

Fully-Compressed Suffix Trees

Luís M. S. Russo

CITI (FCT/UNL)
lsm@di.fct.unl.pt

O Gosto pela Matemática - Uma Década de Talentos

Outline

- 1 Motivation
 - Exact Matching
- 2 Suffix Tree Representation
 - Classical Representation
 - Modern Representations
- 3 Conclusions
 - Summary

Pattern Matching Problem

29 min

Problem (Exact Matching)

Find pattern P in text T .

Example

- 1 $T = \text{acctg}c\text{gctagct}$, $n = 13$, $\sigma = 4$
- 2 $\text{acctg}c\text{gctagct}$, $P = c$, $m = 1$, $\text{occ} = 5$
- 3 $\text{acctg}c\text{gctagct}$, $P = \text{cct}$, $m = 3$, $\text{occ} = 1$

Pattern Matching Problem

29 min

Problem (Exact Matching)

Find pattern P in text T .

Example

- 1 $T = acctgcgctagct$, $n = 13$, $\sigma = 4$
- 2 $acctg**cg**ctag**ct**$, $P = c$, $m = 1$, $occ = 5$
- 3 $acctg**cg**ctagct$, $P = cct$, $m = 3$, $occ = 1$

Pattern Matching Problem

29 min

Problem (Exact Matching)

Find pattern P in text T .

Example

- 1 $T = acctgcgctagct$, $n = 13$, $\sigma = 4$
- 2 $acctg**cg**ctag**ct**$, $P = c$, $m = 1$, $occ = 5$
- 3 $ac**ct**gcgctagct$, $P = cct$, $m = 3$, $occ = 1$

Text Database Problems

27 min

Problem (On-line Limitations)

On-line solutions are not viable for large text databases, $O(m + n)$.

Example (On-line scan)

- Scanning 3Gb DNA sequences takes too long.
- $T = \text{acctgcgctagct}$
- We need indexes that have $O(m + occ)$ time.
- $P = \text{cct}$

Text Database Problems

27 min

Problem (On-line Limitations)

On-line solutions are not viable for large text databases, $O(m + n)$.

Example (On-line scan)

- Scanning 3Gb DNA sequences takes too long.
- $T = \text{acctgcgctagct}$
- We need indexes that have $O(m + occ)$ time.
- $P = \text{cct}$

Text Database Problems

27 min

Problem (On-line Limitations)

On-line solutions are not viable for large text databases, $O(m + n)$.

Example (On-line scan)

- Scanning 3Gb DNA sequences takes too long.
- $T = \text{acctgcgctagct}$
- We need indexes that have $O(m + occ)$ time.
- $P = \text{cct}$

Text Database Problems

27 min

Problem (On-line Limitations)

On-line solutions are not viable for large text databases, $O(m + n)$.

Example (On-line scan)

- Scanning 3Gb DNA sequences takes too long.
- $T = \text{acctgcgctagct}$
- We need indexes that have $O(m + occ)$ time.
- $P = \text{cct}$

Text Database Problems

27 min

Problem (On-line Limitations)

On-line solutions are not viable for large text databases, $O(m + n)$.

Example (On-line scan)

- Scanning 3Gb DNA sequences takes too long.
- $T = \text{a}cctg\text{c}gctagct$
- We need indexes that have $O(m + occ)$ time.
- $P = cct$

Text Database Problems

27 min

Problem (On-line Limitations)

On-line solutions are not viable for large text databases, $O(m + n)$.

Example (On-line scan)

- Scanning 3Gb DNA sequences takes too long.
- $T = acctg*cgctagct*$
- We need indexes that have $O(m + occ)$ time.
- $P = cct$

Text Database Problems

27 min

Problem (On-line Limitations)

On-line solutions are not viable for large text databases, $O(m + n)$.

Example (On-line scan)

- Scanning 3Gb DNA sequences takes too long.
- $T = acctg*cgctagct*$
- We need indexes that have $O(m + occ)$ time.
- $P = cct$

Text Database Problems

27 min

Problem (On-line Limitations)

On-line solutions are not viable for large text databases, $O(m + n)$.

Example (On-line scan)

- Scanning 3Gb DNA sequences takes too long.
- $T = \text{acctgcgctagct}$
- We need indexes that have $O(m + occ)$ time.
- $P = \text{cct}$

Text Database Problems

27 min

Problem (On-line Limitations)

On-line solutions are not viable for large text databases, $O(m + n)$.

Example (On-line scan)

- Scanning 3Gb DNA sequences takes too long.
- $T = acctg\color{red}cgctagct$
- We need indexes that have $O(m + occ)$ time.
- $P = cct$

Text Database Problems

27 min

Problem (On-line Limitations)

On-line solutions are not viable for large text databases, $O(m + n)$.

Example (On-line scan)

- Scanning 3Gb DNA sequences takes too long.
- $T = acctgcgctagct$
- We need indexes that have $O(m + occ)$ time.
- $P = cct$

Text Database Problems

27 min

Problem (On-line Limitations)

On-line solutions are not viable for large text databases, $O(m + n)$.

Example (On-line scan)

- Scanning 3Gb DNA sequences takes too long.
- $T = \text{acctg}c\text{gctagct}$
- We need indexes that have $O(m + occ)$ time.
- $P = \text{cct}$

Text Database Problems

27 min

Problem (On-line Limitations)

On-line solutions are not viable for large text databases, $O(m + n)$.

Example (On-line scan)

- Scanning 3Gb DNA sequences takes too long.
- $T = \text{acctg}cgct\text{tagct}$
- We need indexes that have $O(m + occ)$ time.
- $P = \text{cct}$

Text Database Problems

27 min

Problem (On-line Limitations)

On-line solutions are not viable for large text databases, $O(m + n)$.

Example (On-line scan)

- Scanning 3Gb DNA sequences takes too long.
- $T = \text{acctg}cgct\text{agct}$
- We need indexes that have $O(m + occ)$ time.
- $P = \text{cct}$

Text Database Problems

27 min

Problem (On-line Limitations)

On-line solutions are not viable for large text databases, $O(m + n)$.

Example (On-line scan)

- Scanning 3Gb DNA sequences takes too long.
- $T = \text{acctg}cgctagct$
- We need indexes that have $O(m + occ)$ time.
- $P = cct$

Text Database Problems

27 min

Problem (On-line Limitations)

On-line solutions are not viable for large text databases, $O(m + n)$.

Example (On-line scan)

- Scanning 3Gb DNA sequences takes too long.
- $T = \text{acctgcgctagct}$
- We need indexes that have $O(m + occ)$ time.
- $P = \text{cct}$

Text Database Problems

27 min

Problem (On-line Limitations)

On-line solutions are not viable for large text databases, $O(m + n)$.

Example (On-line scan)

- Scanning 3Gb DNA sequences takes too long.
- $T = \text{acctg}cgctagct$
- We need indexes that have $O(m + occ)$ time.
- $P = cct$

Text Database Problems

27 min

Problem (On-line Limitations)

On-line solutions are not viable for large text databases, $O(m + n)$.

Example (On-line scan)

- Scanning 3Gb DNA sequences takes too long.
- $T = acctgcgctagct$
- We need indexes that have $O(m + occ)$ time.
- $P = cct$

Text Database Problems

27 min

Problem (On-line Limitations)

On-line solutions are not viable for large text databases, $O(m + n)$.

Example (On-line scan)

- Scanning 3Gb DNA sequences takes too long.
- $T = acctgcgctagct$
- We need indexes that have $O(m + occ)$ time.
- $P = cct$

Text Database Problems

27 min

Problem (On-line Limitations)

On-line solutions are not viable for large text databases, $O(m + n)$.

Example (On-line scan)

- Scanning 3Gb DNA sequences takes too long.
- $T = acctgcgctagct$
- We need indexes that have $O(m + occ)$ time.
- $P = cct$

Text Database Problems

27 min

Problem (On-line Limitations)

On-line solutions are not viable for large text databases, $O(m + n)$.

Example (On-line scan)

- Scanning 3Gb DNA sequences takes too long.
- $T = acctgcgctagct$
- We need indexes that have $O(m + occ)$ time.
- $P = cct$

Text Database Problems

27 min

Problem (On-line Limitations)

On-line solutions are not viable for large text databases, $O(m + n)$.

Example (On-line scan)

- Scanning 3Gb DNA sequences takes too long.
- $T = acctgcgctagct$
- We need indexes that have $O(m + occ)$ time.
- $P = cct$

The Suffix Tree

26 min

- Suffix trees have optimal $O(m + occ)$ performance.
- What is a suffix tree ?
- Gather the suffixes & build a tree.

Example (Suffix Tree for *atttat*)

```
atttat$  
tztat$  
ttat$  
tat$  
at$  
t$  
$
```

The Suffix Tree

26 min

- Suffix trees have optimal $O(m + occ)$ performance.
- What is a suffix tree ?
- Gather the suffixes & build a tree.

Example (Suffix Tree for *atttat*)

```
atttat$  
tztat$  
ttat$  
tat$  
at$  
t$  
$
```

The Suffix Tree

26 min

- Suffix trees have optimal $O(m + occ)$ performance.
- What is a suffix tree ?
- Gather the suffixes & build a tree.

Example (Suffix Tree for *atttat*)

```
atttat$  
tztat$  
ttat$  
tat$  
at$  
t$  
$
```

The Suffix Tree

26 min

- Suffix trees have optimal $O(m + occ)$ performance.
- What is a suffix tree ?
- Gather the suffixes & build a tree.

Example (Suffix Tree for *atttat*)

```
atttat$  
tztat$  
ttat$  
tat$  
at$  
t$  
$
```

The Suffix Tree

26 min

- Suffix trees have optimal $O(m + occ)$ performance.
- What is a suffix tree ?
- Gather the suffixes & build a tree.

Example (Suffix Tree for *atttat*)

```
atttat$
tztat$
ttat$
tat$
at$
t$
$
```

The Suffix Tree

26 min

- Suffix trees have optimal $O(m + occ)$ performance.
- What is a suffix tree ?
- Gather the suffixes & build a tree.

Example (Suffix Tree for *atttat*)

```
atttat$
tztat$
ttat$
tat$
at$
t$
$
```


The Suffix Tree

26 min

- Suffix trees have optimal $O(m + occ)$ performance.
- What is a suffix tree ?
- Gather the suffixes & build a tree.

Example (Suffix Tree for *atttat*)

atttat\$

tttat\$

ttat\$

tat\$

at\$

t\$

\$

The Suffix Tree

26 min

- Suffix trees have optimal $O(m + occ)$ performance.
- What is a suffix tree ?
- Gather the suffixes & build a tree.

Example (Suffix Tree for *atttat*)

```
atttat$  
tztat$  
ttat$  
tat$  
at$  
t$  
$
```

The Suffix Tree

26 min

- Suffix trees have optimal $O(m + occ)$ performance.
- What is a suffix tree ?
- Gather the suffixes & build a tree.

Example (Suffix Tree for *atttat*)

```
atttat$  
tztat$  
ttat$  
tat$  
at$  
t$  
$
```

The Suffix Tree

26 min

- Suffix trees have optimal $O(m + occ)$ performance.
- What is a suffix tree ?
- Gather the suffixes & build a tree.

Example (Suffix Tree for *atttat*)

```
atttat$  
tztat$  
ttat$  
tat$  
at$  
t$  
$
```

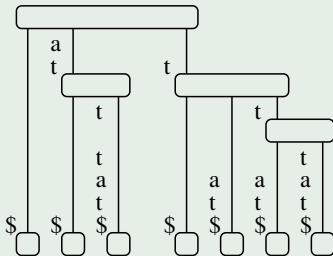
The Suffix Tree

26 min

- Suffix trees have optimal $O(m + occ)$ performance.
- What is a suffix tree ?
- Gather the suffixes & build a tree.

Example (Suffix Tree for *atttat*)

atttat\$
 ttatat\$
 ttat\$
 tat\$
 at\$
 t\$
 \$



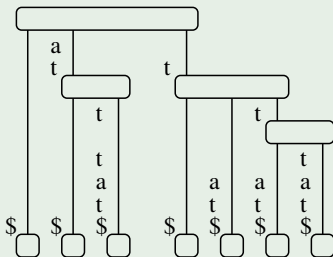
The Suffix Tree

24 min

- Searching means descending from the ROOT & reporting leaves below.
- Suffix trees have problems.

Example (Suffix Tree for *atttat*)

atttat\$
tttat\$
ttat\$
tat\$
at\$
t\$
\$



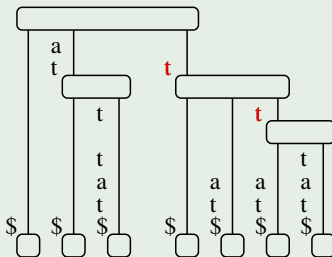
The Suffix Tree

24 min

- Searching means descending from the ROOT & reporting leaves below.
- Suffix trees have problems.

Example (Suffix Tree for *atttat*)

atttat\$
 ttatat\$
 ttatat\$
 ttatat\$
 ttatat\$
 ttatat\$
 ttatat\$
 ttatat\$



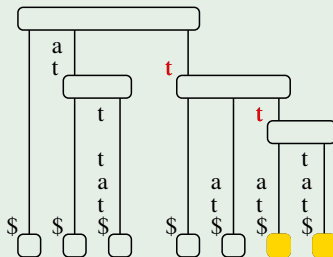
The Suffix Tree

24 min

- Searching means descending from the ROOT & reporting leaves below.
- Suffix trees have problems.

Example (Suffix Tree for *atttat*)

atttat\$
 ttat\$
 tat\$
 at\$
 t\$
 \$



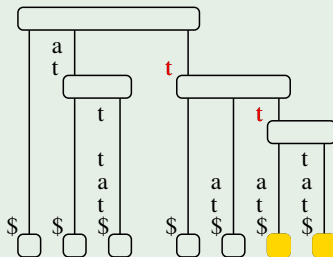
The Suffix Tree

24 min

- Searching means descending from the ROOT & reporting leaves below.
- Suffix trees have problems.

Example (Suffix Tree for *atttat*)

atttat\$
 ttat\$
 tat\$
 at\$
 t\$
 \$



Suffix Tree Representation

22 min

- Naive construction takes $O(n^2)$ space & time
- Efficient algorithms using pointers and amortized analysis take $O(n)$ space & time.

Example (Suffix Tree for *atttat*)

atttat\$	7	$n + 1$
tttat\$	+6	$+n$
ttat\$	+5	$+(n - 1)$
tat\$	+4	$+(n - 2)$
at\$	+3	...
t\$	+2	...
\$	+1	+1
	= 28	$= (n + 2)(n + 1)/2$
		$= O(n^2)$

Suffix Tree Representation

22 min

- Naive construction takes $O(n^2)$ space & time
- Efficient algorithms using pointers and amortized analysis take $O(n)$ space & time.

Example (Suffix Tree for *atttat*)

atttat\$	7	$n + 1$
tttat\$	+6	$+n$
ttat\$	+5	$+(n - 1)$
tat\$	+4	$+(n - 2)$
at\$	+3	...
t\$	+2	...
\$	+1	+1
	= 28	$= (n + 2)(n + 1)/2$
		$= O(n^2)$

Suffix Tree Representation

22 min

- Naive construction takes $O(n^2)$ space & time
- Efficient algorithms using pointers and amortized analysis take $O(n)$ space & time.

Example (Suffix Tree for *atttat*)

atttat\$	7	$n + 1$
tttat\$	+6	$+n$
ttat\$	+5	$+(n - 1)$
tat\$	+4	$+(n - 2)$
at\$	+3	...
t\$	+2	...
\$	+1	+1
	= 28	$= (n + 2)(n + 1)/2$
		$= O(n^2)$

Suffix Tree Representation

22 min

- Naive construction takes $O(n^2)$ space & time
- Efficient algorithms using pointers and amortized analysis take $O(n)$ space & time.

Example (Suffix Tree for *atttat*)

atttat\$	7	$n + 1$
tttat\$	+6	$+n$
ttat\$	+5	$+(n - 1)$
tat\$	+4	$+(n - 2)$
at\$	+3	...
t\$	+2	...
\$	+1	+1
	= 28	$= (n + 2)(n + 1)/2$
		$= O(n^2)$

Suffix Tree Representation

22 min

- Naive construction takes $O(n^2)$ space & time
- Efficient algorithms using pointers and amortized analysis take $O(n)$ space & time.

Example (Suffix Tree for *atttat*)

atttat\$	7	$n + 1$
tttat\$	+6	$+n$
ttat\$	+5	$+(n - 1)$
tat\$	+4	$+(n - 2)$
at\$	+3	...
t\$	+2	...
\$	+1	+1
	= 28	$= (n + 2)(n + 1)/2$
		$= O(n^2)$

Suffix Tree Representation

22 min

- Naive construction takes $O(n^2)$ space & time
- Efficient algorithms using pointers and amortized analysis take $O(n)$ space & time.

Example (Suffix Tree for *atttat*)

atttat\$	7	$n + 1$
tttat\$	+6	$+n$
ttat\$	+5	$+(n - 1)$
tat\$	+4	$+(n - 2)$
at\$	+3	...
t\$	+2	...
\$	+1	+1
	= 28	$= (n + 2)(n + 1)/2$
		$= O(n^2)$

Suffix Tree Representation

22 min

- Naive construction takes $O(n^2)$ space & time
- Efficient algorithms using pointers and amortized analysis take $O(n)$ space & time.

Example (Suffix Tree for *atttat*)

atttat\$	7	$n + 1$
tttat\$	+6	$+n$
ttat\$	+5	$+(n - 1)$
tat\$	+4	$+(n - 2)$
at\$	+3	...
t\$	+2	...
\$	+1	+1
	= 28	$= (n + 2)(n + 1)/2$
		$= O(n^2)$

Suffix Tree Representation

22 min

- Naive construction takes $O(n^2)$ space & time
- Efficient algorithms using pointers and amortized analysis take $O(n)$ space & time.

Example (Suffix Tree for *atttat*)

atttat\$	7	$n + 1$
tttat\$	+6	$+n$
ttat\$	+5	$+(n - 1)$
tat\$	+4	$+(n - 2)$
at\$	+3	...
t\$	+2	...
\$	+1	+1
	= 28	$= (n + 2)(n + 1)/2$
		$= O(n^2)$

Suffix Tree Representation

22 min

- Naive construction takes $O(n^2)$ space & time
- Efficient algorithms using pointers and amortized analysis take $O(n)$ space & time.

Example (Suffix Tree for *atttat*)

atttat\$	7	$n + 1$
tttat\$	+6	$+n$
ttat\$	+5	$+(n - 1)$
tat\$	+4	$+(n - 2)$
at\$	+3	...
t\$	+2	...
\$	+1	+1
	= 28	$= (n + 2)(n + 1)/2$
		$= O(n^2)$

Suffix Tree Representation

22 min

- Naive construction takes $O(n^2)$ space & time
- Efficient algorithms using pointers and amortized analysis take $O(n)$ space & time.

Example (Suffix Tree for *atttat*)

atttat\$	7	$n + 1$
tttat\$	+6	$+n$
ttat\$	+5	$+(n - 1)$
tat\$	+4	$+(n - 2)$
at\$	+3	...
t\$	+2	...
\$	+1	+1
	= 28	$= (n + 2)(n + 1)/2$
		$= O(n^2)$

Suffix Tree Representation

22 min

- Naive construction takes $O(n^2)$ space & time
- Efficient algorithms using pointers and amortized analysis take $O(n)$ space & time.

Example (Suffix Tree for *atttat*)

atttat\$	7	$n + 1$
tttat\$	+6	$+n$
ttat\$	+5	$+(n - 1)$
tat\$	+4	$+(n - 2)$
at\$	+3	...
t\$	+2	...
\$	+1	+1
	= 28	$= (n + 2)(n + 1)/2$
		$= O(n^2)$

Suffix Tree Representation

22 min

- Naive construction takes $O(n^2)$ space & time
- Efficient algorithms using pointers and amortized analysis take $O(n)$ space & time.

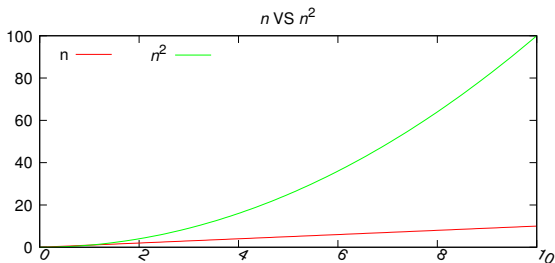
Example (Suffix Tree for *atztat*)

atztat\$	7	$n + 1$
tztat\$	+6	$+n$
tat\$	+5	$+(n - 1)$
at\$	+4	$+(n - 2)$
t\$	+3	...
\$	+2	...
	+1	+1
	= 28	$= (n + 2)(n + 1)/2$
		$= O(n^2)$

What is the difference ?

20 min

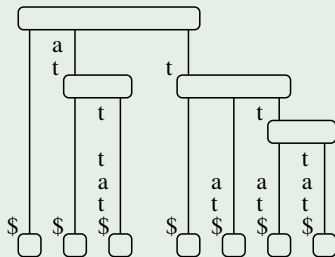
There is no way to index 3Gb DNA with an $O(n^2)$ algorithm



Suffix Link

19 min

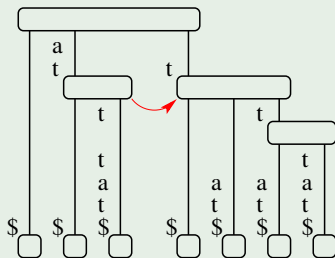
- Not going to explain Ukkonen's algorithm
- Key concept SLINK, always exists

Example (Suffix Tree for *attat*)

Suffix Link

19 min

- Not going to explain Ukkonen's algorithm
- Key concept SLINK, always exists

Example (Suffix Tree for *attat*)

Representation Problems

17 min

Problem (Suffix Trees need too much space)

Pointer based representations require $O(n \log n)$ bits.

- This is much larger than the indexed string, $n \log \sigma$ bits.
- State of the art implementations require $[8, 10] \times 8n$ bits.
- This is 48 Gb for 3 Gb of DNA, i.e. for the human genome.
- Modern representations use data compression techniques.
- Compressing the tree requires finding regularities.

Representation Problems

17 min

Problem (Suffix Trees need too much space)

Pointer based representations require $O(n \log n)$ bits.

- This is much larger than the indexed string, $n \log \sigma$ bits.
- State of the art implementations require $[8, 10] \times 8n$ bits.
- This is 48 Gb for 3 Gb of DNA, i.e. for the human genome.
- Modern representations use data compression techniques.
- Compressing the tree requires finding regularities.

Representation Problems

17 min

Problem (Suffix Trees need too much space)

Pointer based representations require $O(n \log n)$ bits.

- This is much larger than the indexed string, $n \log \sigma$ bits.
- State of the art implementations require $[8, 10] \times 8n$ bits.
- This is 48 Gb for 3 Gb of DNA, i.e. for the human genome.
- Modern representations use data compression techniques.
- Compressing the tree requires finding regularities.

Representation Problems

17 min

Problem (Suffix Trees need too much space)

Pointer based representations require $O(n \log n)$ bits.

- This is much larger than the indexed string, $n \log \sigma$ bits.
- State of the art implementations require $[8, 10] \times 8n$ bits.
- This is 48 Gb for 3 Gb of DNA, i.e. for the human genome.
- Modern representations use data compression techniques.
- Compressing the tree requires finding regularities.

Representation Problems

17 min

Problem (Suffix Trees need too much space)

Pointer based representations require $O(n \log n)$ bits.

- This is much larger than the indexed string, $n \log \sigma$ bits.
- State of the art implementations require $[8, 10] \times 8n$ bits.
- This is 48 Gb for 3 Gb of DNA, i.e. for the human genome.
- Modern representations use data compression techniques.
- Compressing the tree requires finding regularities.

Representation Problems

17 min

Problem (Suffix Trees need too much space)

Pointer based representations require $O(n \log n)$ bits.

- This is much larger than the indexed string, $n \log \sigma$ bits.
- State of the art implementations require $[8, 10] \times 8n$ bits.
- This is 48 Gb for 3 Gb of DNA, i.e. for the human genome.
- Modern representations use data compression techniques.
- Compressing the tree requires finding regularities.

Lowest Common Ancestors

15 min

- The LCA operation.
- can be computed in $O(1)$ time, $O(n)$ space.
- No dark magic, just good algorithms ;-)

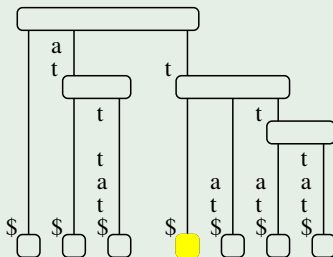
Example (Suffix Tree for *atttat*)

Lowest Common Ancestors

15 min

- The LCA operation.
- can be computed in $O(1)$ time, $O(n)$ space.
- No dark magic, just good algorithms ;-)

Example (Suffix Tree for *atztat*)

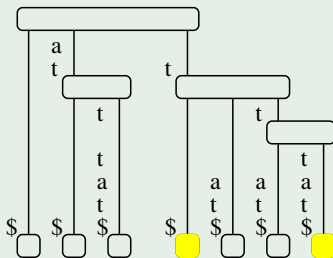


Lowest Common Ancestors

15 min

- The LCA operation.
- can be computed in $O(1)$ time, $O(n)$ space.
- No dark magic, just good algorithms ;-)

Example (Suffix Tree for *atttat*)

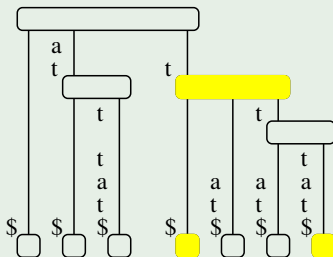


Lowest Common Ancestors

15 min

- The LCA operation.
- can be computed in $O(1)$ time, $O(n)$ space.
- No dark magic, just good algorithms ;-)

Example (Suffix Tree for *atttat*)

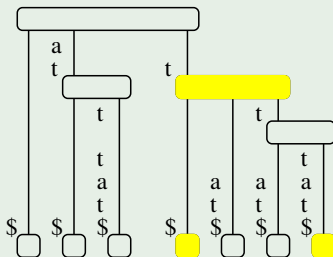


Lowest Common Ancestors

15 min

- The LCA operation.
- can be computed in $O(1)$ time, $O(n)$ space.
- No dark magic, just good algorithms ;-)

Example (Suffix Tree for *atztat*)

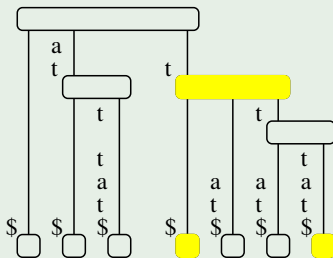


Lowest Common Ancestors

15 min

- The LCA operation.
- can be computed in $O(1)$ time, $O(n)$ space.
- No dark magic, just good algorithms ;-)

Example (Suffix Tree for *atttat*)



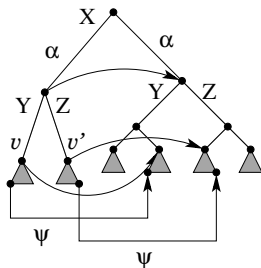
LCA and SLINK

12 min

Lemma

When $LCA(v, v') \neq \text{ROOT}$ we have that:

$$\text{SLINK}(LCA(v, v')) = LCA(\text{SLINK}(v), \text{SLINK}(v'))$$



Using this lemma we do not need to store all the nodes, only some sampled ones.

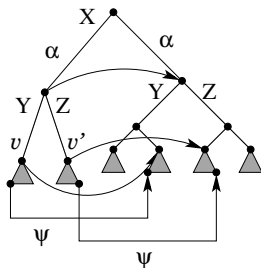
LCA and SLINK

12 min

Lemma

When $LCA(v, v') \neq \text{ROOT}$ we have that:

$$\text{SLINK}(LCA(v, v')) = LCA(\text{SLINK}(v), \text{SLINK}(v'))$$



Using this lemma we do not need to store all the nodes, only some sampled ones.

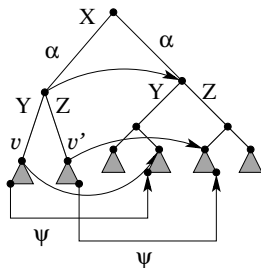
LCA and SLINK

12 min

Lemma

When $LCA(v, v') \neq \text{ROOT}$ we have that:

$$\text{SLINK}(LCA(v, v')) = LCA(\text{SLINK}(v), \text{SLINK}(v'))$$



Using this lemma we do not need to store all the nodes, only some sampled ones.

Fundamental lemma

10 min

Lemma

If $\text{SLINK}^r(\text{LCA}(v, v')) = \text{ROOT}$, and let $d = \min(\delta, r + 1)$.

Then $\text{SDEP}(\text{LCA}(v, v')) =$

$$\max_{0 \leq i < d} \{i + \text{SDEP}(\text{LCSA}(\text{SLINK}^i(v), \text{SLINK}^i(v')))\}$$

Proof.

$$\text{SDEP}(\text{LCA}(v, v'))$$

$$= i + \text{SDEP}(\text{SLINK}^i(\text{LCA}(v, v')))$$

$$= i + \text{SDEP}(\text{LCA}(\text{SLINK}^i(v), \text{SLINK}^i(v')))$$

$$\geq i + \text{SDEP}(\text{LCSA}(\text{SLINK}^i(v), \text{SLINK}^i(v')))$$

The last inequality is an equality for some $i \leq d$. □

Fundamental lemma

10 min

Lemma

If $\text{SLINK}^r(\text{LCA}(v, v')) = \text{ROOT}$, and let $d = \min(\delta, r + 1)$.

Then $\text{SDEP}(\text{LCA}(v, v')) =$

$$\max_{0 \leq i < d} \{i + \text{SDEP}(\text{LCSA}(\text{SLINK}^i(v), \text{SLINK}^i(v')))\}$$

Proof.

$\text{SDEP}(\text{LCA}(v, v'))$

$$= i + \text{SDEP}(\text{SLINK}^i(\text{LCA}(v, v')))$$

$$= i + \text{SDEP}(\text{LCA}(\text{SLINK}^i(v), \text{SLINK}^i(v')))$$

$$\geq i + \text{SDEP}(\text{LCSA}(\text{SLINK}^i(v), \text{SLINK}^i(v')))$$

The last inequality is an equality for some $i \leq d$. □

Fundamental lemma

10 min

Lemma

If $\text{SLINK}^r(\text{LCA}(v, v')) = \text{ROOT}$, and let $d = \min(\delta, r + 1)$.

Then $\text{SDEP}(\text{LCA}(v, v'))?$

$$\max_{0 \leq i < d} \{i + \text{SDEP}(\text{LCSA}(\text{SLINK}^i(v), \text{SLINK}^i(v')))\}$$

Proof.

$$\text{SDEP}(\text{LCA}(v, v'))$$

$$= i + \text{SDEP}(\text{SLINK}^i(\text{LCA}(v, v')))$$

$$= i + \text{SDEP}(\text{LCA}(\text{SLINK}^i(v), \text{SLINK}^i(v')))$$

$$\geq i + \text{SDEP}(\text{LCSA}(\text{SLINK}^i(v), \text{SLINK}^i(v')))$$

The last inequality is an equality for some $i \leq d$. □

Fundamental lemma

10 min

Lemma

If $\text{SLINK}^r(\text{LCA}(v, v')) = \text{ROOT}$, and let $d = \min(\delta, r + 1)$.

Then $\text{SDEP}(\text{LCA}(v, v'))?$

$$\max_{0 \leq i < d} \{i + \text{SDEP}(\text{LCSA}(\text{SLINK}^i(v), \text{SLINK}^i(v')))\}$$

Proof.

$$\text{SDEP}(\text{LCA}(v, v'))$$

$$= i + \text{SDEP}(\text{SLINK}^i(\text{LCA}(v, v')))$$

$$= i + \text{SDEP}(\text{LCA}(\text{SLINK}^i(v), \text{SLINK}^i(v')))$$

$$\geq i + \text{SDEP}(\text{LCSA}(\text{SLINK}^i(v), \text{SLINK}^i(v')))$$

The last inequality is an equality for some $i \leq d$. □

Fundamental lemma

10 min

Lemma

If $\text{SLINK}^r(\text{LCA}(v, v')) = \text{ROOT}$, and let $d = \min(\delta, r + 1)$.

Then $\text{SDEP}(\text{LCA}(v, v')) \geq$

$$\max_{0 \leq i < d} \{i + \text{SDEP}(\text{LCSA}(\text{SLINK}^i(v), \text{SLINK}^i(v')))\}$$

Proof.

$$\text{SDEP}(\text{LCA}(v, v'))$$

$$= i + \text{SDEP}(\text{SLINK}^i(\text{LCA}(v, v')))$$

$$= i + \text{SDEP}(\text{LCA}(\text{SLINK}^i(v), \text{SLINK}^i(v')))$$

$$\geq i + \text{SDEP}(\text{LCSA}(\text{SLINK}^i(v), \text{SLINK}^i(v')))$$

The last inequality is an equality for some $i \leq d$. □

Fundamental lemma

10 min

Lemma

If $\text{SLINK}^r(\text{LCA}(v, v')) = \text{ROOT}$, and let $d = \min(\delta, r + 1)$.

Then $\text{SDEP}(\text{LCA}(v, v')) =$

$$\max_{0 \leq i < d} \{i + \text{SDEP}(\text{LCSA}(\text{SLINK}^i(v), \text{SLINK}^i(v')))\}$$

Proof.

$$\text{SDEP}(\text{LCA}(v, v'))$$

$$= i + \text{SDEP}(\text{SLINK}^i(\text{LCA}(v, v')))$$

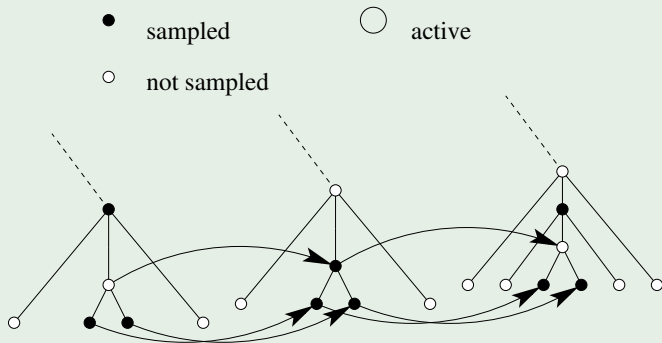
$$= i + \text{SDEP}(\text{LCA}(\text{SLINK}^i(v), \text{SLINK}^i(v')))$$

$$\geq i + \text{SDEP}(\text{LCSA}(\text{SLINK}^i(v), \text{SLINK}^i(v')))$$

The last inequality is an equality for some $i \leq d$. □

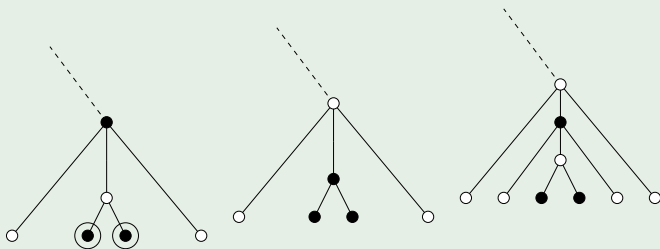
Fundamental lemma

7 min

Example ($\delta = 3$)

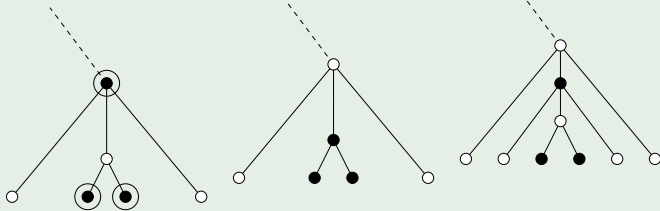
Fundamental lemma

7 min

Example ($\delta = 3$)

Fundamental lemma

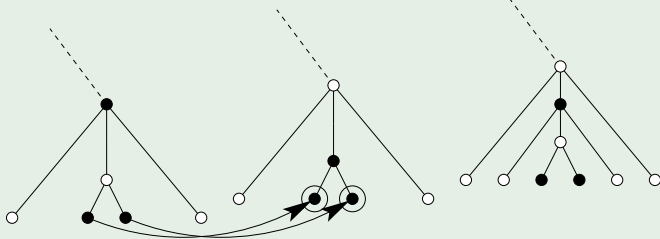
7 min

Example ($\delta = 3$)

SDEP : 5

Fundamental lemma

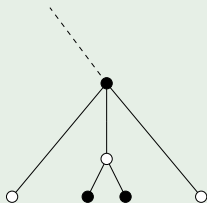
7 min

Example ($\delta = 3$)

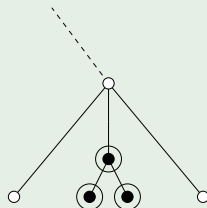
SDEP : 5

Fundamental lemma

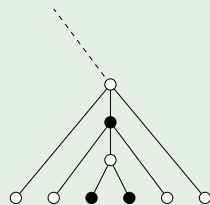
7 min

Example ($\delta = 3$)

SDEP : 5

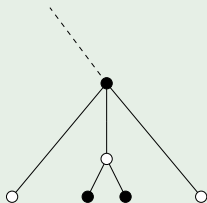


10

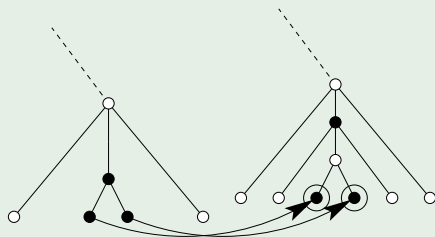


Fundamental lemma

7 min

Example ($\delta = 3$)

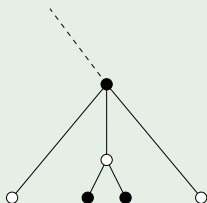
SDEP : 5



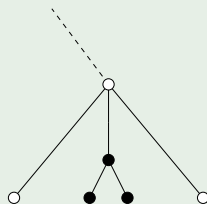
10

Fundamental lemma

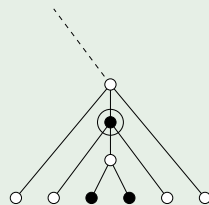
7 min

Example ($\delta = 3$)

SDEP : 5



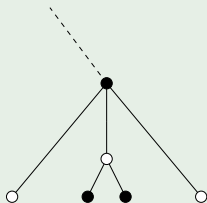
10



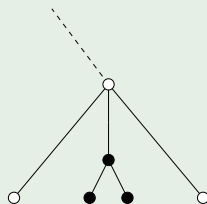
7

Fundamental lemma

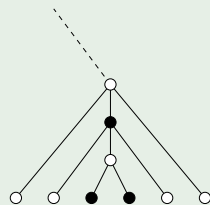
7 min

Example ($\delta = 3$)

5+0



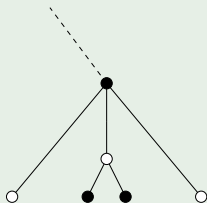
10+1



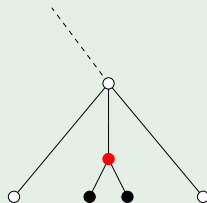
7+2

Fundamental lemma

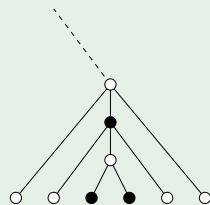
7 min

Example ($\delta = 3$)

5+0



10+1

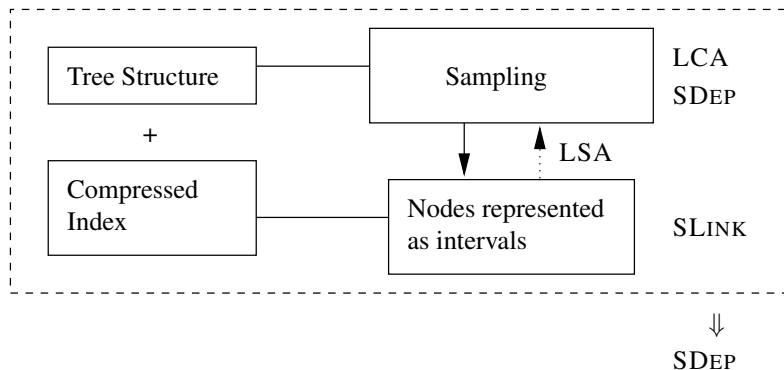


7+2

Entangled Operations

6 min

Why is the lemma important ?



Experimental Results

3 min

Space in MBs	File	FCST	FFCST	LSA	LSAF	CST
Pitches	53	44	134	43	50	214
Proteins	63	50	121	48	56	204
DNA	100	57	142	54	69	287
XML	100	55	198	52	67	316

CHILD Operation	Pitches	Proteins	DNA	XML
FCST	1.3E-2	3.4E-3	1.6E-3	8.7E-3
FFCST	9.4E-3	5.7E-3	7.2E-1	1.9E-2
LSA	7.6E-3	2.7E-3	2.E-3	9.7E-3
LSAF	1.3E-2	3.5E-3	1.6E-3	8.9E-3
CST	5.4E-4	4.2E-4	1.2E-4	6.4E-4

Experimental Results

3 min

Space in MBs	File	FCST	FFCST	LSA	LSAF	CST
Pitches	53	44	134	43	50	214
Proteins	63	50	121	48	56	204
DNA	100	57	142	54	69	287
XML	100	55	198	52	67	316

CHILD Operation	Pitches	Proteins	DNA	XML
FCST	1.3E-2	3.4E-3	1.6E-3	8.7E-3
FFCST	9.4E-3	5.7E-3	7.2E-1	1.9E-2
LSA	7.6E-3	2.7E-3	2.E-3	9.7E-3
LSAF	1.3E-2	3.5E-3	1.6E-3	8.9E-3
CST	5.4E-4	4.2E-4	1.2E-4	6.4E-4

Summary

2 min

We presented a representation of suffix trees that:

- occupies $n \log \sigma + o(u \log \sigma)$ bits.
- in fact it is even better $nH_k + o(u \log \sigma)$ bits.
- supports usual operations in a reasonable time.
- current prototypes show that this performance holds in practice.

Summary

2 min

We presented a representation of suffix trees that:

- occupies $n \log \sigma + o(u \log \sigma)$ bits.
- in fact it is even better $nH_k + o(u \log \sigma)$ bits.
- supports usual operations in a reasonable time.
- current prototypes show that this performance holds in practice.

Summary

2 min

We presented a representation of suffix trees that:

- occupies $n \log \sigma + o(u \log \sigma)$ bits.
- in fact it is even better $nH_k + o(u \log \sigma)$ bits.
- supports usual operations in a reasonable time.
- current prototypes show that this performance holds in practice.

Summary

2 min

We presented a representation of suffix trees that:

- occupies $n \log \sigma + o(u \log \sigma)$ bits.
- in fact it is even better $nH_k + o(u \log \sigma)$ bits.
- supports usual operations in a reasonable time.
- current prototypes show that this performance holds in practice.