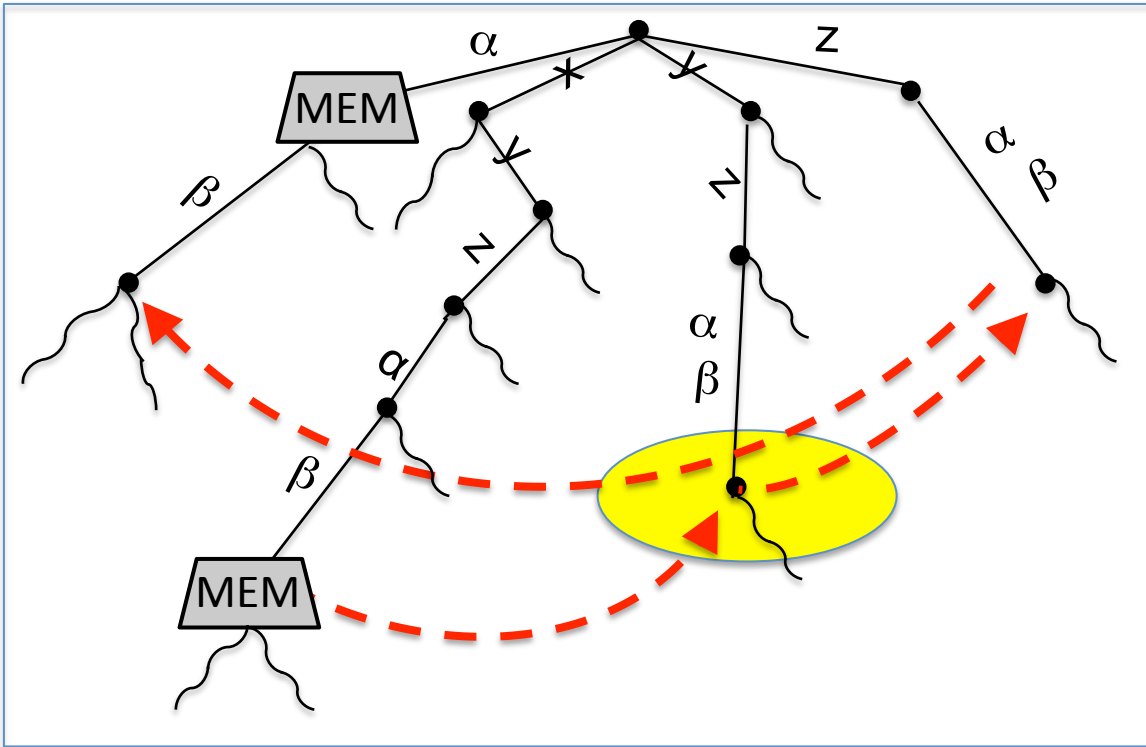
... x x y z α β ... y x y z α β ... u α γ ...



Find MEM in suffix tree.

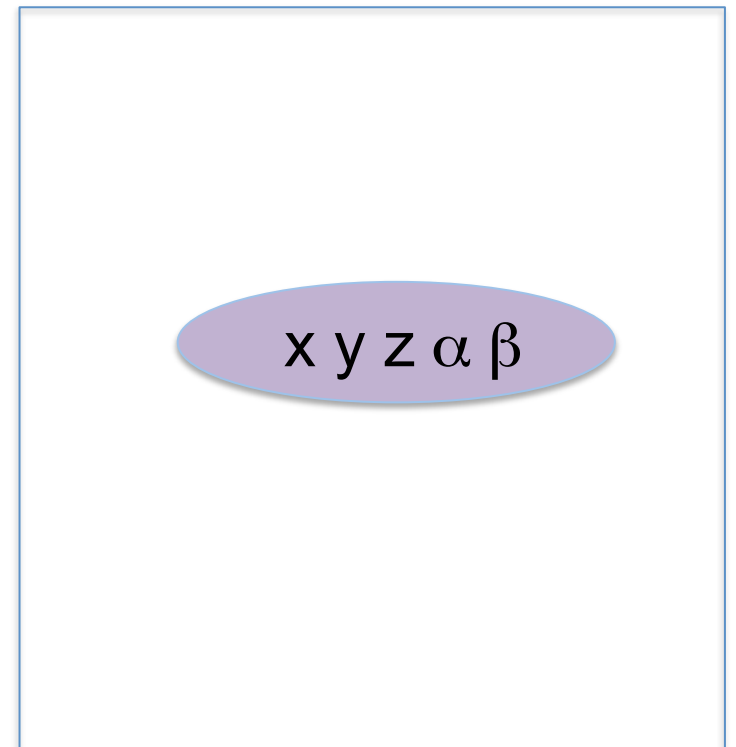
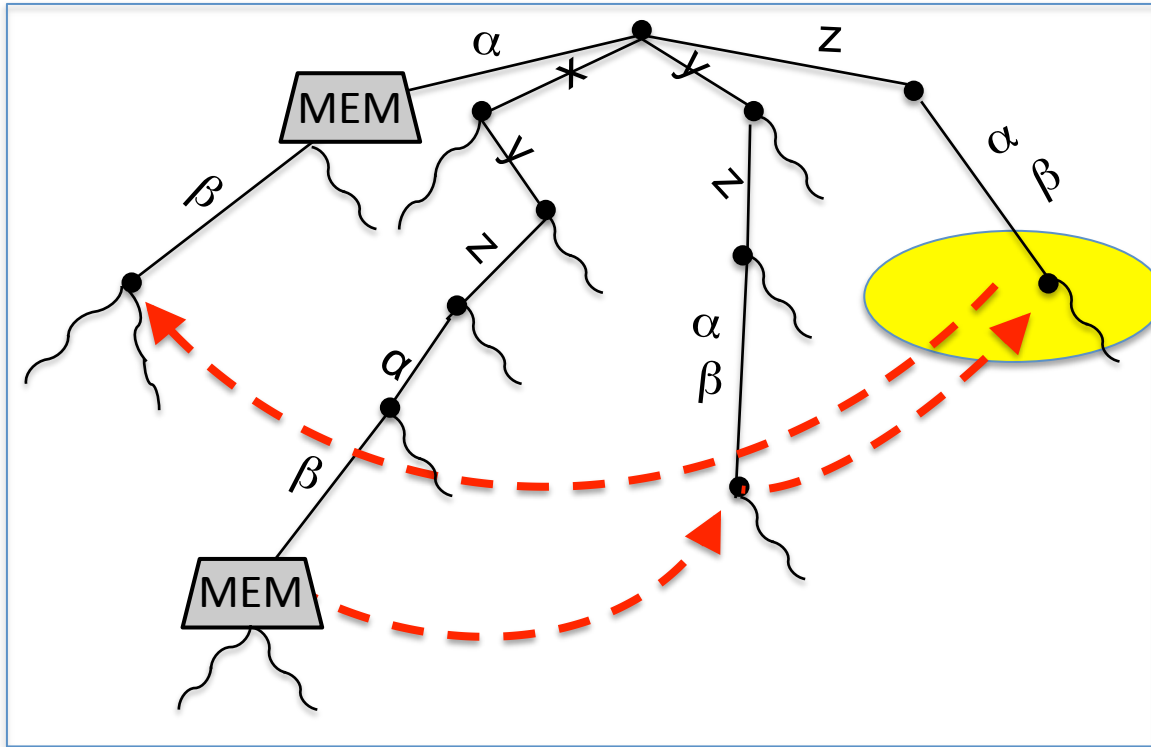
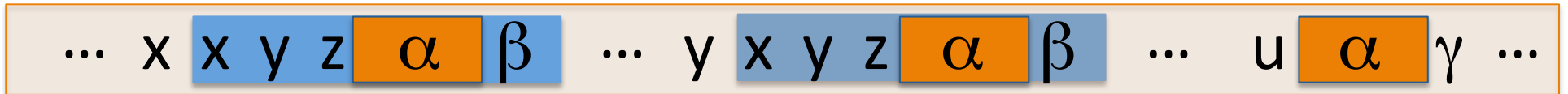
Split MEM to repeatNodes

... x x y z α β ... y x y z α β ... u α γ ...



Traverse suffix link.
Look for MEM as ancestor.

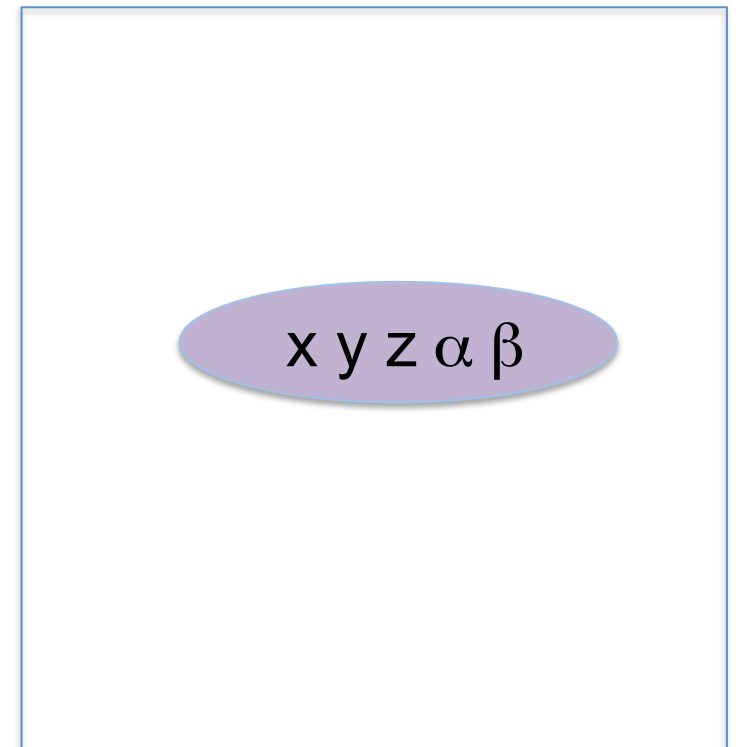
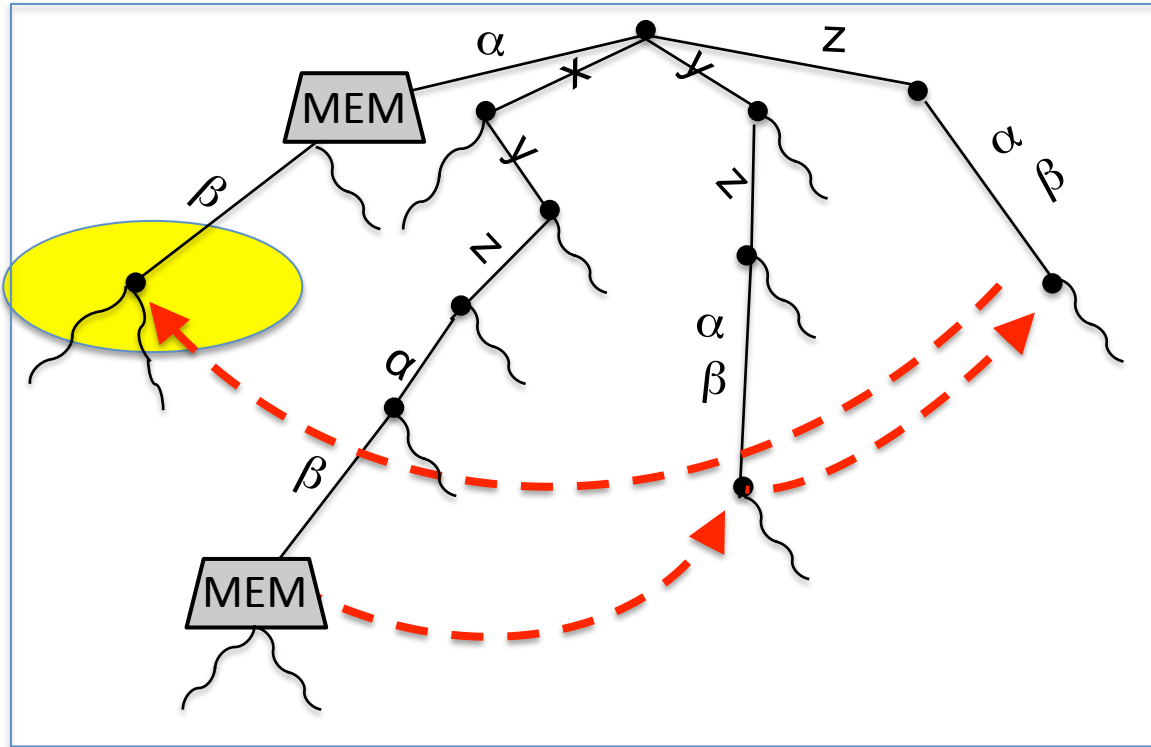
Split MEM to repeatNodes



Traverse suffix link.
Look for MEM as ancestor.

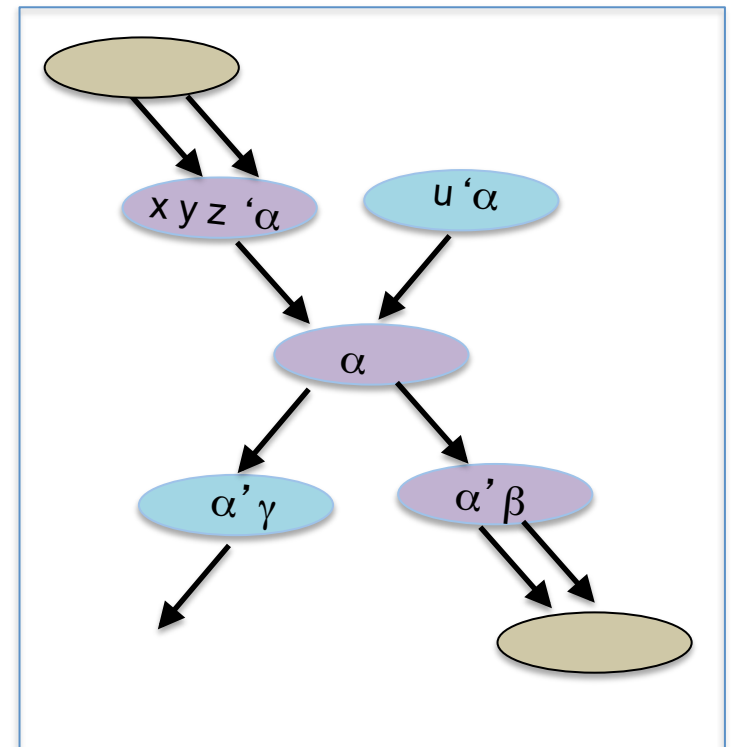
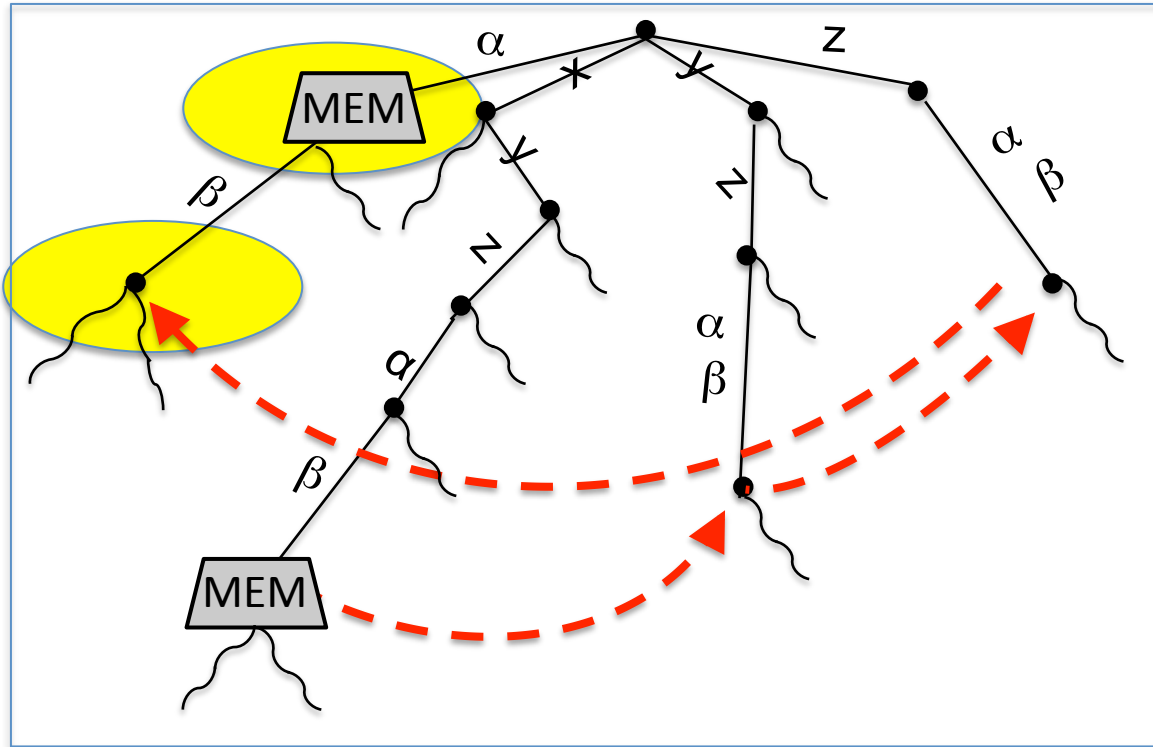
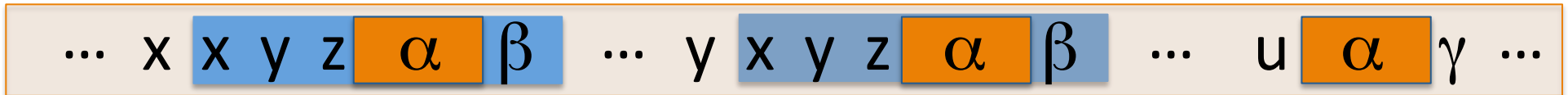
Split MEM to repeatNodes

... x x y z α β ... y x y z α β ... u α γ ...



Traverse suffix link.
Look for MEM as ancestor.

Split MEM to repeatNodes



Found MEM as ancestor. Decompose.

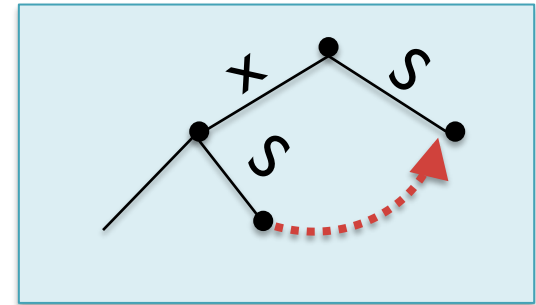
Remove embedded MEM (suffix links). Find next embedded MEM.

Suffix Skips

✧ Reduce $O(n^2)$ time to $O(n \log n)$ time

Suffix link: quickly navigate to distant part of tree

- Pointer from internal node labeled xS to node S
- Trim 1 character in $O(1)$ time
- Trim c characters in $O(c)$ time

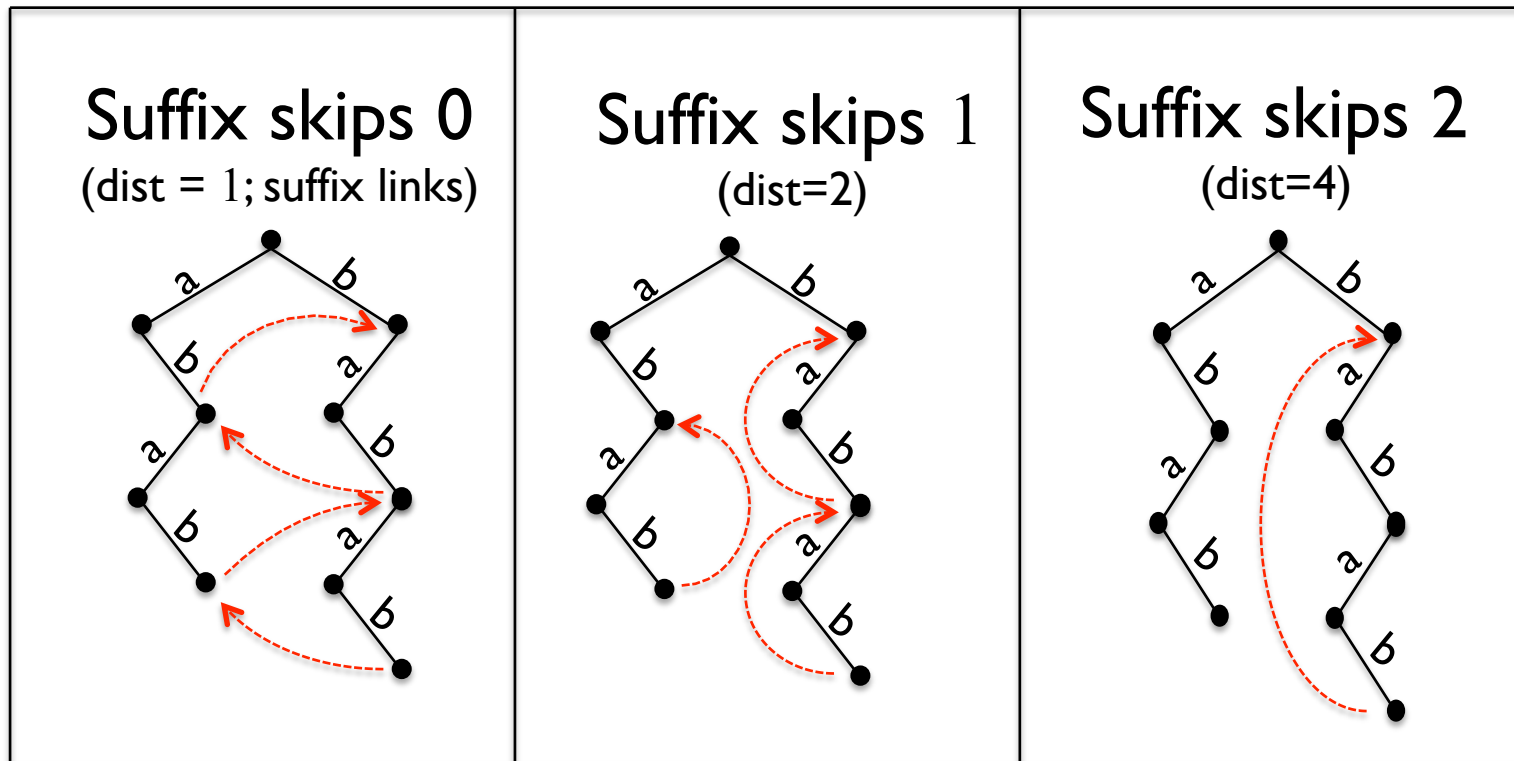


Suffix skip:

- Trim c characters in $O(\log c)$ time

Suffix Skips

Genome: babab



Additional Preprocessing:

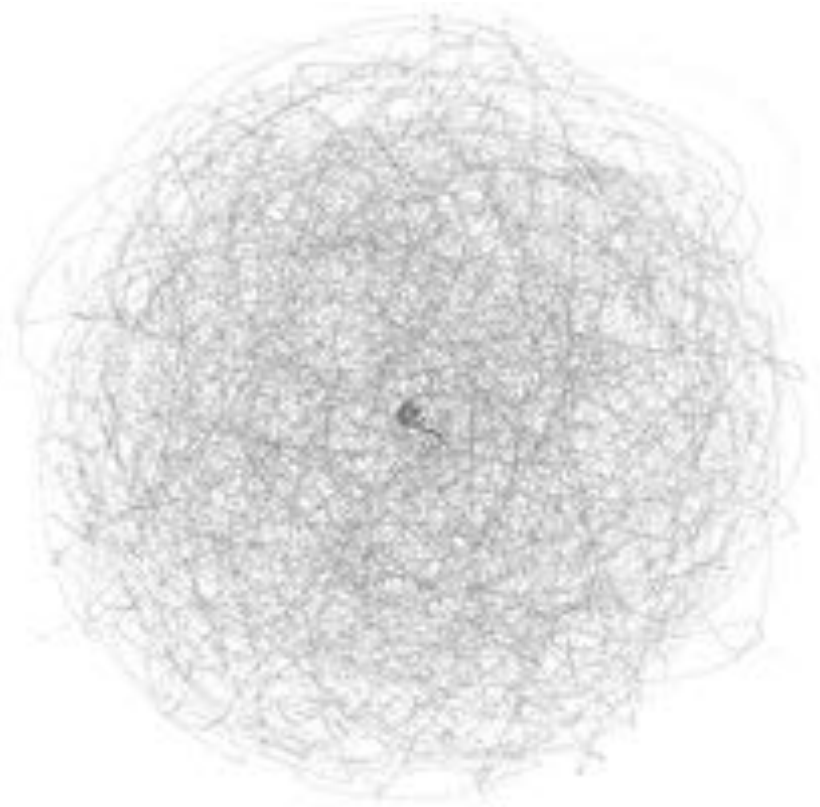
pointer jumping to rapidly add additional links

splitMEM

- splitMEM software
 - C++
 - open source <http://splitmem.sourceforge.net>
- Input modes:
 - single genome: fasta file
 - pan-genome: multi-fasta file
- Multi k -mer
construct several compressed de Bruijn graphs
without rebuilding suffix tree

Outline

- 1 Overview
- 2 Data Structures
- 3 splitMEM Algorithm
- 4 Pan-genome Analysis



Pan-genome analysis

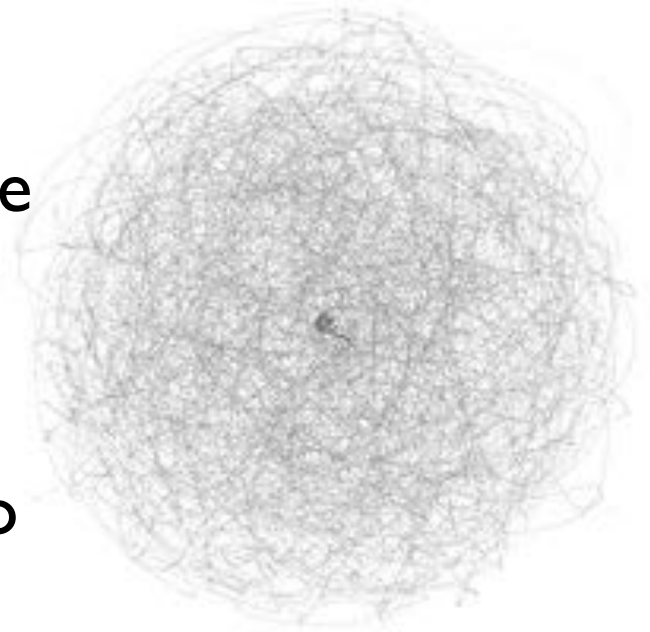
B. Anthracis and *E. coli*

Examine graph properties:

- Number nodes, edges, avg. degree
- Node length distribution
- Genome sharing among nodes
- Distribution of node distances to core genome

Other properties that can be studied:

- Girth, Diameter, Modularity, Network Motifs, etc.
- Functional enrichment of highly conserved or genome specific genes.



Pan-genome analysis

Strain	Size	Accession
<i>B. anthracis</i> A0246 uid11543	3176 kb	CP001508
<i>B. anthracis</i> A168 uid40103	3179 kb	CP001974
<i>B. anthracis</i> A16 uid40100	3179 kb	CP001970
<i>B. anthracis</i> Ames 0801 uid10784	3178 kb	AJ001714
<i>B. anthracis</i> Ames uid309	3178 kb	AJ0014879
<i>B. anthracis</i> CDC 484 uid11329	3181 kb	CP001215
<i>B. anthracis</i> CI uid24309	3147 kb	CP001746
<i>B. anthracis</i> HP401 uid49561	3179 kb	CP002093
<i>B. anthracis</i> str. Sterne uid10874	3180 kb	AJ0017215
<i>E. coli</i> BL21 DE3 uid12111	4519 kb	FM182948
<i>E. coli</i> 042 uid10647	5195 kb	FN54766
<i>E. coli</i> 536 uid16215	4895 kb	CP001247
<i>E. coli</i> 5399 uid14113	5107 kb	CLN28145
<i>E. coli</i> ABU 83072 uid28725	5063 kb	CP001671
<i>E. coli</i> APEC 04 uid10718	5034 kb	CP001648
<i>E. coli</i> APEC 078 uid142188	4753 kb	CP001609
<i>E. coli</i> BL21 DE3 uid20713	4516 kb	CP001509
<i>E. coli</i> BL21 DE3 uid29965	4516 kb	AB014698

Pan-genome analysis

Graphs of main chromosomes

- 9 strains of *Bacillus anthracis*
- Selection of 9 strains of *Escherichia coli*

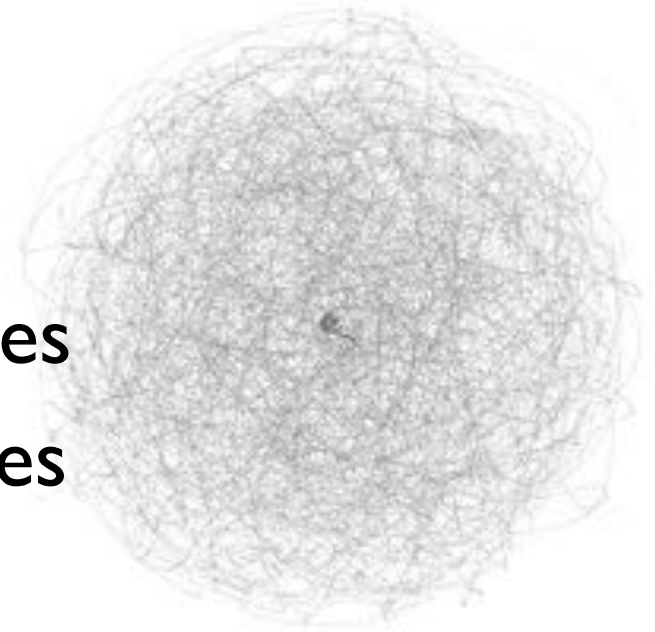
Species	K	Nodes	Edges	Avg. Degree
B. anthracis	25	103926	138468	1.33
B. anthracis	100	41343	54954	1.32
B. anthracis	1000	6627	8659	1.30
E. coli	25	494783	662081	1.33
E. coli	100	230996	308256	1.33
E. coli	1000	11900	15695	1.31

Pan-genome analysis

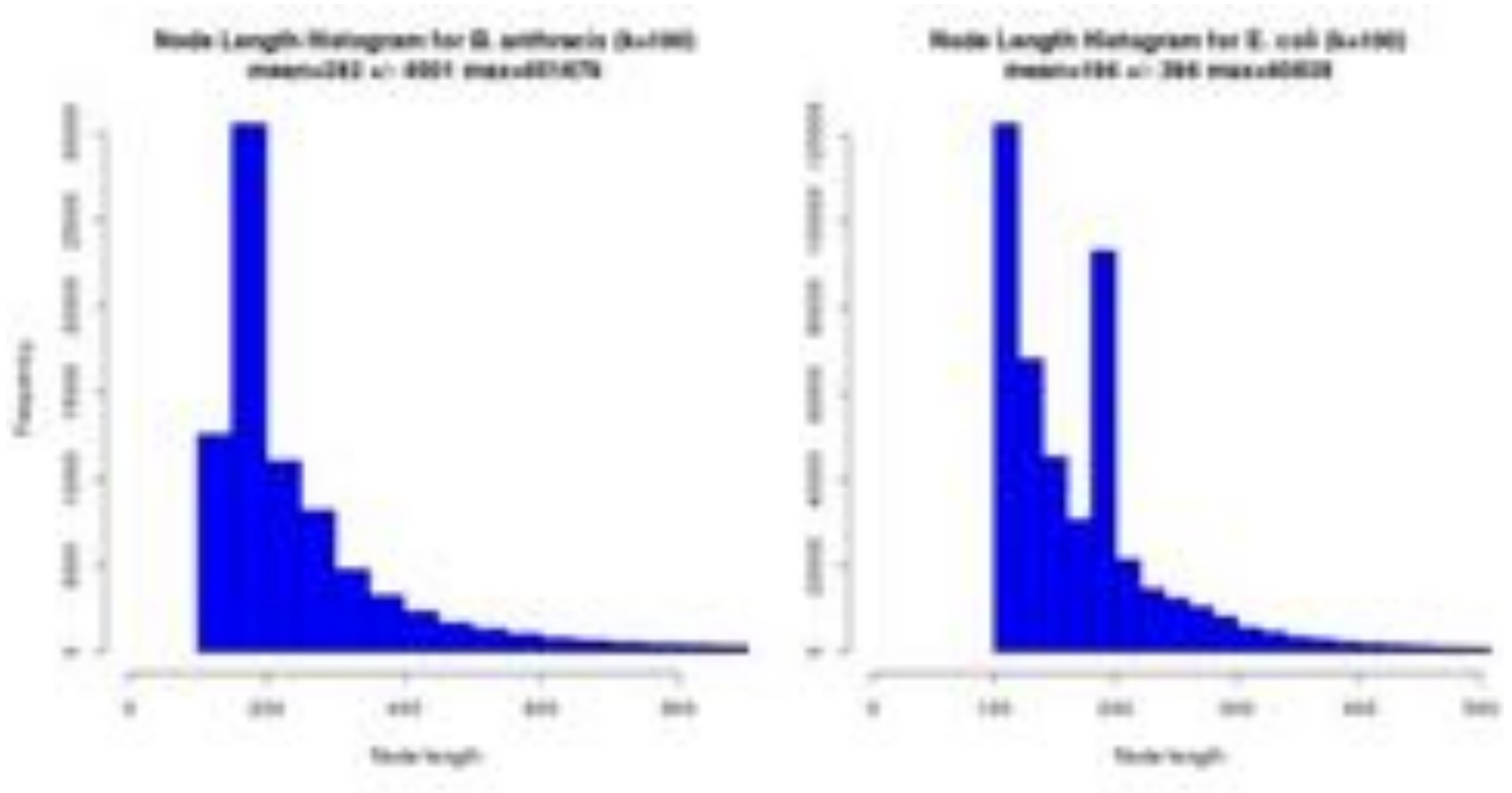
B. Anthracis and *E. coli*

Examine graph properties

- Node length distribution
- Genome sharing among nodes
- Distribution of node distances to core genome

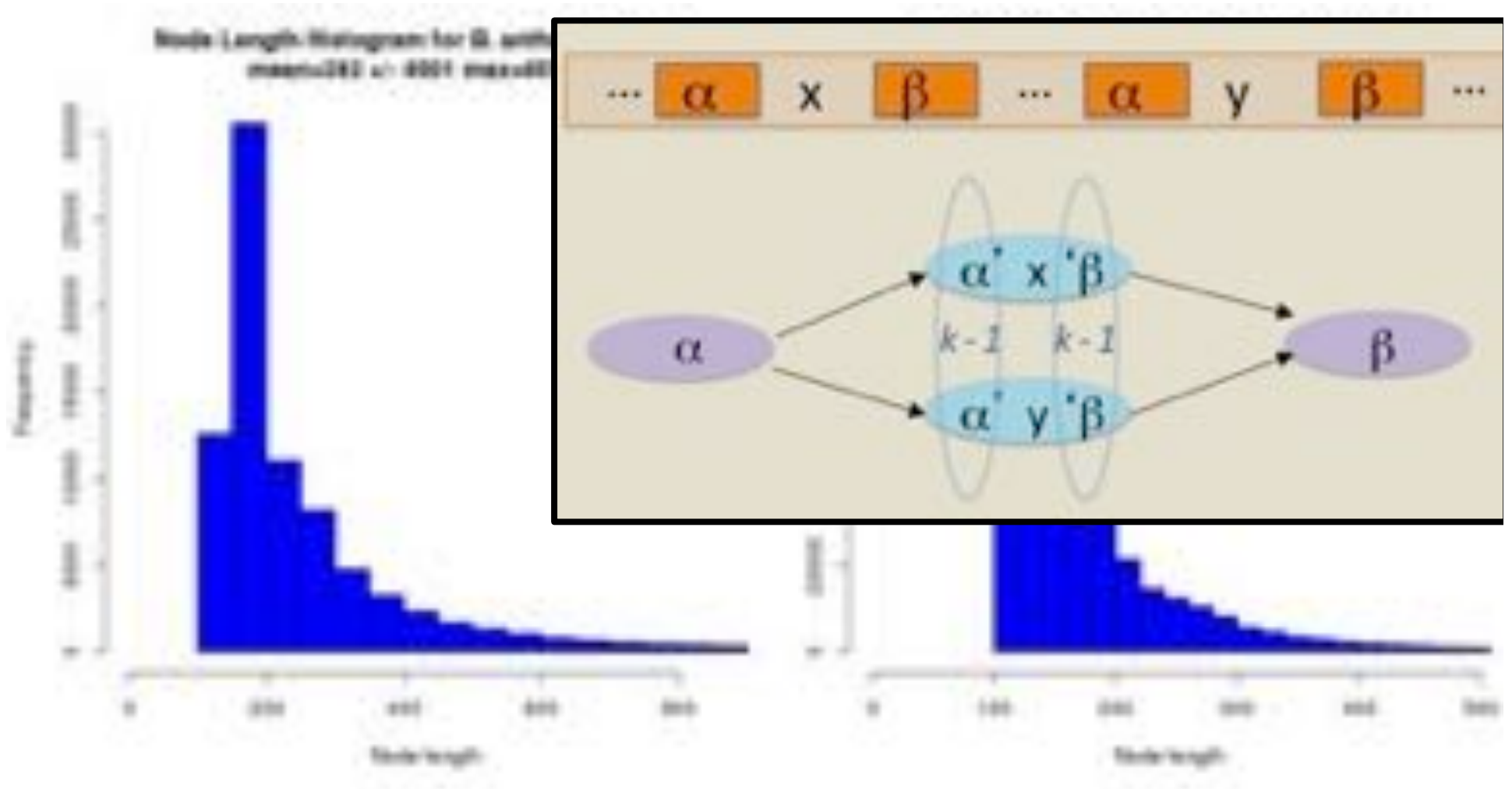


Pan-genome analysis



Histogram of Node Lengths

Pan-genome analysis



Histogram of Node Lengths

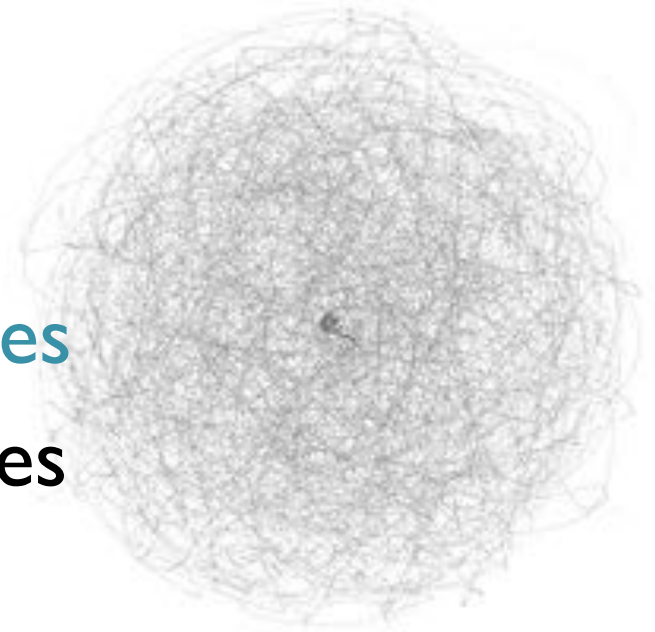
Spike at 2k:
SNPs

Pan-genome analysis

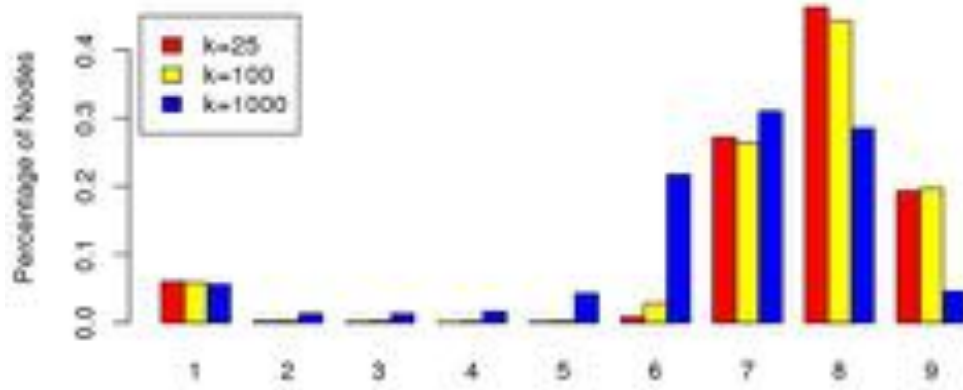
B. Anthracis and *E. coli*

Examine graph properties

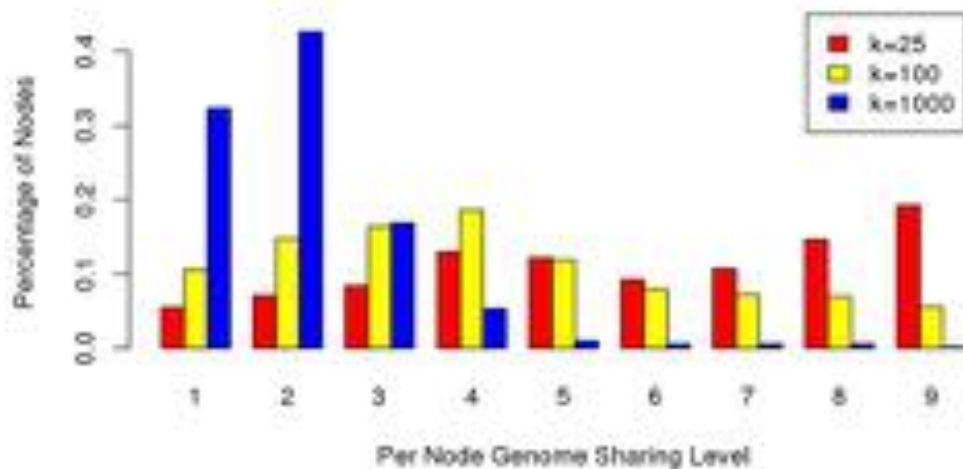
- Node length distribution
- Genome sharing among nodes
- Distribution of node distances to core genome



Pan-genome analysis



B. anthracis



E. coli

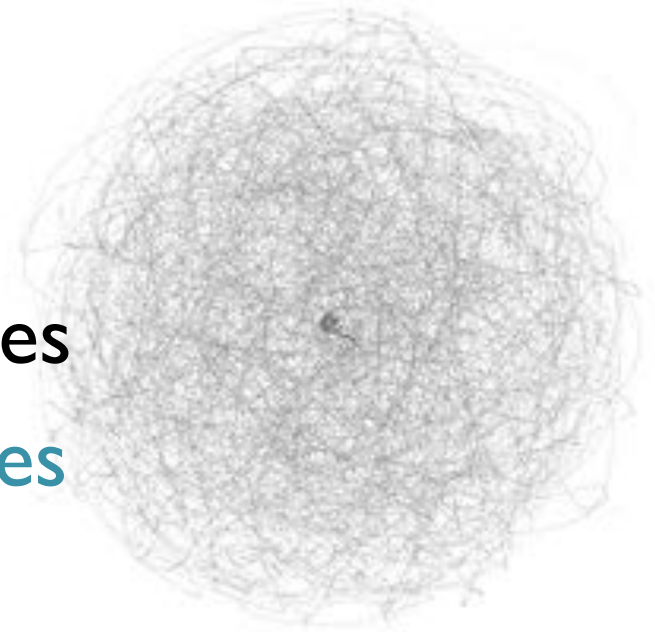
Fraction of nodes with each level of genome sharing

Pan-genome analysis

B. Anthracis and *E. coli*

Examine graph properties

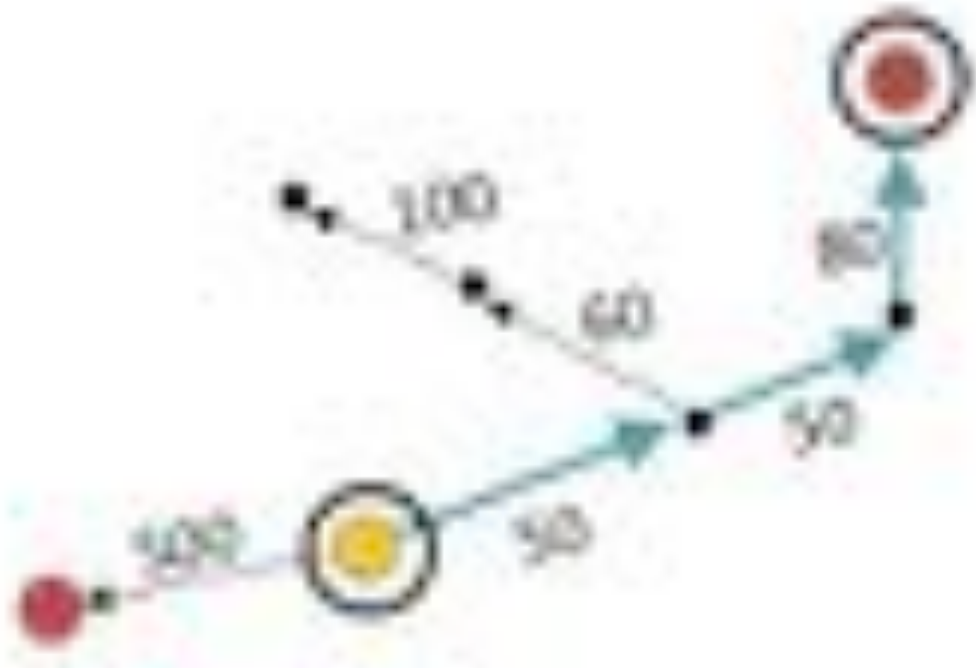
- Node length distribution
- Genome sharing among nodes
- Distribution of node distances to core genome



Pan-genome analysis

Graph encodes sequence **context** of segments.

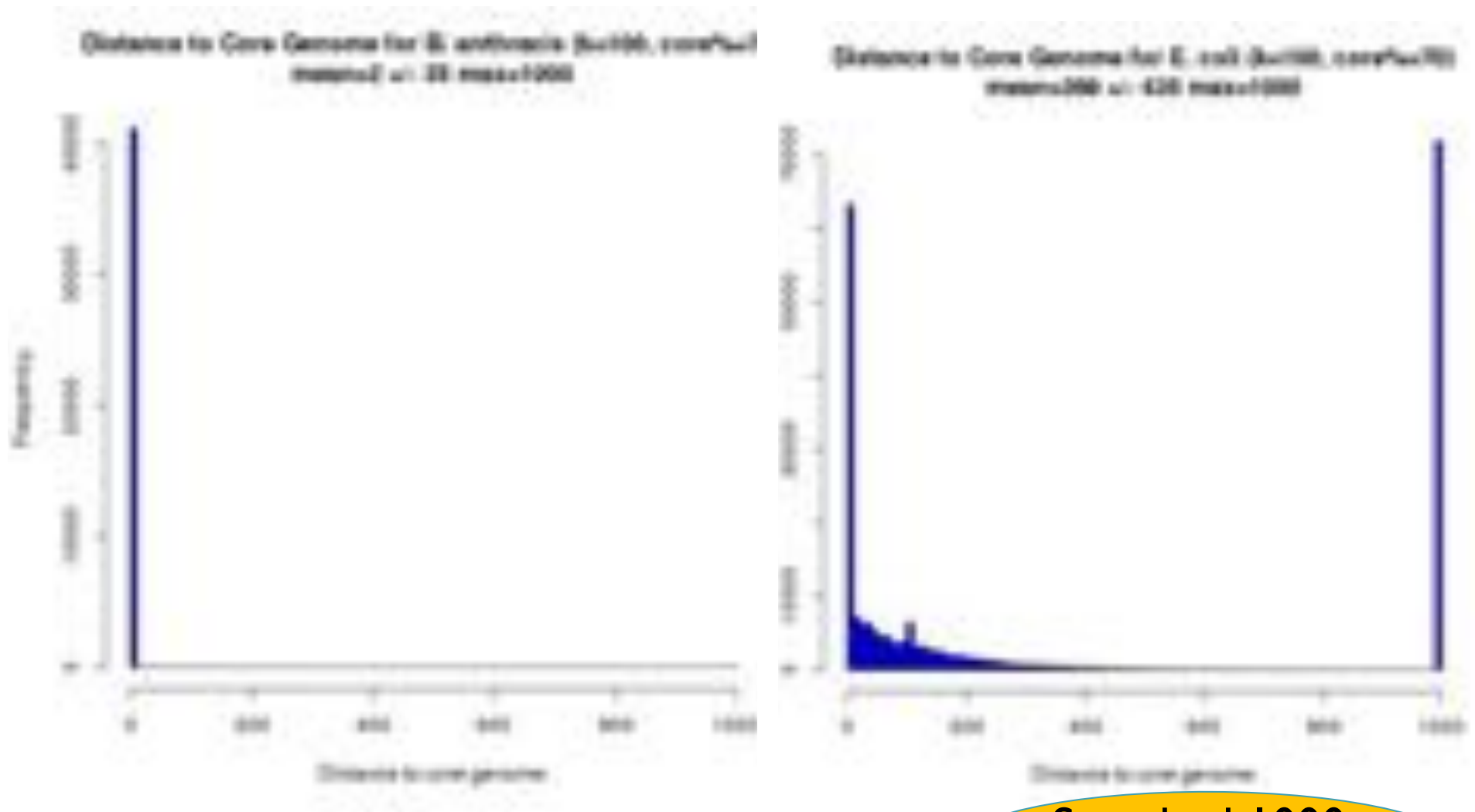
Core genome: subsequences that occur in at least 70% of underlying genomes.



Branch and
Bound Search

Nodes can be further in terms of hops while closer by base pairs.

Pan-genome analysis



Node distances to core genome

Searched 1000-hop radius

Summary

- Identify pan-genome relationships graphically.
- Topological relationship between suffix tree and compressed de Bruijn graph.
- Direct construction of compressed de Bruijn graph for single or pan-genome.
- Introduce suffix skips.
- Explore pan-genome graphs of *B. anthracis*, *E. coli*.

SplitMEM: Graphical pan-genome analysis with suffix skips.

Marcus, S, Lee, H, Schatz, MC (2014) *BioRxiv*

<http://biorxiv.org/content/early/2014/04/06/003954>

Future work

Improve splitMEM software:

- Reduce space using compressed full-text index instead of suffix tree
- Approximate indexing of strains to form a pan-genome graph
- Alignment of reads to pan-genome

Biological applications:

- Functional enrichment of core-genome and genome specific segments
- Expand study to larger collection of microbes and larger genomes

Acknowledgments

Michael Schatz

Hayan Lee

Giuseppe Narzisi

James Gurtowski

Schatz Lab

IT department

Todd Heywood



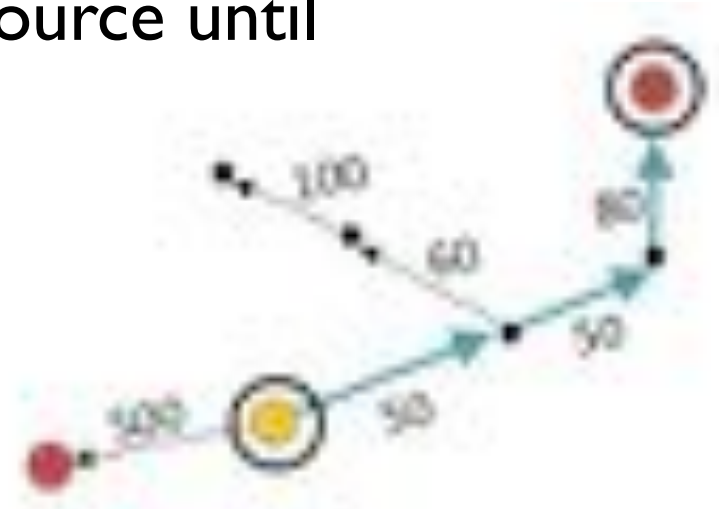
Thank You!

Pan-genome analysis

Branch and bound search (like Dijkstra's shortest path algorithm) to compute bp distance from each non-core node to core genome:

Traverse all distinct paths from source until

- a core node is reached
- OR
- current node was visited by a shorter path



Bounded search

once a core node is found, its distance bounds maximum search distance along other paths