Lempel Ziv Computation In Small Space (LZ-CISS)

Johannes Fischer Tomohiro I *Dominik Köppl*

Department of Computer Science, TU Dortmund, Germany

30.6.2015
CPM

LZ Parsing

Text: aaabaabaaabaa$

LZ77

	1	2	3	4	5	6
Factor	a	aa	b	aabaa	abaa	$
Coding	a	1,2	b	2,5	3,4	$

LZ78

	1	2	3	4	5	6	7
Factor	a	aa	b	aab	aaa	ba	a$
Coding	a	1,a	b	2,b	2,a	3,a	1,$

Quest for **small working space**

- LZ77

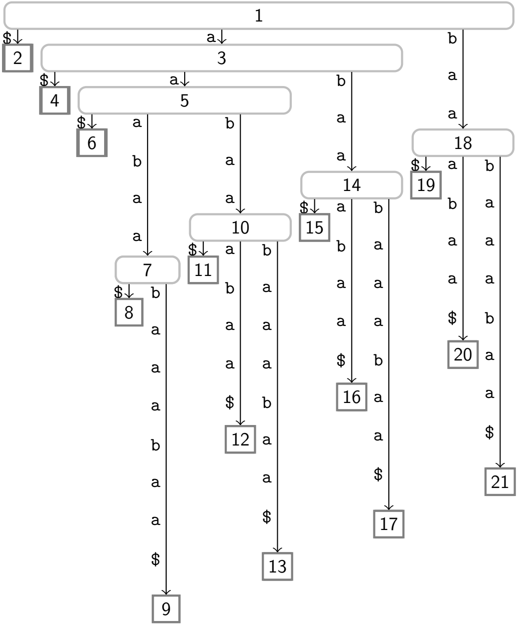
Time	Bits of working space	Authors
$\mathcal{O}(n)$	$3n \lg n$	Goto and Bannai'13
$\mathcal{O}(n)$	$2n \lg n$	Kärkkäinen et al.'13
$\mathcal{O}(n)$	$n \lg n + \mathcal{O}(\sigma \lg n)$	Goto and Bannai'14
$\mathcal{O}(n)$	$(1+\epsilon) n \lg n + \mathcal{O}(n)$	this paper

- LZ78

Time	Bits of working space	Authors
$\mathcal{O}(n \lg \sigma)$	$\mathcal{O}(z \lg z)$	folklore
$\mathcal{O}\left(n + z \frac{\lg^2 \lg \sigma}{\lg \lg \lg \sigma}\right)$	$\mathcal{O}(z \lg z)$	Fischer,Gawrychowski'15
$\mathcal{O}\left(\frac{n \lg^2 \lg n}{(\lg_\sigma n \lg \lg \lg n)}\right)$	$\mathcal{O}\left(n \lg \sigma + n \frac{\lg \lg_\sigma n}{\lg_\sigma n}\right)$	Jansson et al.'15
$\mathcal{O}(n)$	$\mathcal{O}(n \lg n)$	Nakashima et al.'15
$\mathcal{O}(n)$	$(1+\epsilon) n \lg n + \mathcal{O}(n)$	this paper

n: text size σ: alphabet size z: #factors $0 < \epsilon \leq 1$ constant

Suffix Tree



ST of aaabaabaaabaa

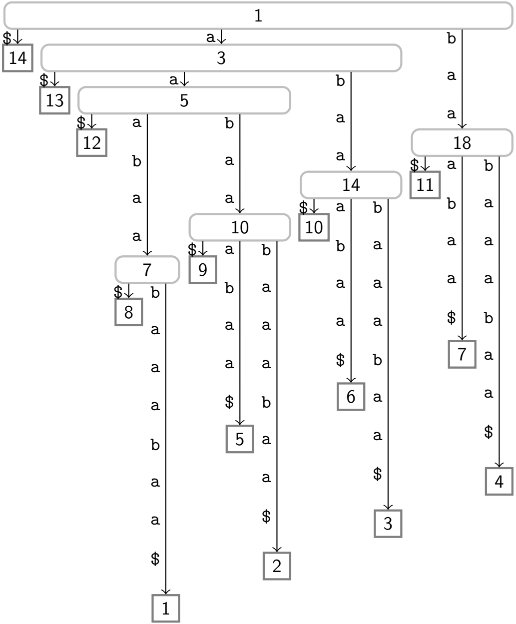
Labels

- internal nodes: DFS number
- leaves: text pos. of suffix

Superimposition

- suffix trie *superimposed* by suffix tree
- LZ78 trie subtree of suffix trie
- edge counts how far it got explored.

Suffix Tree



ST of aaabaabaaabaa

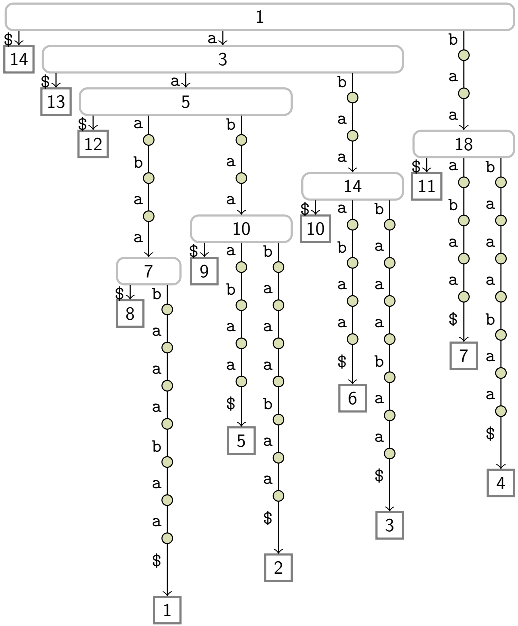
Labels

- internal nodes: DFS number
- leaves: text pos. of suffix

Superimposition

- suffix trie *superimposed* by suffix tree
- LZ78 trie subtree of suffix trie
- edge counts how far it got explored.

Suffix Tree



ST of aaabaabaaabaa

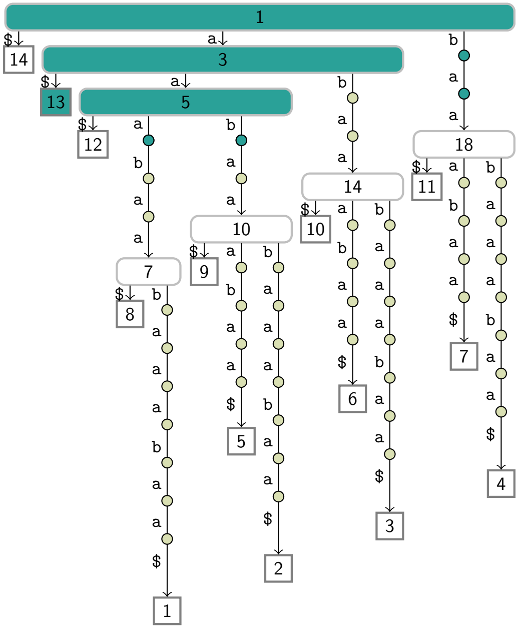
Labels

- internal nodes: DFS number
- leaves: text pos. of suffix

Superimposition

- suffix trie *superimposed* by suffix tree
- LZ78 trie subtree of suffix trie
- edge counts how far it got explored.

Suffix Tree



ST of aaabaabaaabaa

Labels

- internal nodes: DFS number
- leaves: text pos. of suffix

Superimposition

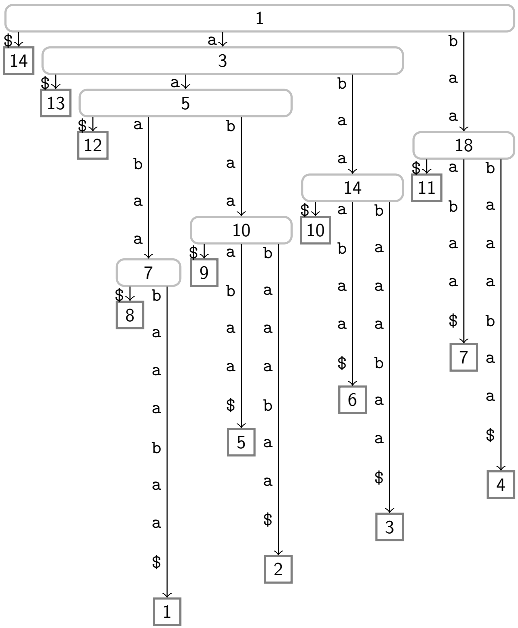
- suffix trie *superimposed* by suffix tree
- LZ78 trie subtree of suffix trie
- edge counts how far it got explored.

LZ78

LZ78

Add new factor $\equiv$ append new node to trie:

- Classic Trie: Walk down from root.
- Now: Use Level Ancestor Query on ST leaf
- Traverse in $\mathcal{O}(1)$ time.



LZ78 parsing of
aaabaabaaabaa:

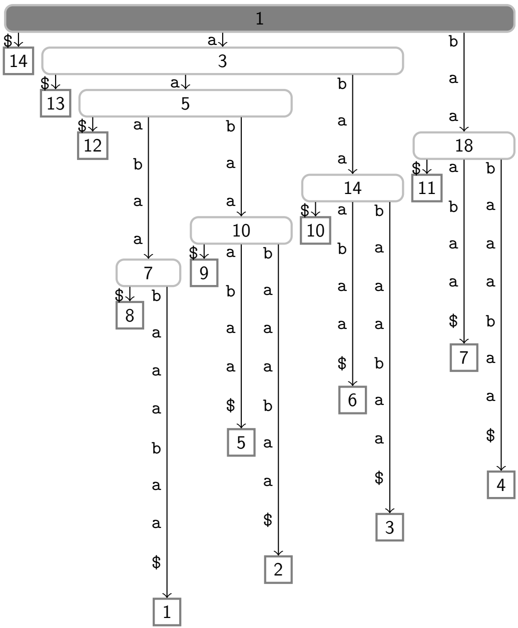
a |aa |b |aab |aaa |ba |a$

Witnesses:

3, 5, 18, 10, 7, 18, 4

Definition

Witness: highest ST node *below of or equal to* LZ78 trie node.



LZ78 parsing of aaabaabaaabaa:

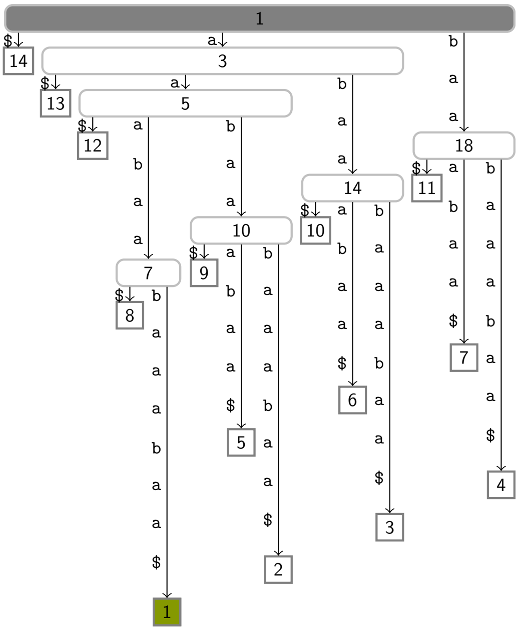
a |aa |b |aab |aaa |ba |a$

Witnesses:

3, 5, 18, 10, 7, 18, 4

Definition

Witness: highest ST node *below of or equal to* LZ78 trie node.



LZ78 parsing of
aaabaabaaabaa:

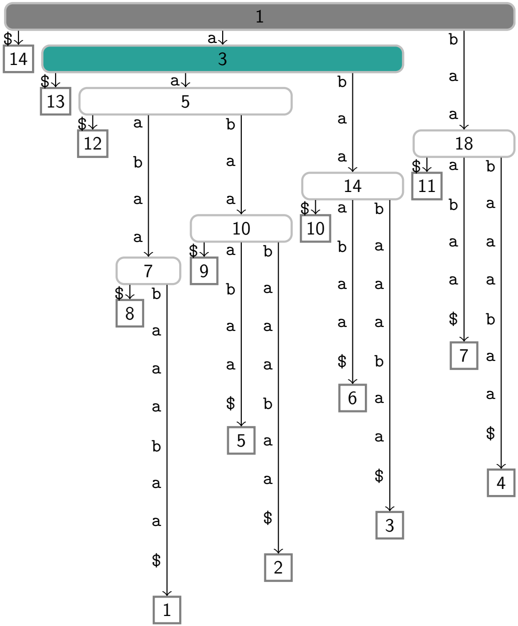
a |aa |b |aab |aaa |ba |a$

Witnesses:

3, 5, 18, 10, 7, 18, 4

Definition

Witness: highest ST node *below of or equal to* LZ78 trie node.



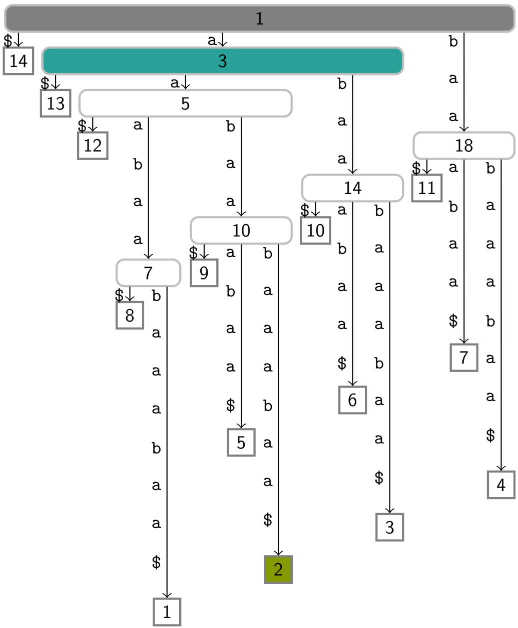
LZ78 parsing of
aaabaabaaabaa:

a |aa |b |aab |aaa |ba |a$

Witnesses:
3, 5, 18, 10, 7, 18, 4

Definition

Witness: highest ST node *below of or equal to* LZ78 trie node.



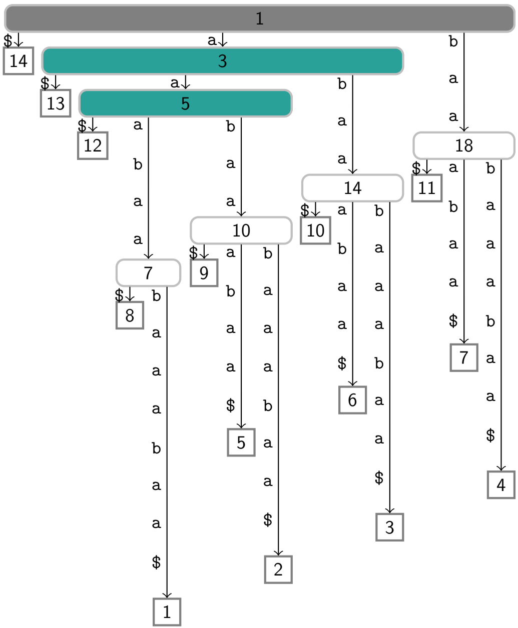
LZ78 parsing of
aaabaabaaabaa:

a |aa |b |aab |aaa |ba |a$

Witnesses:
3, 5, 18, 10, 7, 18, 4

Definition

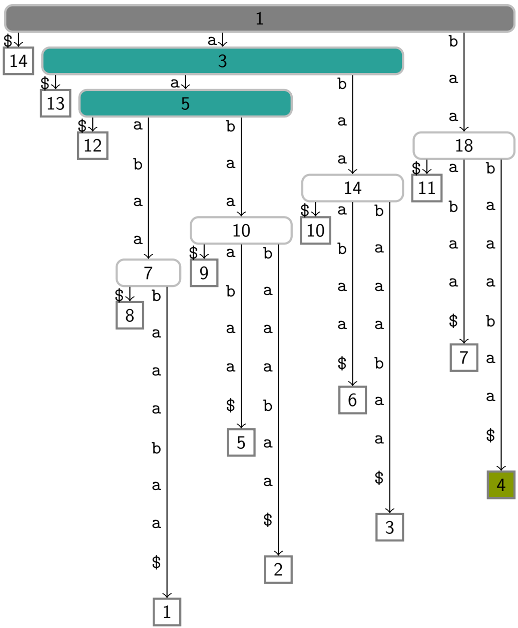
Witness: highest ST node *below of or equal to* LZ78 trie node.



LZ78 parsing of
aaabaabaaabaa:
a |aa |b |aab |aaa |ba |a$

Witnesses:
3, 5, 18, 10, 7, 18, 4

Definition
Witness: highest ST node *below of or equal to* LZ78 trie node.



LZ78 parsing of
aaabaabaaabaa:

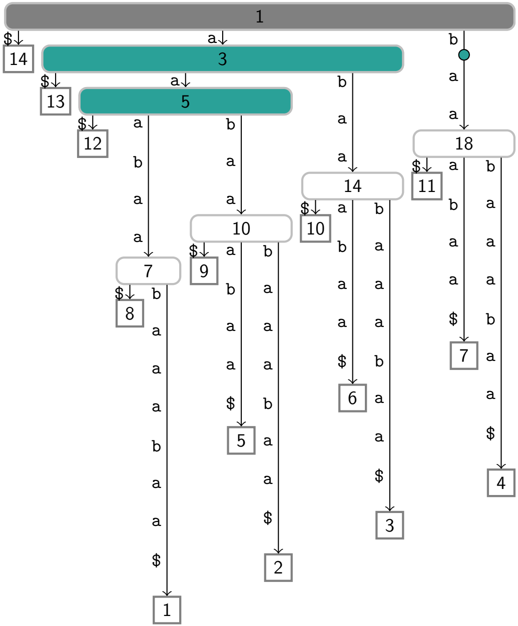
a |aa |b |aab |aaa |ba |a$

Witnesses:

3, 5, 18, 10, 7, 18, 4

Definition

Witness: highest ST node *below of or equal to* LZ78 trie node.



LZ78 parsing of aaabaabaaabaa:

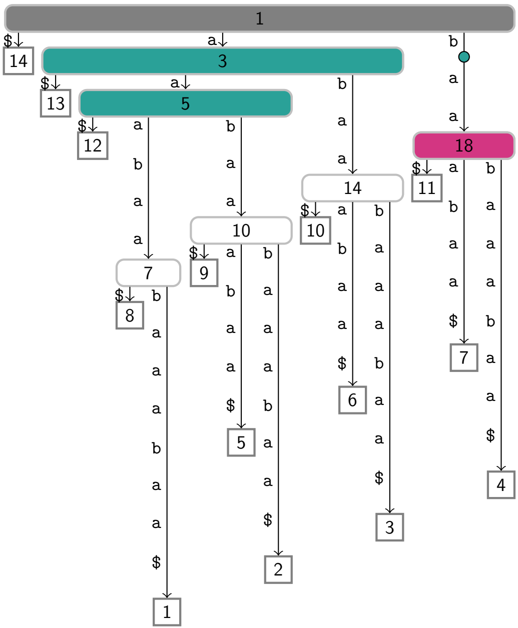
a |aa |b |aab |aaa |ba |a$

Witnesses:

3, 5, 18, 10, 7, 18, 4

Definition

Witness: highest ST node *below of or equal to* LZ78 trie node.



LZ78 parsing of
aaabaabaaabaa:

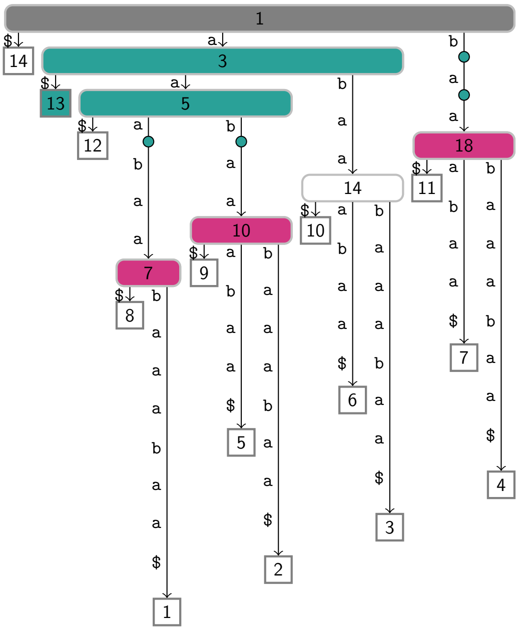
a |aa |b |aab |aaa |ba |a$

Witnesses:

3, 5, 18, 10, 7, 18, 4

Definition

Witness: highest ST node *below of or equal to* LZ78 trie node.

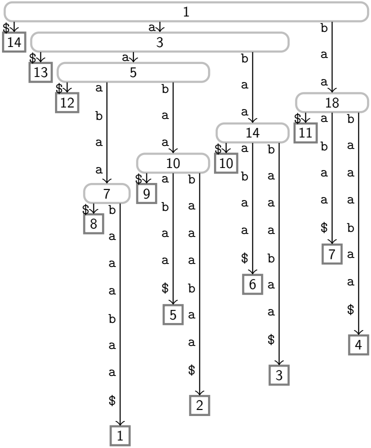


LZ78 parsing of
aaabaabaaabaa:
a |aa |b |aab |aaa |ba |a$

Witnesses:
3, 5, 18, 10, 7, 18, 4

Definition
Witness: highest ST node *below of or equal to* LZ78 trie node.

Witnesses → References

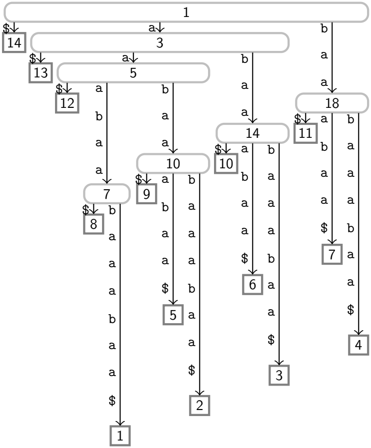


1	2	3	4	5	6	7
3	5	18	10	7	18	4

3	4	5	7	10	18

Working Space

Witnesses → References



1	2	3	4	5	6	7
3	5	18	10	7	18	4

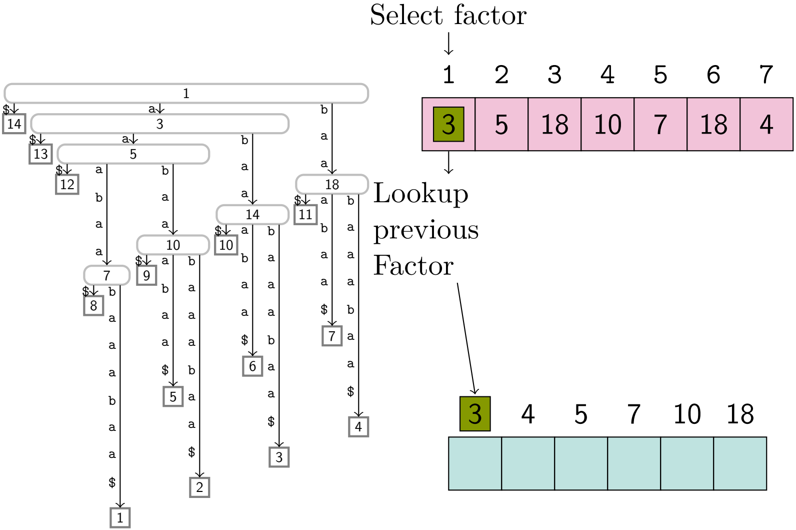
Witnesses

Witnesses

3	4	5	7	10	18

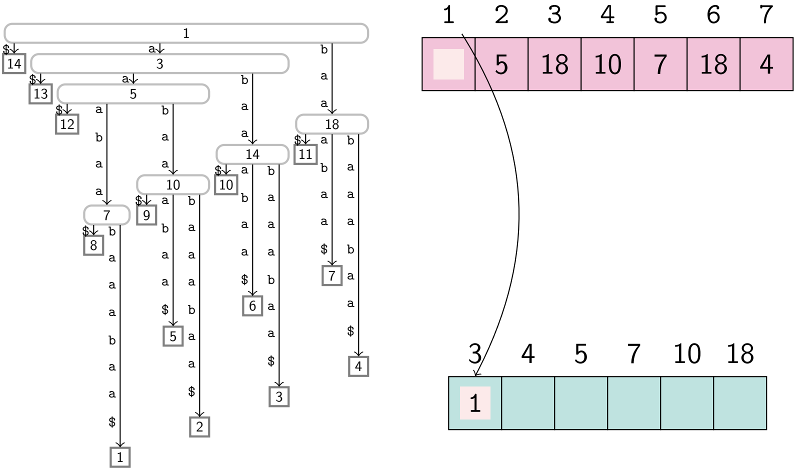
Working Space

Witnesses → References

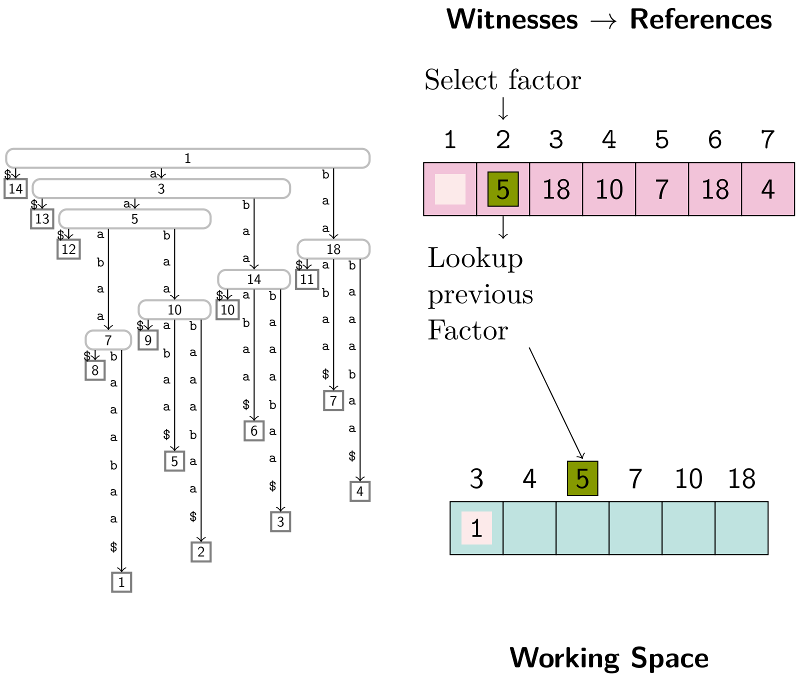


Working Space

Witnesses → References

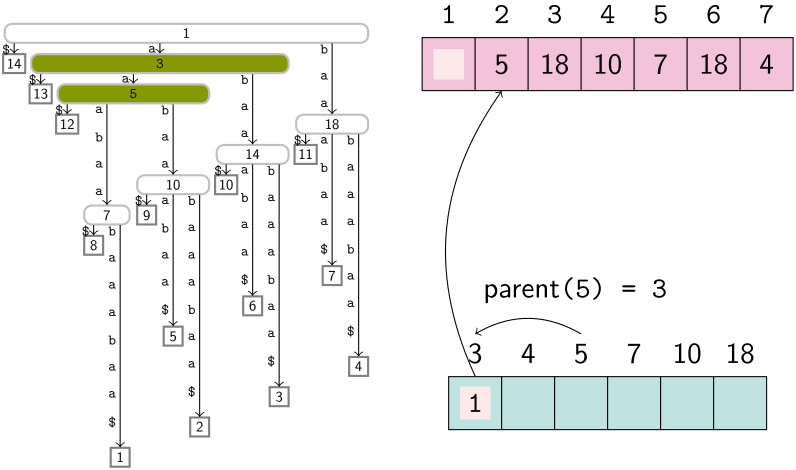


Working Space



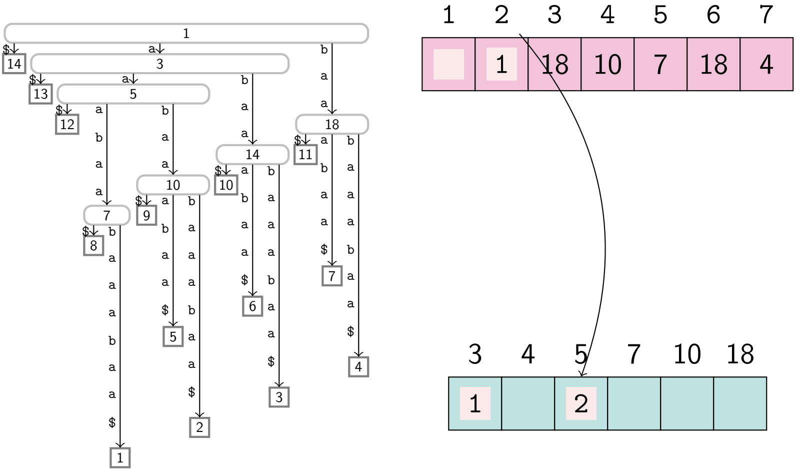
Witnesses → References
Select factor
1 2 3 4 5 6 7
5 18 10 7 18 4
Lookup previous Factor
3 4 5 7 10 18
1
Working Space

Witnesses → References

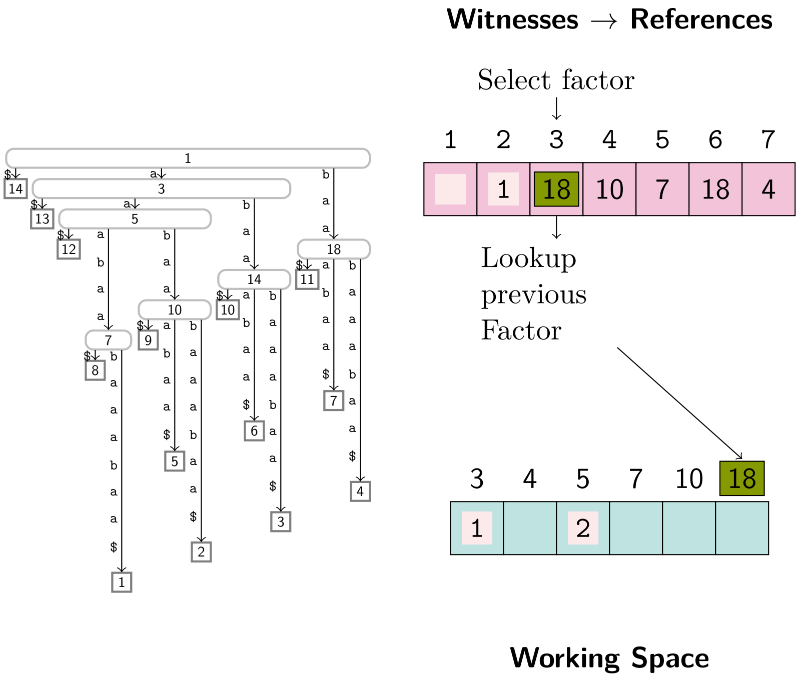


Working Space

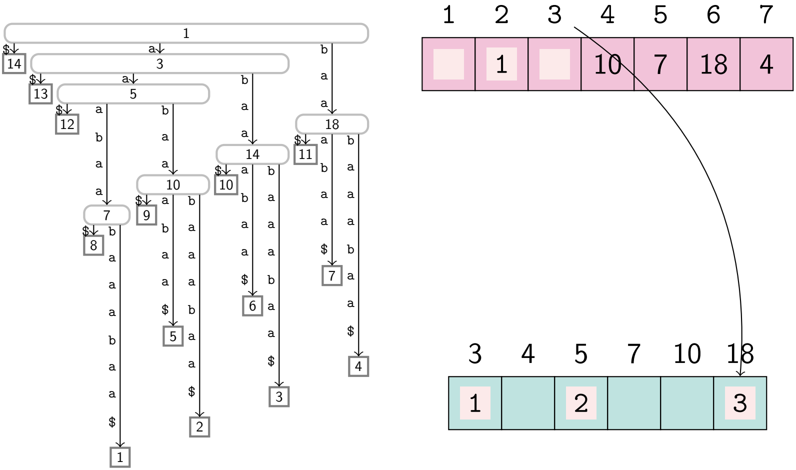
Witnesses → References



Working Space

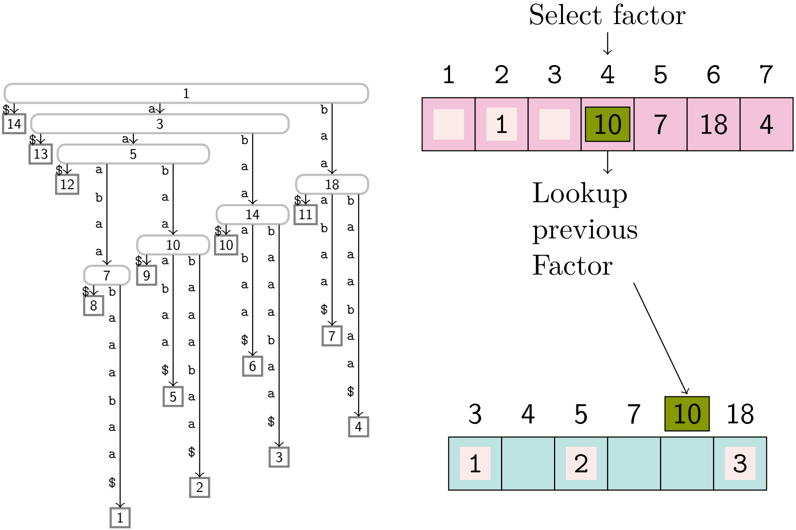

Witnesses → References
Select factor
1 2 3 4 5 6 7
1 18 10 7 18 4
Lookup previous Factor
3 4 5 7 10 18
1 2
Working Space

Witnesses → References



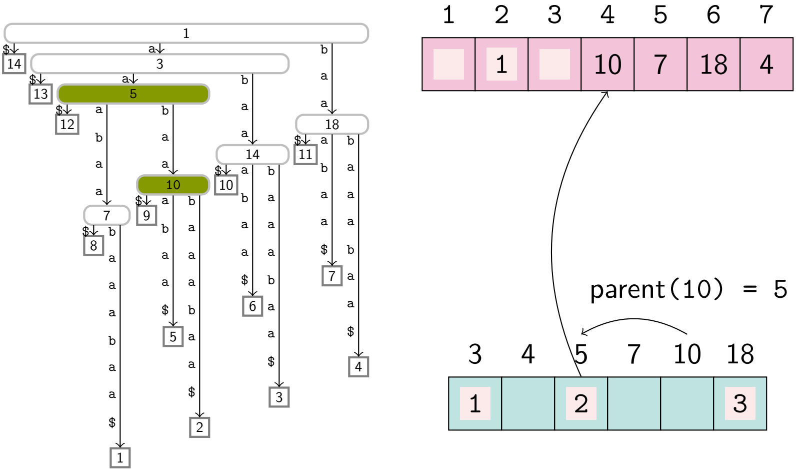
Working Space

Witnesses → References



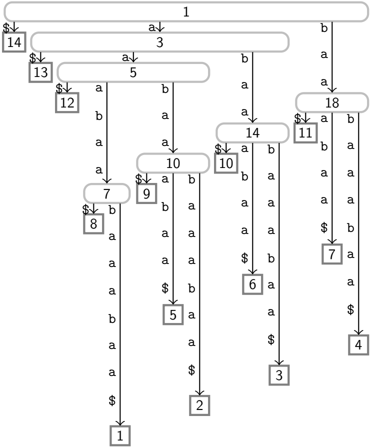
Working Space

Witnesses → References



Working Space

Witnesses → References

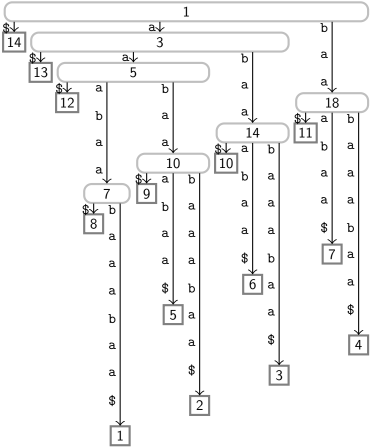


1	2	3	4	5	6	7
	1		2	7	18	4

3	4	5	7	10	18
1		2		4	3

Working Space

Witnesses → References



1	2	3	4	5	6	7
	1		2	2	3	1

Referred Factors

3	4	5	7	10	18
1	7	2	5	4	6

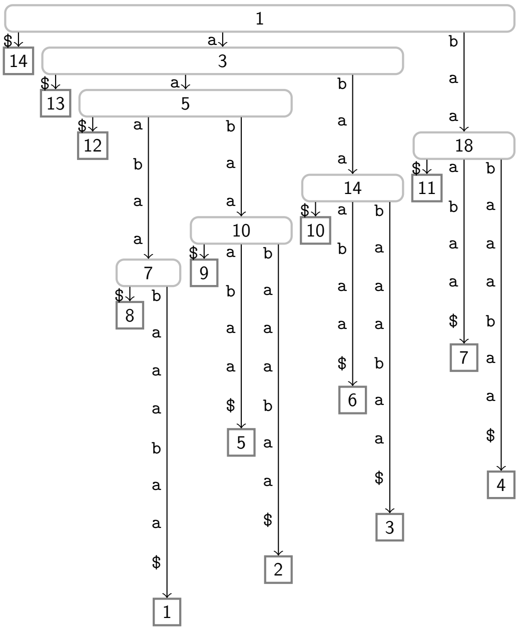
Working Space

LZ77

LZ77

Two passes over the suffix tree:

1. Pass:
   - Traverse from *every* leaf to the root
   - Mark visited nodes
   - Already marked nodes $\equiv$ some reference
   - These nodes witness references
2. Pass:
   - Same procedure
   - We know witnesses already!
   - Which leaf discovers a witness first?



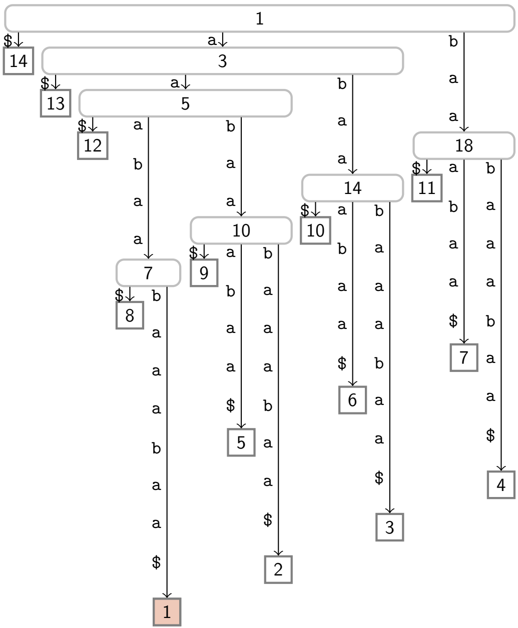
LZ77 parsing of
aaabaabaaabaa :
a |aa |b |aabaa |abaa |$.

Starting positions of
factors (≡ leaf labels):
1, 2, 4, 5, 10, 14

Witnesses:
5, 10, 14

Definition

Witness: already visited node accessed by LZ leaf.

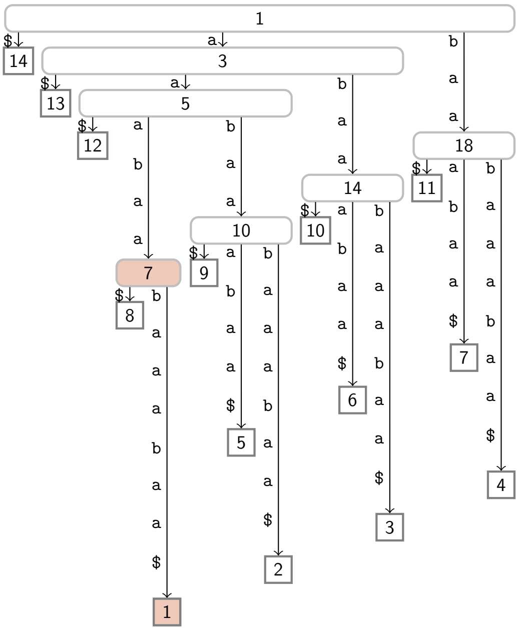


LZ77 parsing of
aaabaabaaabaa :
a |aa |b |aabaa |abaa |$.

Starting positions of
factors (≡ leaf labels):
1, 2, 4, 5, 10, 14

Witnesses:
5, 10, 14

Definition
Witness: already visited node accessed by LZ leaf.

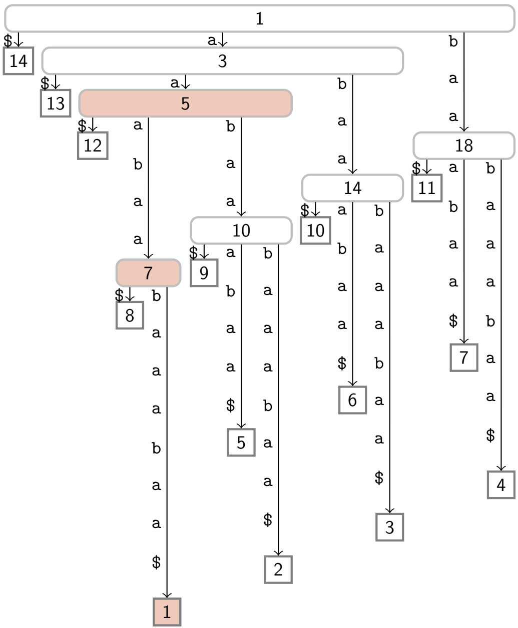


LZ77 parsing of
aaabaabaaabaa :
a |aa |b |aabaa |abaa |$.

Starting positions of
factors (≡ leaf labels):
1, 2, 4, 5, 10, 14

Witnesses:
5, 10, 14

Definition
Witness: already visited
node accessed by LZ leaf.

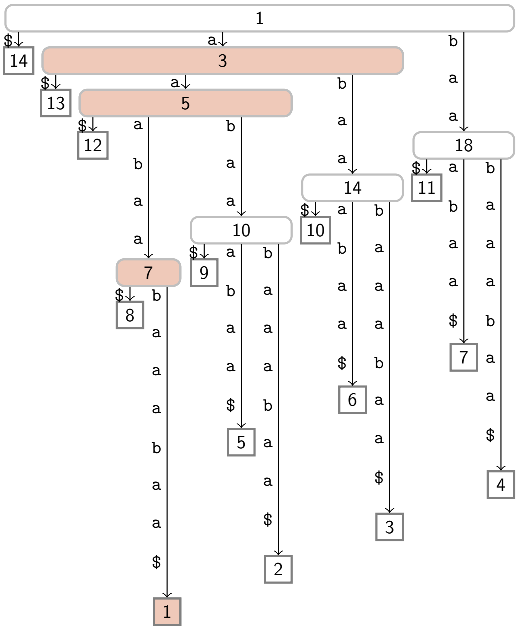


LZ77 parsing of
aaabaabaaabaa :
a |aa |b |aabaa |abaa |$.

Starting positions of
factors ($\equiv$ leaf labels):
1, 2, 4, 5, 10, 14

Witnesses:
5, 10, 14

Definition
Witness: already visited node accessed by LZ leaf.



LZ77 parsing of

aaabaabaaabaa :

a |aa |b |aabaa |abaa |$.

Starting positions of factors (≡ leaf labels):

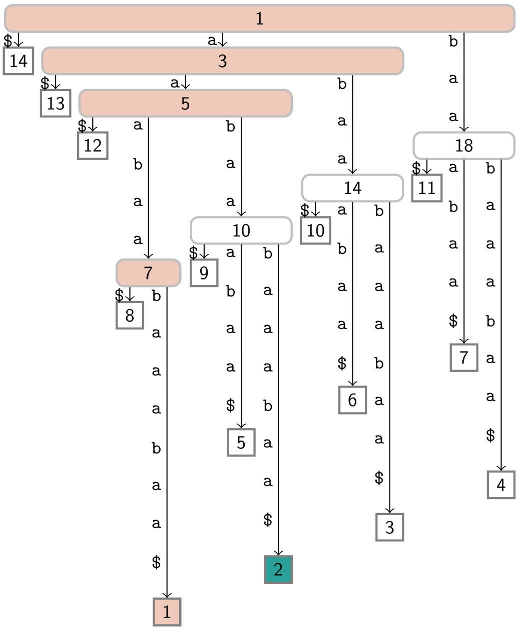
1, 2, 4, 5, 10, 14

Witnesses:

5, 10, 14

Definition

Witness: already visited node accessed by LZ leaf.

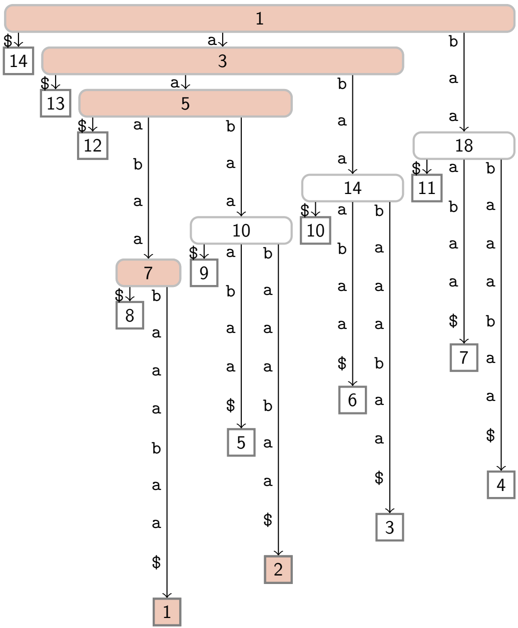


LZ77 parsing of
aaabaabaaabaa :
a |aa |b |aabaa |abaa |$.

Starting positions of
factors (≡ leaf labels):
1, 2, 4, 5, 10, 14

Witnesses:
5, 10, 14

Definition
Witness: already visited node accessed by LZ leaf.

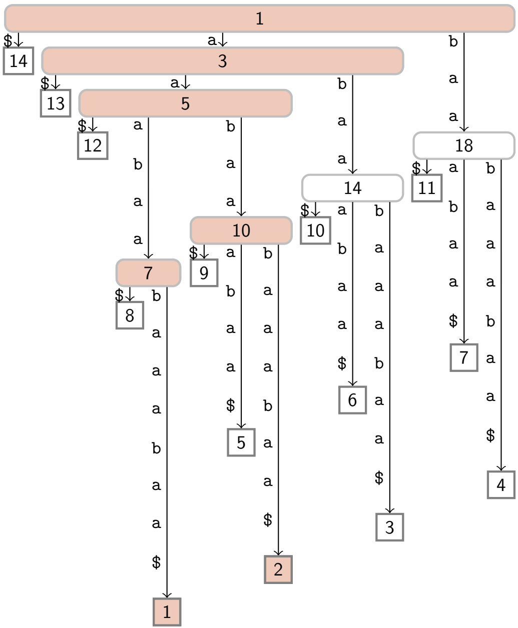


LZ77 parsing of
aaabaabaaabaa :
a |aa |b |aabaa |abaa |$.

Starting positions of
factors (≡ leaf labels):
1, 2, 4, 5, 10, 14

Witnesses:
5, 10, 14

Definition
Witness: already visited node accessed by LZ leaf.

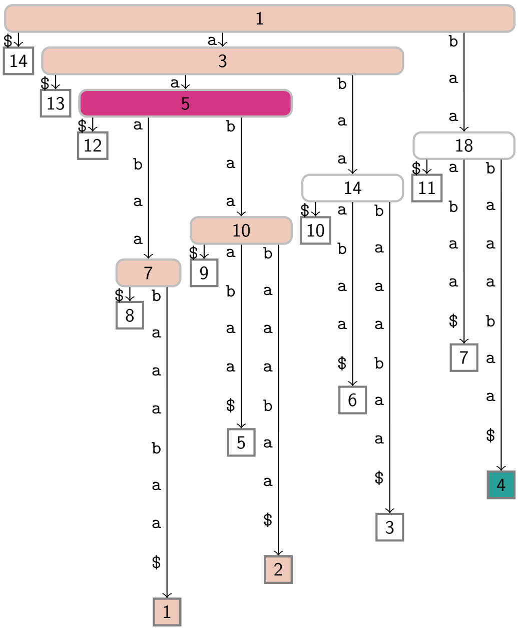


LZ77 parsing of
`aaabaabaaabaa` :
`a |aa |b |aabaa |abaa |$.`

Starting positions of factors (≡ leaf labels):
1, 2, 4, 5, 10, 14

Witnesses:
5, 10, 14

Definition
Witness: already visited node accessed by LZ leaf.

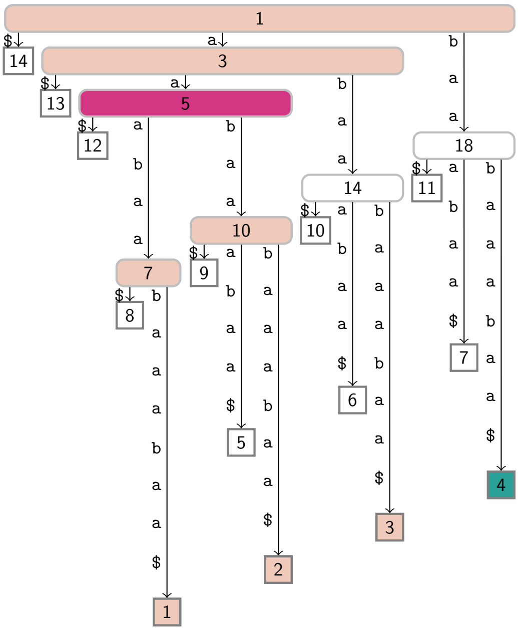


LZ77 parsing of
aaabaabaaabaa :
a |aa |b |aabaa |abaa |$.

Starting positions of factors (≡ leaf labels):
1, 2, 4, 5, 10, 14

Witnesses:
5, 10, 14

Definition
Witness: already visited node accessed by LZ leaf.

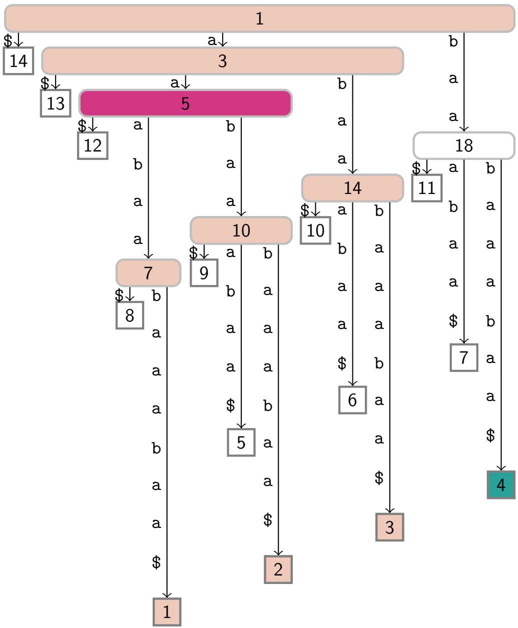


LZ77 parsing of
aaabaabaaabaa :
a |aa |b |aabaa |abaa |$.

Starting positions of factors (≡ leaf labels):
1, 2, 4, 5, 10, 14

Witnesses:
5, 10, 14

Definition
Witness: already visited node accessed by LZ leaf.

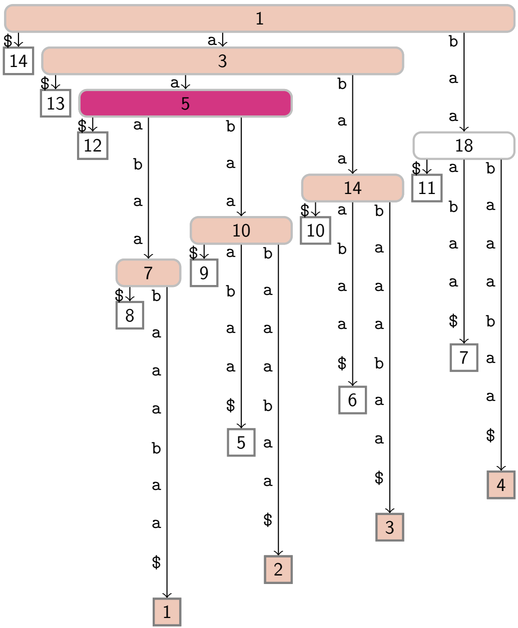


LZ77 parsing of
`aaabaabaaabaa` :
`a |aa |b |aabaa |abaa |$.`

Starting positions of factors ($\equiv$ leaf labels):
1, 2, 4, 5, 10, 14

Witnesses:
5, 10, 14

Definition
Witness: already visited node accessed by LZ leaf.

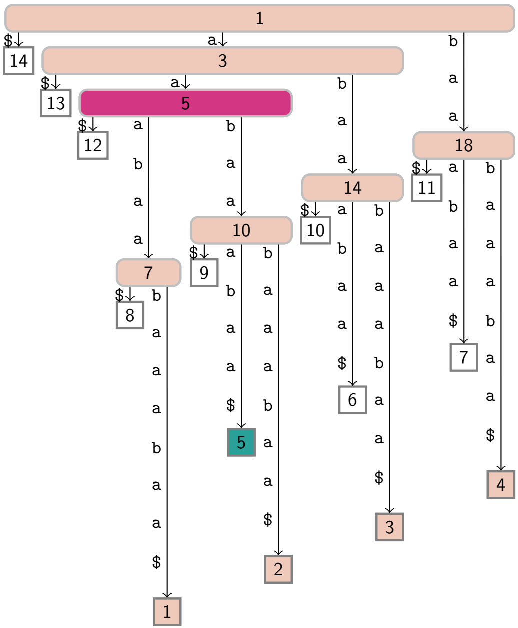


LZ77 parsing of
aaabaabaaabaa :
a |aa |b |aabaa |abaa |$.

Starting positions of factors (≡ leaf labels):
1, 2, 4, 5, 10, 14

Witnesses:
5, 10, 14

Definition
Witness: already visited node accessed by LZ leaf.

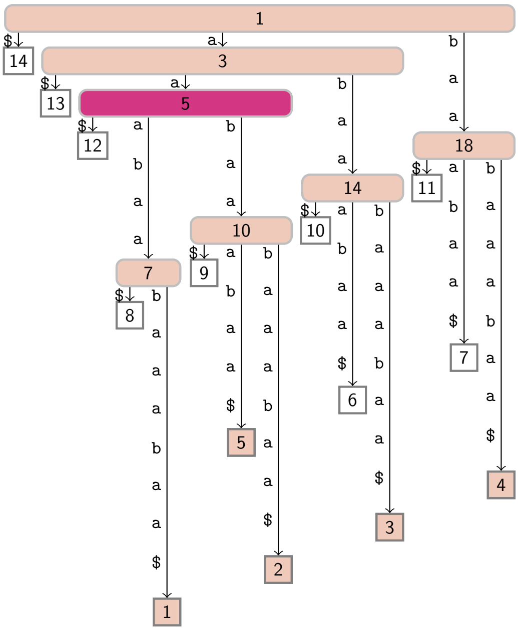


LZ77 parsing of
aaabaabaaabaa :
a |aa |b |aabaa |abaa |$.

Starting positions of factors (≡ leaf labels):
1, 2, 4, 5, 10, 14

Witnesses:
5, 10, 14

Definition
Witness: already visited node accessed by LZ leaf.

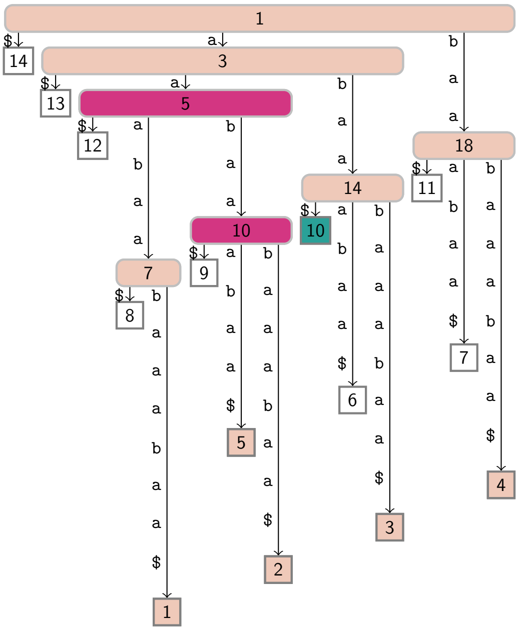


LZ77 parsing of
aaabaabaaabaa :
a |aa |b |aabaa |abaa |$.

Starting positions of factors (≡ leaf labels):
1, 2, 4, 5, 10, 14

Witnesses:
5, 10, 14

Definition
Witness: already visited node accessed by LZ leaf.



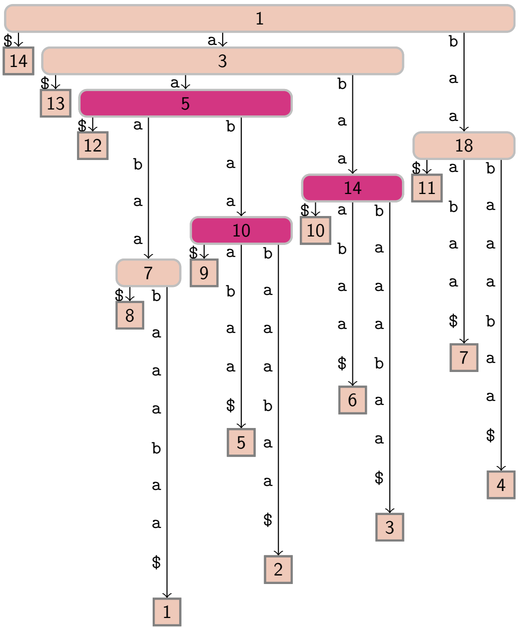
LZ77 parsing of
`aaabaabaaabaa` :
`a |aa |b |aabaa |abaa |$.`

Starting positions of
factors (≡ leaf labels):
1, 2, 4, 5, 10, 14

Witnesses:
5, 10, 14

Definition

Witness: already visited node accessed by LZ leaf.



LZ77 parsing of
`aaabaabaaabaa` :
`a |aa |b |aabaa |abaa |$.`

Starting positions of
factors (≡ leaf labels):
1, 2, 4, 5, 10, 14

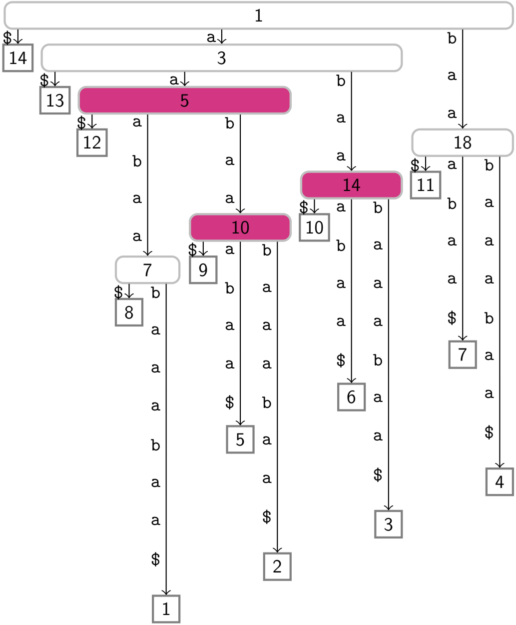
Witnesses:
5, 10, 14

Definition
Witness: already visited node accessed by LZ leaf.

LZ77

Two passes over the suffix tree:

1. Pass:
   - Traverse from *every* leaf to the root
   - Mark visited nodes
   - Already marked nodes $\equiv$ some reference
   - These nodes witness references
2. Pass:
   - Same procedure
   - We know witnesses already!
   - Which leaf discovers a witness first?



LZ77 parsing of `T` = `aaabaabaaabaa` as

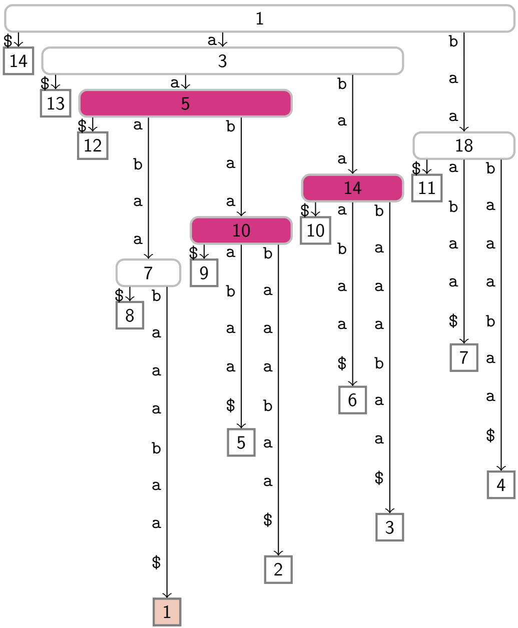
`a |aa |b |aabaa |abaa |$.`

Starting Positions of Factors (≡ Leaf Labels):
1 ,2 ,4 ,5 ,10 ,14

Witnesses:
5, 10, 14

Map text positions to nodes:

1	2		3	5	10
5	10	5	14	10	14



LZ77 parsing of `T` = `aaabaabaaabaa` as

`a |aa |b |aabaa |abaa |$.`

Starting Positions of Factors (≡ Leaf Labels):

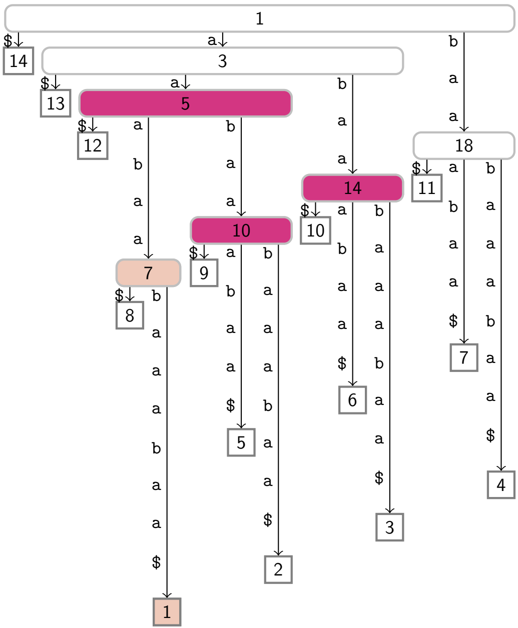
1 ,2 ,4 ,5 ,10 ,14

Witnesses:

5, 10, 14

Map text positions to nodes:

1	2		3	5	10
5	10	5	14	10	14



LZ77 parsing of `T` = `aaabaabaaabaa` as

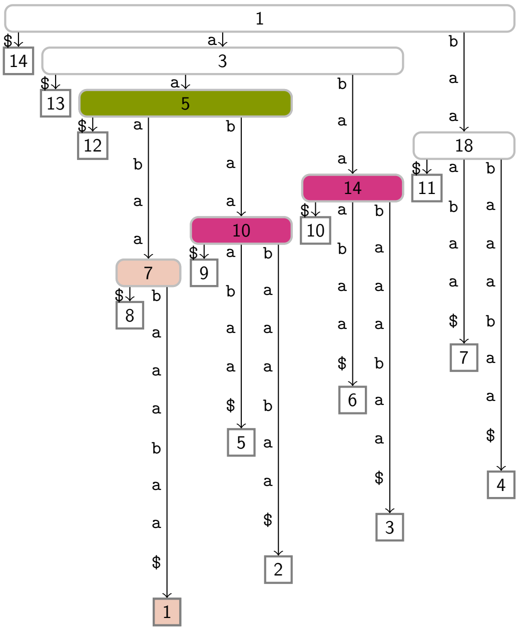
`a |aa |b |aabaa |abaa |$.`

Starting Positions of Factors (≡ Leaf Labels): 1 ,2 ,4 ,5 ,10 ,14

Witnesses:
5, 10, 14

Map text positions to nodes:

1	2		3	5	10
5	10	5	14	10	14



LZ77 parsing of `T` = `aaabaabaaabaa` as

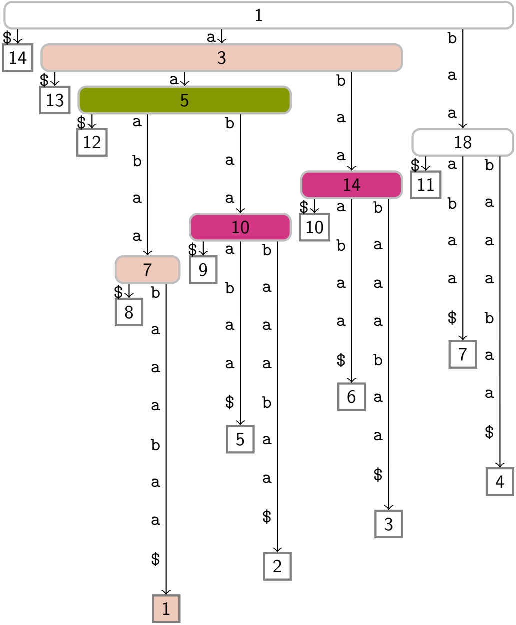
`a |aa |b |aabaa |abaa |$.`

Starting Positions of Factors (≡ Leaf Labels): 1 ,2 ,4 ,5 ,10 ,14

Witnesses:
5, 10, 14

Map text positions to nodes:

1	2		3	5	10
5	10	5	14	10	14



LZ77 parsing of `T` = `aaabaabaaabaa` as

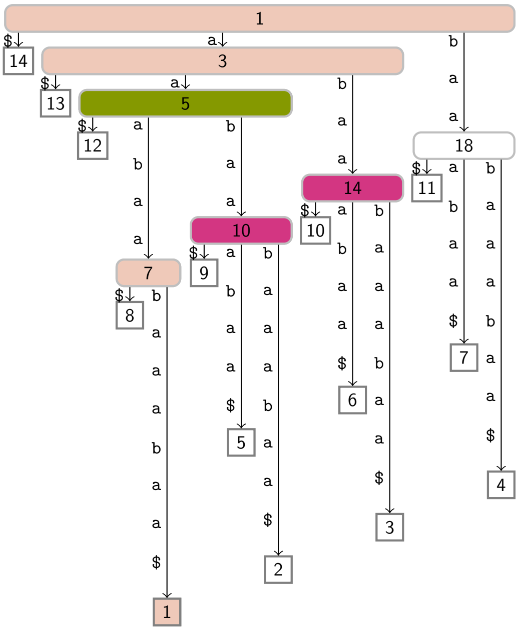
`a |aa |b |aabaa |abaa |$.`

Starting Positions of Factors (≡ Leaf Labels): 1 ,2 ,4 ,5 ,10 ,14

Witnesses: 5, 10, 14

Map text positions to nodes:

1	2		3	5	10
5	10	5	14	10	14



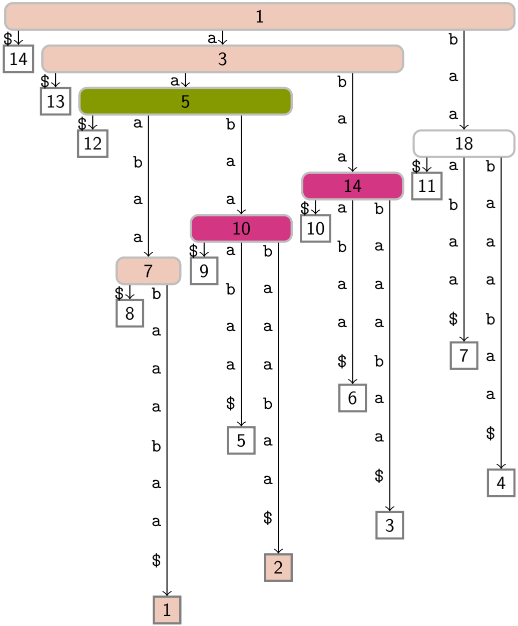
LZ77 parsing of `T` = `aaabaabaaabaa` as
`a |aa |b |aabaa |abaa |$`.

Starting Positions of Factors (≡ Leaf Labels):
1 ,2 ,4 ,5 ,10 ,14

Witnesses:
5, 10, 14

Map text positions to nodes:

1	2		3	5	10
5	10	5	14	10	14



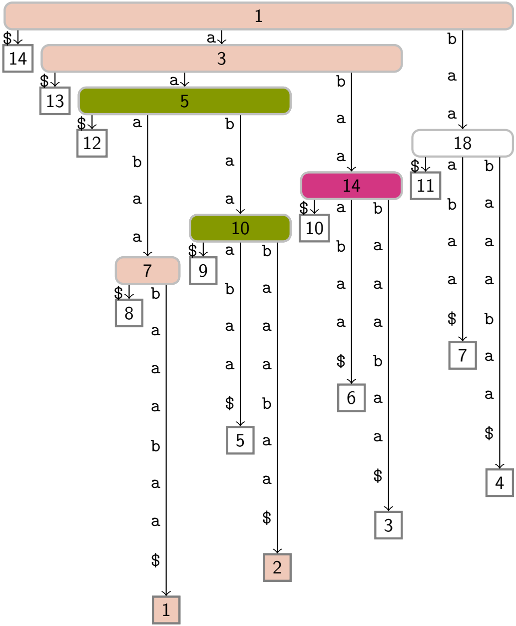
LZ77 parsing of `T` = `aaabaabaaabaa` as
`a |aa |b |aabaa |abaa |$`.

Starting Positions of Factors (≡ Leaf Labels): 1 ,2 ,4 ,5 ,10 ,14

Witnesses: 5, 10, 14

Map text positions to nodes:

1	2		3	5	10
5	10	5	14	10	14



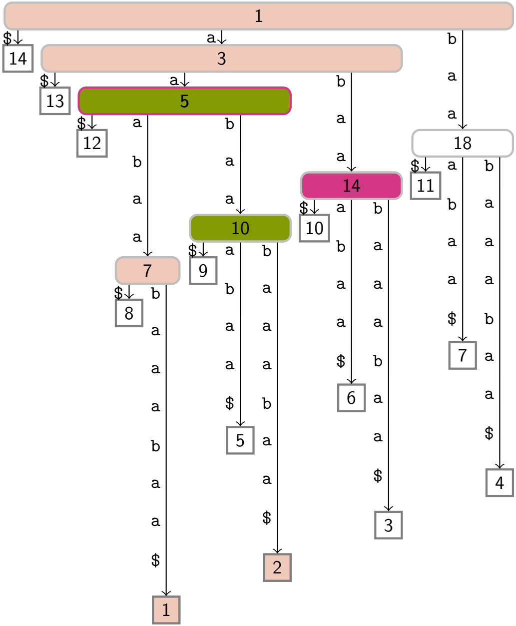
LZ77 parsing of `T` = `aaabaabaaabaa` as
`a |aa |b |aabaa |abaa |$.`

Starting Positions of Factors (≡ Leaf Labels): 1 ,2 ,4 ,5 ,10 ,14

Witnesses:
5, 10, 14

Map text positions to nodes:

1	2		3	5	10
5	10	5	14	10	14



LZ77 parsing of T = aaabaabaaabaa as

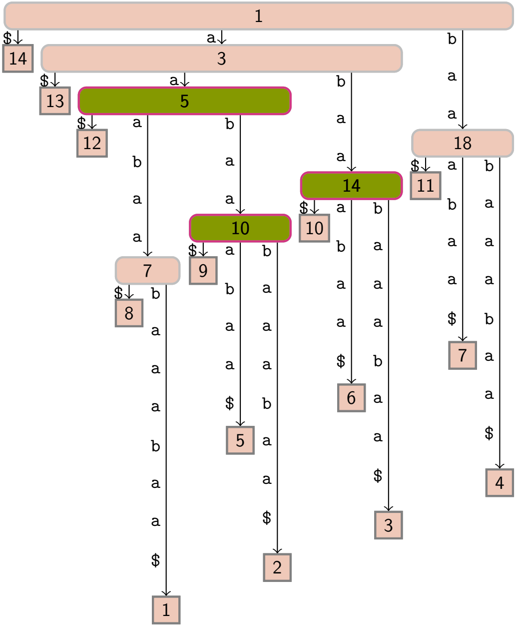
a |aa |b |aabaa |abaa |$.

Starting Positions of Factors (≡ Leaf Labels): 1 ,2 ,4 ,5 ,10 ,14

Witnesses: 5, 10, 14

Map text positions to nodes:

1	2		3	5	10
5	10	5	14	10	14



LZ77 parsing of T = aaabaabaaabaa as a |aa |b |aabaa |abaa |$.

Starting Positions of Factors (≡ Leaf Labels): 1 ,2 ,4 ,5 ,10 ,14

Witnesses: 5, 10, 14

Map text positions to nodes:

1	2		3	5	10
5	10	5	14	10	14

Find Reference Starting Positions

Text positions

1	2	2	3	5	10
5	10	5	14	10	14

Witnesses

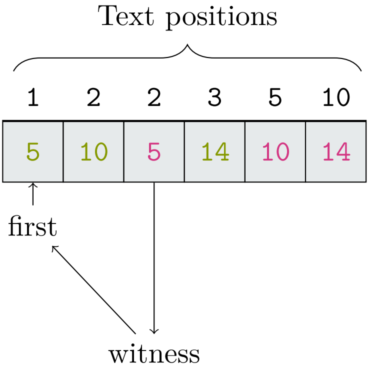
Match factors with referencing text positions.

- Take witness of factor.
- Search first occurence of witness.
- We found the referring text position!

By this we get:

- Factor at text-pos 2 refers to pos 1.
- Factor at text-pos 5 refers to pos 2.

Find Reference Starting Positions



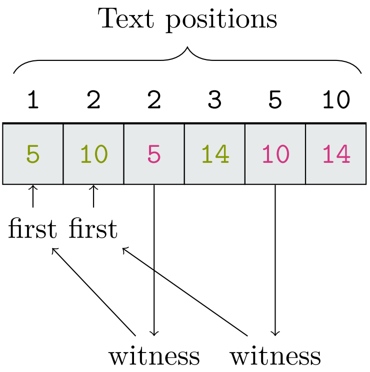
Match factors with referencing text positions.

- Take witness of factor.
- Search first occurence of witness.
- We found the referring text position!

By this we get:

- Factor at text-pos 2 refers to pos 1.
- Factor at text-pos 5 refers to pos 2.

Find Reference Starting Positions



Match factors with referencing text positions.

- Take witness of factor.
- Search first occurence of witness.
- We found the referring text position!

By this we get:

- Factor at text-pos 2 refers to pos 1.
- Factor at text-pos 5 refers to pos 2.

Tricks & Techniques

Main Idea

Use Suffix Tree (ST):

- Take leaves by text position.
- String depth reveals factor length.
- Compute references indirectly by nodes called witnesses.

Auxiliary Data Structures

Build a lightweight Suffix Tree (ST) with:

- Enhanced Suffix Array
- DFUDS tree topology
- MinMax tree for ST navigation

DS	space in bits	constr. time	constr. space	Authors
SA	$n \lg n$	$\mathcal{O}(n/\epsilon^2)$	$(1+\epsilon)n \lg n$	Kärkkäinen et al. '06
LCP	$2n + o(n)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	Välimäki et al. '09
DFUDS	$4n + 4$	$\mathcal{O}(n)$	$n + o(n)$	Ohlebusch et al. '10
MinMax	$o(n)$	$o(n)$	$o(n)$	Navarro,Sadakane'14
RMQ	$2n + o(n)$	$\mathcal{O}(n)$	$n + o(n)$	Fischer'10
total	$n \lg n + \mathcal{O}(n)$	$\mathcal{O}(n)$	$(1+\epsilon)n \lg n + \mathcal{O}(n)$	
result		$\mathcal{O}(n)$	$(1+\epsilon)n \lg n + \mathcal{O}(n)$	this paper

Two heavyweight arrays

SA ST string depth.

- $+$ LCP: Factor length

ISA fetch ST leaf

- LZ scans text linear from left to right
- ST leaves in ISA order $=$ LZ parsing order

Problem

Cannot store SA, ISA *and* **output** *in* $(1+\epsilon)n \lg n$ *bits!*

DS for SA and ISA

But with a DS of Munro et al.'12:

Theorem

Given a permutation $A[1..n]$, A uses $n \lg n$ bits.
The array+inverse of A

- *answers A^{-1} in $\mathcal{O}(1/\epsilon)$ time.*
- *uses additional $\epsilon n \lg n$ bits*
- *is built in $\mathcal{O}(n)$ time.*

Managing Space

Store two arrays in $(1+\epsilon)n \lg n$ bits:

- $n \lg n$ bits for
  - SA (in the beginning)
  - ISA (invert SA)
  - storing witnesses (indirect references)
- $\epsilon n \lg n$ bits for
  - array+inverse: SA with $\mathcal{O}(1/\epsilon)$ access time
  - helper array for comparisons

Summary

Theorem

Given text T of length n. LZ77 and LZ78 of T can be computed

- *with $\mathcal{O}(n)$ time*
- *with $(1+\epsilon)n \lg n + \mathcal{O}(n)$ bits space*
- *without extra output space! (see paper)*

Techniques used:

- Succinct, but fast ST.
- Storing SA and ISA in $(1+\epsilon)n \lg n$ bits.
- Indirect Matching (witnesses).

Thank you for listening. Any questions are welcome!

Summary

Theorem

Given text T of length n. LZ77 and LZ78 of T can be computed

- *with $\mathcal{O}(n)$ time*
- *with $(1+\epsilon)n \lg n + \mathcal{O}(n)$ bits space*
- *without extra output space! (see paper)*

Techniques used:

- Succinct, but fast ST.
- Storing SA and ISA in $(1+\epsilon)n \lg n$ bits.
- Indirect Matching (witnesses).

Thank you for listening. Any questions are welcome!