Computation of Variations of the LZ77 factorization and the LPF Array with Suffix Trees

Dominik Köppl

M&D Data Science Center, Tokyo Medical and Dental University, Japan

results of:

- "Non-Overlapping LZ77 Factorization and LZ78 Substring Compression Queries with Suffix Trees." *Algorithms* 14(2): 44 (2021)
- "Reversed Lempel-Ziv Factorization with Suffix Trees." *Algorithms* 14(6): 161 (2021)

in this talk

variations of Lempel-Ziv 77 (LZ77) factorization:

- non-overlapping Lempel-Ziv 77 (NOV LZ)
- reversed LZ — Kolpakov, Kucherov'09

variations of longest previous factor array (LPF):

- LPnF: longest previous non-overlapping factor array
- LPnrF: longest previous non-overlapping reversed factor array

our contribution:

- $2n$-bit representations of LPnF and LPnrF
- (near) linear-time algorithms computing
  the mentioned factorizations/arrays in small space

setting & example

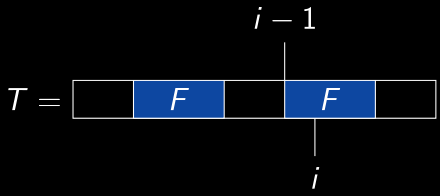
- T: input text, $n - 1 := |T|$ length of T
- Σ: alphabet of T, $\sigma := |\Sigma|$ size of Σ
- $\$ < c \; \forall \; c \in \Sigma$

i	1	2	3	4	5	6	7	8	9	10	11
$T\$$	a	b	b	a	b	b	a	b	a	b	$
LPF	0	0	1	5	4	3	2	3	2	1	0
LPnF	0	0	1	3	3	3	2	3	2	1	0
LPnrF	0	0	2	1	3	3	2	3	2	1	0

succinct representation

Sadakane'07: $2n$-bit representation PLCP array

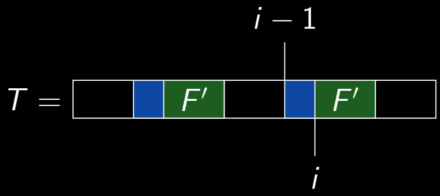
- PLCP[i]: longest common prefix of $(T\$)[i..]$ with its lexicographically preceding suffix
- $\text{PLCP}[n] = 0$ since $(T\$)[n] = \$$
- $\text{PLCP}[i] \leq n \;\forall\; i \in [1..n]$
- $\text{PLCP}[i] \geq \text{PLCP}[i-1] - 1$



succinct representation

Sadakane'07: $2n$-bit representation PLCP array

- PLCP$[i]$: longest common prefix of $(T\$)[i..]$ with its lexicographically preceding suffix
- PLCP$[n] = 0$ since $(T\$)[n] = \$$
- PLCP$[i] \leq n \; \forall \; i \in [1..n]$
- PLCP$[i] \geq$ PLCP$[i-1] - 1$



succinct representation

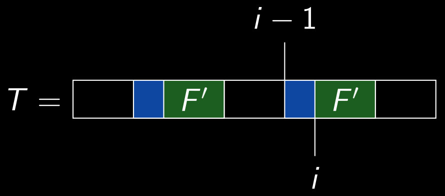
Sadakane'07: $2n$-bit representation PLCP array

- PLCP[i]: longest common prefix of $(T\$)[i..]$ with its lexicographically preceding suffix
- $\mathrm{PLCP}[n] = 0$ since $(T\$)[n] = \$$
- $\mathrm{PLCP}[i] \leq n \; \forall \; i \in [1..n]$
- $\mathrm{PLCP}[i] \geq \mathrm{PLCP}[i-1] - 1$
- hence: $\mathrm{PLCP}[i] - \mathrm{PLCP}[i-1] + 1 \geq 0$.

$\Rightarrow$ store all values of $\mathrm{PLCP}[i] - \mathrm{PLCP}[i-1] + 1$ in unary:

- $\sum_{i=1}^{n} (\mathrm{PLCP}[i] - \mathrm{PLCP}[i-1] + 1) = \mathrm{PLCP}[n] + n = n$ with $\mathrm{PLCP}[0] := 0$
- need n '1's and n '0's

$\Rightarrow$ $2n$ bits.



related arrays with same representation

Ref.	array	factorization
Sadakane'07	PLPF	lcpcomp (Dinklage+'17)
Belazzougui and Cunial'14	matching statistics	Relative LZ (Ziv+'93)
Bannai,Inenaga,K.'17	LPF	LZ77
this talk	LPnF	NOV LZ
this talk	LPnrF	reversed LZ (Kolpakov+'09)

- NOV LZ: Non OVerlapping Lempel-Ziv 77
- Ref.: first occurrence of $2n$-bit representation

previous work

LPnF:

Ref.	time	bits
Crochemore, Tischler'11	$\mathcal{O}(n)$	$\mathcal{O}(n \lg n)$
Crochemore+'12	$\mathcal{O}(n)$	$\mathcal{O}(n \lg n)$
Ohlebusch, Weber'19:	$\mathcal{O}(n)$	$\mathcal{O}(n \lg n)$

LPnrF:

Ref.	time	bits
Kolpakov, Kucherov'09	$\mathcal{O}(n \lg \sigma)$	$\mathcal{O}(n \lg n)$
Chairungsee, Crochemore'09	$\mathcal{O}(n \lg \sigma)$	$\mathcal{O}(n \lg n)$
Sugimoto+'16	$\mathcal{O}(n \lg^2 \sigma)$	$\mathcal{O}(n \lg \sigma)$
Crochemore+'12	$\mathcal{O}(n)$	$\mathcal{O}(n \lg n)$

- with $2n$-bit representation: $\mathcal{O}(n \lg n)$ bits of working space no longer optimal!
- can we improve working space while not sacrificing time (too much)?

our results

- $\epsilon > 0$ selectable constant
- basic time: $\mathcal{O}(\epsilon^{-1} n)$
- $t_{\mathsf{SA}} = \log_\sigma^\epsilon n$: suffix array query

structure	bits	multiplicative time penalty
NOV LZ or LPnF	$(1+\epsilon)n \lg n + \mathcal{O}(n)$ $\mathcal{O}(\epsilon^{-1} n \lg \sigma)$	$\mathcal{O}(1)$ $\mathcal{O}(t_{\mathsf{SA}})$
LPnrF	$(2+\epsilon)n \lg n + \mathcal{O}(n)$ $\mathcal{O}(\epsilon^{-1} n \lg \sigma)$	$\mathcal{O}(1)$ $\mathcal{O}(t_{\mathsf{SA}})$
reversed LZ	$(2+\epsilon)n \lg n + \mathcal{O}(n)$ $\mathcal{O}(\epsilon^{-1} n \lg \sigma)$	$\mathcal{O}(1)$ $\mathcal{O}(1)$

NOV LZ factorization

NOV LZ factorization

$$T = \texttt{a b b a b b a b a b}$$

1	2	3	4	5	6	7	8	9	10

Coding:

NOV LZ factorization

$$T = \boxed{\mathtt{a}}\,\mathtt{b}\,\mathtt{b}\,\mathtt{a}\,\mathtt{b}\,\mathtt{b}\,\mathtt{a}\,\mathtt{b}\,\mathtt{a}\,\mathtt{b}$$

1 2 3 4 5 6 7 8 9 10

Coding: a

NOV LZ factorization

$T = \boxed{\mathtt{a}}\,\boxed{\mathtt{b}}\,\mathtt{b}\,\mathtt{a}\,\mathtt{b}\,\mathtt{b}\,\mathtt{a}\,\mathtt{b}\,\mathtt{a}\,\mathtt{b}$

1 2 3 4 5 6 7 8 9 10

Coding: ab

NOV LZ factorization

$(2,1)$

$T = \texttt{a}\,\texttt{b}\,\texttt{b}\,\texttt{a}\,\texttt{b}\,\texttt{b}\,\texttt{a}\,\texttt{b}\,\texttt{a}\,\texttt{b}$

1 2 3 4 5 6 7 8 9 10

Coding: $\texttt{ab}(2,1)$

NOV LZ factorization

(2,1)

(1,3)

$T =$ a b b a b b a b a b

1 2 3 4 5 6 7 8 9 10

Coding: ab(2,1)(1,3)

NOV LZ factorization

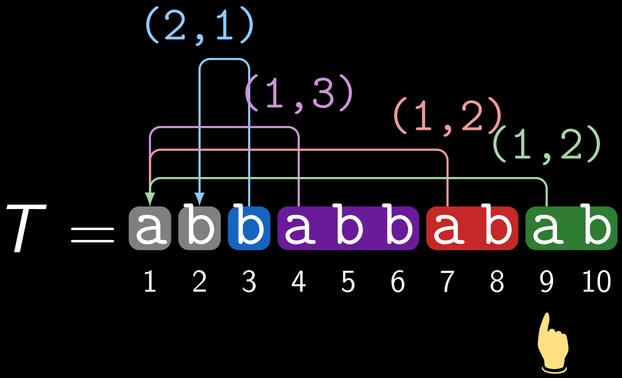
(2,1) (1,3) (1,2)

$T =$ a b b a b b a b a b

a	b	b	a	b	b	a	b	a	b
1	2	3	4	5	6	7	8	9	10

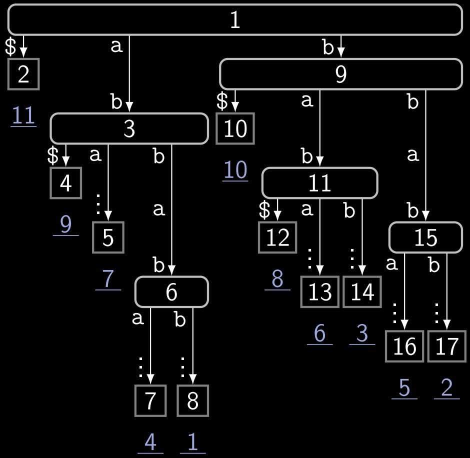
Coding: ab(2,1)(1,3)(1,2)

NOV LZ factorization



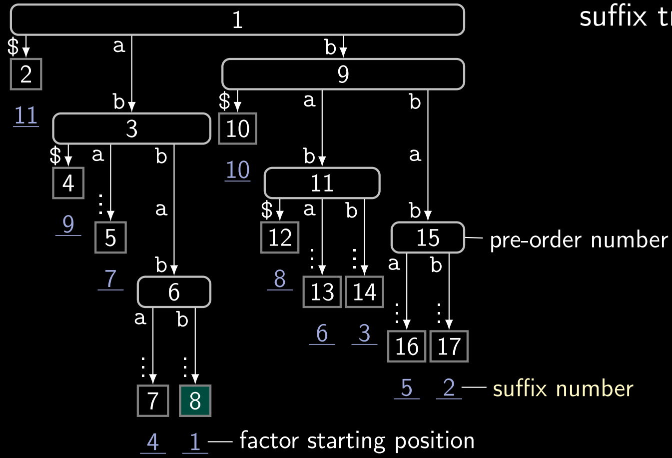
Coding: ab(2,1)(1,3)(1,2)(1,2)

suffix tree of $T\$ = \texttt{abbabbabab\$}$

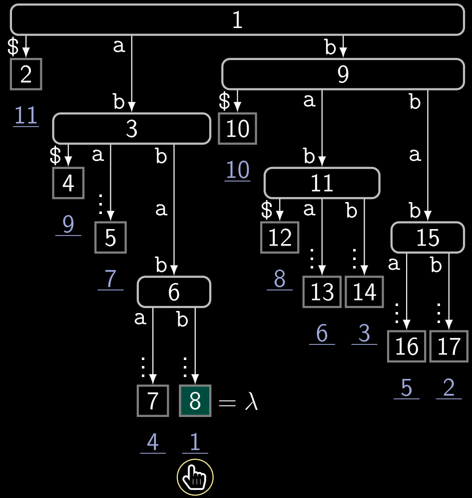


$T = \texttt{abbabbabab}$

suffix tree of $T\$ = \texttt{abbabbabab\$}$



$T = \texttt{abbabbabab}$



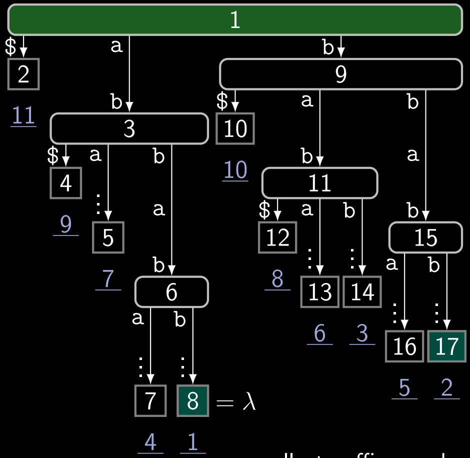
suffix tree of $T\$$ = abbabbabab$

- descend from root to leaf λ with suffix number = factor starting position

T = abbabbabab

suffix tree of $T\$ = \texttt{abbabbabab\$}$

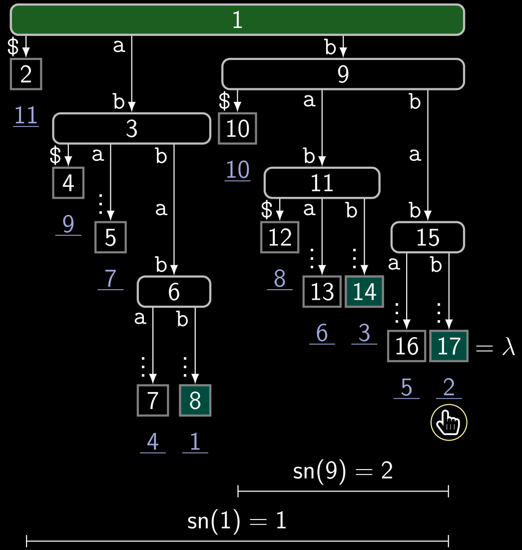
- descend from root to leaf λ with suffix number = factor starting position
- as long as smallest suffix number in subtree is $< \text{sn}(\lambda)$



smallest suffix number in subtree of node 1

$\text{sn}(1) = 1$

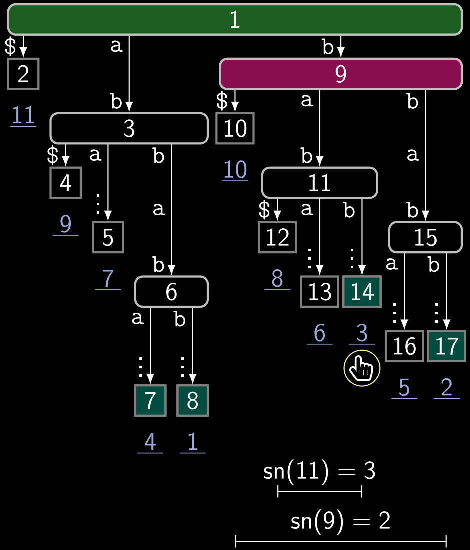
$T = \texttt{a}|\texttt{bbabbabab}$



suffix tree of $T\$ = \mathtt{abbabbabab\$}$

- descend from root to leaf λ with suffix number = factor starting position
- as long as smallest suffix number in subtree is $< \mathrm{sn}(\lambda)$

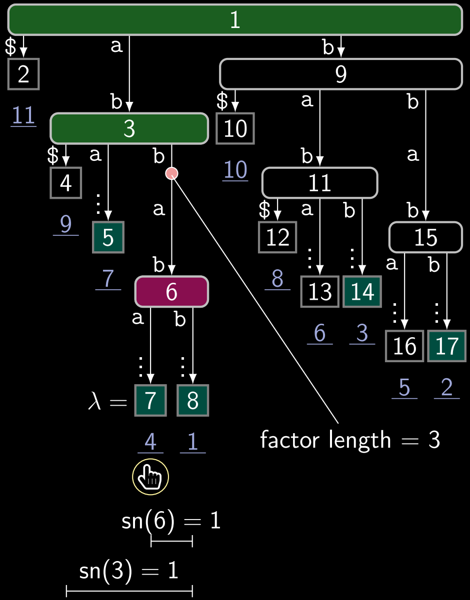
$T = \mathtt{a}|\mathtt{b}|\mathtt{babbabab}$



suffix tree of $T\$ = \texttt{abbababbabab\$}$

- descend from root to leaf λ with suffix number = factor starting position
- as long as smallest suffix number in subtree is $< \mathrm{sn}(\lambda)$
- lowest visited nodes witnesses reference
- previous occurrence must not overlap

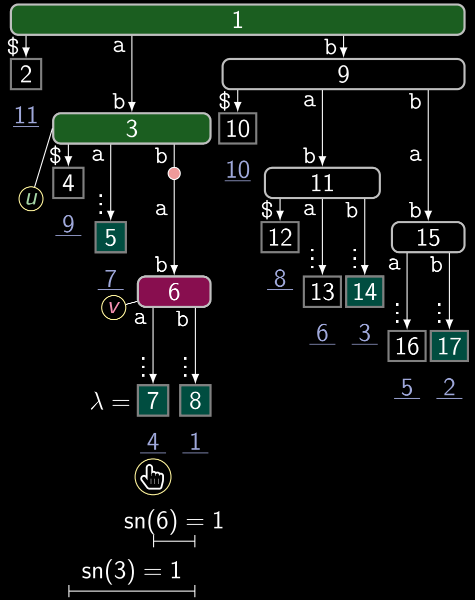
$T = \texttt{a|b|b|abbabab}$



suffix tree of $T\$ = \texttt{abbababbabab\$}$

- descend from root to leaf λ with suffix number = factor starting position
- as long as smallest suffix number in subtree is $< \mathrm{sn}(\lambda)$
- lowest visited nodes witnesses reference
- previous occurrence must not overlap
- otherwise trim overlap

$T = \texttt{a}|\texttt{b}|\texttt{b}|\texttt{abb}|\texttt{abab}$



general case:

- $\mathrm{sn}(w)$: smallest suffix number in subtree of node w
- $\mathrm{strdepth}(w)$: string depth of node w
- node v : lowest traversed node
- node u : v's parent

length is

- $\min(\mathrm{sn}(\lambda) - \mathrm{sn}(v), \mathrm{strdepth}(v))$
- $\min(\mathrm{sn}(\lambda) - \mathrm{sn}(u), \mathrm{strdepth}(u))$

$\Rightarrow$ take the max of both

$T = \mathtt{a}|\mathtt{b}|\mathtt{b}|\mathtt{abb}|\mathtt{abab\$}$

recap

can compute NOV LZ with

- $\mathcal{O}(n)$ calls to $\mathrm{sn}(\cdot)$
- $\mathcal{O}(z)$ calls to $\mathrm{strdepth}(\cdot)$

to compute LPnF:

- treat each leaf as a factor starting position
- use suffix links to omit the traversal from the top

$\Rightarrow$ $\mathcal{O}(n)$ calls to $\mathrm{sn}(\cdot)$ and $\mathrm{strdepth}(\cdot)$

how much time costs a call to $\mathrm{sn}(\cdot)$ or $\mathrm{strdepth}(\cdot)$?

suffix trees

construction by Farach-Colton+'00:

- $\mathcal{O}(n)$ time
- $\mathcal{O}(n \lg n)$ bits working space

space-efficient $\mathcal{O}(n)$ time constructions:

- Fischer+'18:
  - $(1+\epsilon)n \lg n + \mathcal{O}(n)$ bits with $\epsilon \in (0,1]$
  - $t_{\mathsf{SA}} = \mathcal{O}(1/\epsilon)$ time for sn$(\cdot)$ and strdepth$(\cdot)$
- Munro+'17,Belazzougui+'20:
  - $\mathcal{O}(n \lg \sigma)$ bits
  - $t_{\mathsf{SA}} = \log_\sigma^\epsilon n$ with SA sampling
  - alternatively: strdepth$(\cdot)$ in $\mathcal{O}(\text{strdepth}(\cdot))$ time

reversed LZ factorization

reversed LZ

$T = \texttt{a b b a b b a b a b}$

a	b	b	a	b	b	a	b	a	b
1	2	3	4	5	6	7	8	9	10

Coding:

reversed LZ

$T =$

a	b	b	a	b	b	a	b	a	b
1	2	3	4	5	6	7	8	9	10

Coding: a

reversed LZ

$T =$ a b b a b b a b a b

1 2 3 4 5 6 7 8 9 10

Coding: ab

reversed LZ

$(1,2)$

$T = \texttt{a b b a b b a b a b}$

1 2 3 4 5 6 7 8 9 10

Coding: $\texttt{ab(1,2)}$

reversed LZ

(1,2)

(1,3)

$T =$ a b b a b b a b a b

1 2 3 4 5 6 7 8 9 10

Coding: ab(1,2)(1,3)

reversed LZ

(1,2)

(1,3)

(3,3)

$T =$ a b b a b b a b a b

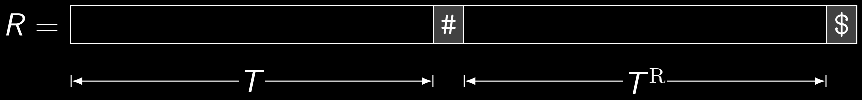
1 2 3 4 5 6 7 8 9 10

Coding: ab(1,2)(1,3)(3,3)

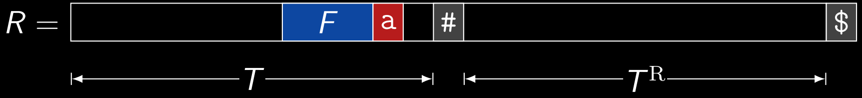
reversed LZ

- T^{R} : reverse of T
- use suffix tree of $R := T\#T^{\mathrm{R}}\$$ with $\$ < \# < c \ \ \forall\, c \in \Sigma$
- key lemma: each factor is the string label of a suffix tree node.

key lemma

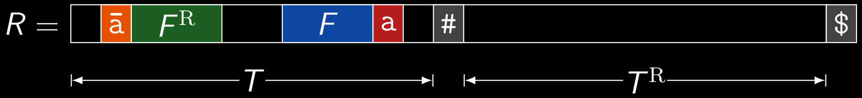


key lemma



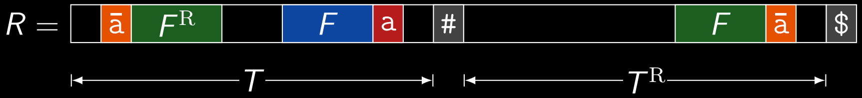
- assume factor F ending before an a with

key lemma



- assume factor F ending before an a with
- reference preceded by $\bar{a} \in \Sigma - \{a\}$ (otherwise F can be prolonged)

key lemma



- assume factor F ending before an a with
- reference preceded by $\bar{a} \in \Sigma - \{a\}$ (otherwise F can be prolonged)
- mirrored reference in T^{R} ends with $\bar{a}$

$\Rightarrow$ R has substrings Fa and $F\bar{a}$

$\Rightarrow$ $\exists$ node with string label F

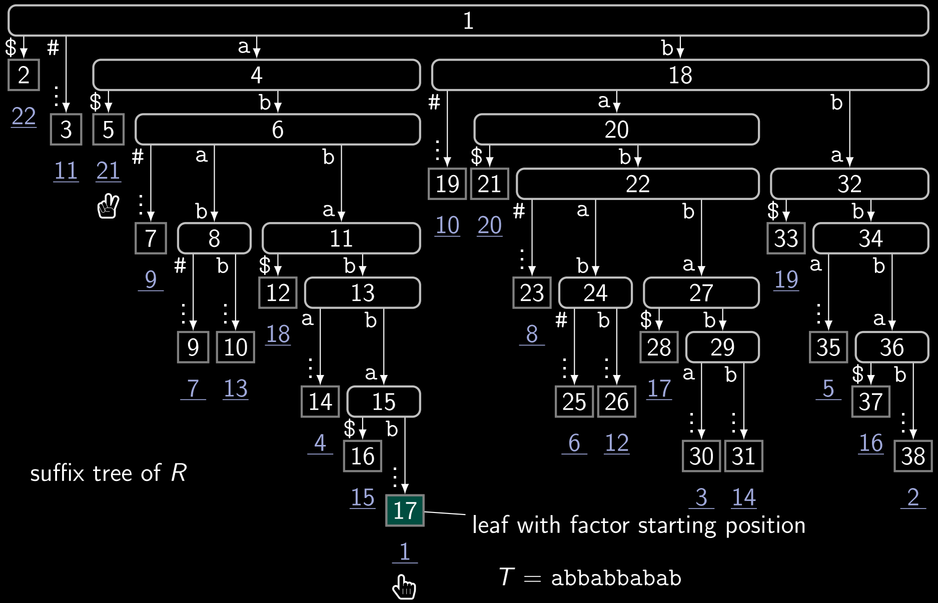
(if F^{R} is prefix of $T \Rightarrow F\$$ is suffix of R)

cooperative 2 player game

each player takes turns at the same pace

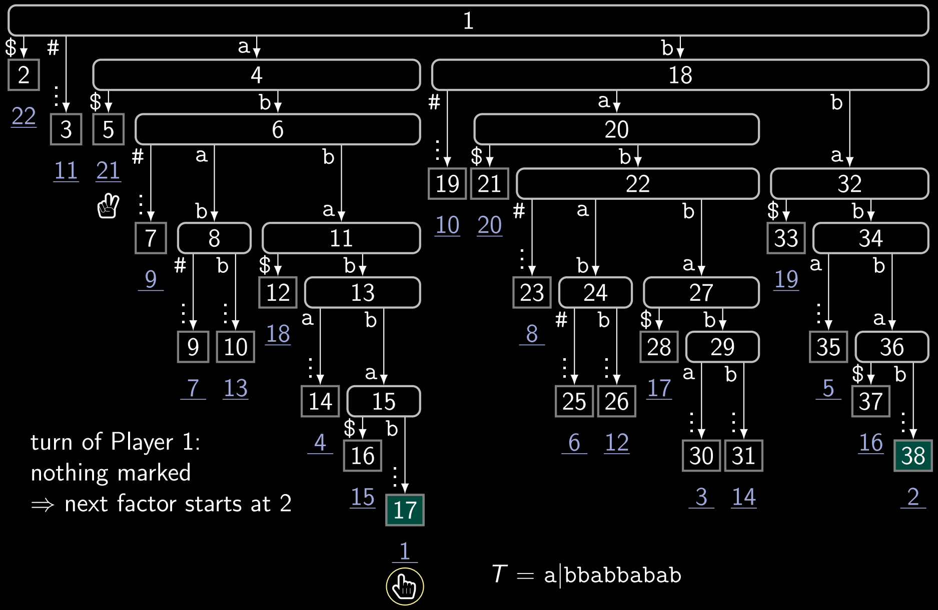
- Player 2
  - selects leaves in descending suffix number order
  - marks all nodes on the path up to the root
- Player 1
  - selects leaves in ascending suffix number order
  - if selected leaf λ starts with a factor F:
    search the lowest marked ancestor v with $\mathrm{strdepth}(v) = |F|$

Player 1 starts.



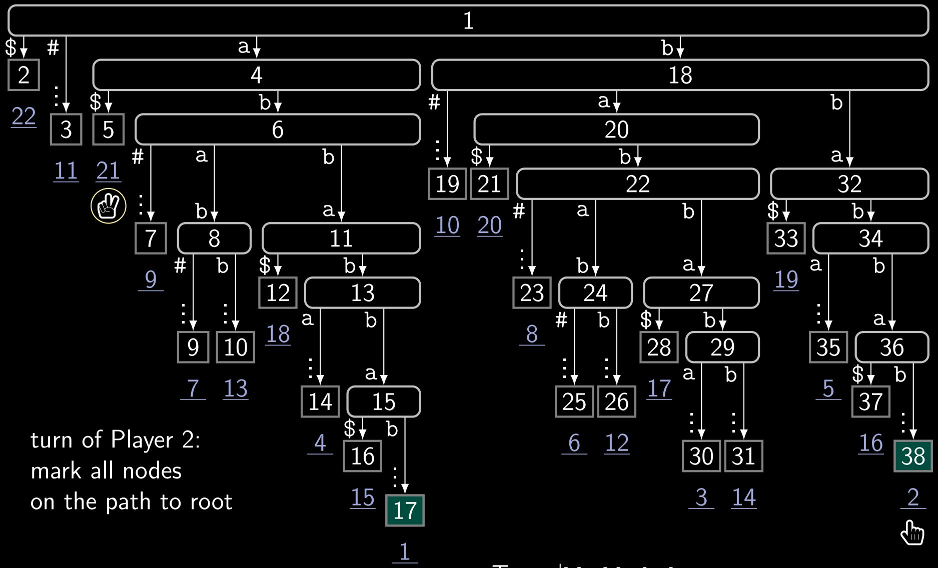
suffix tree of R

T = abbabbabab



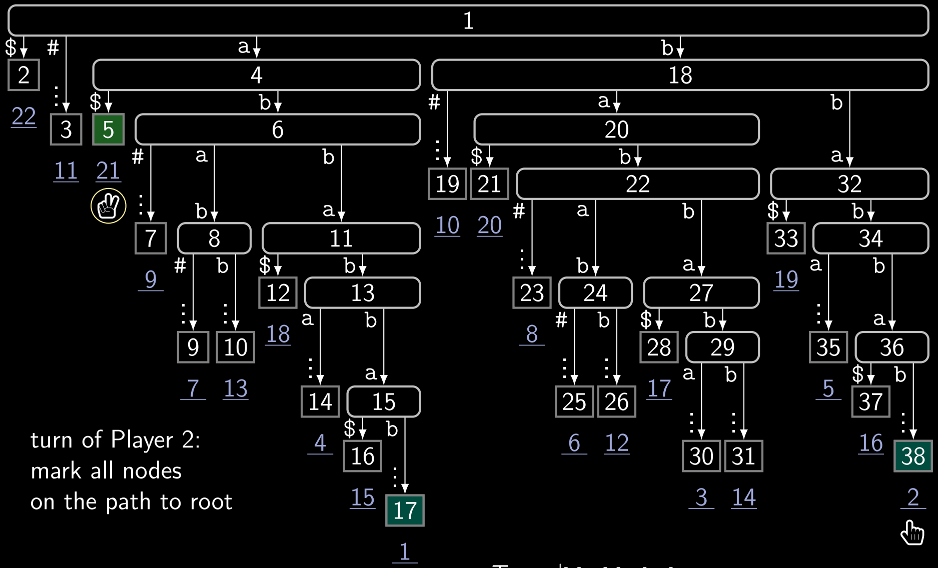
turn of Player 1:
nothing marked
$\Rightarrow$ next factor starts at 2

$T = \mathrm{a}|\mathrm{bbabbabab}$



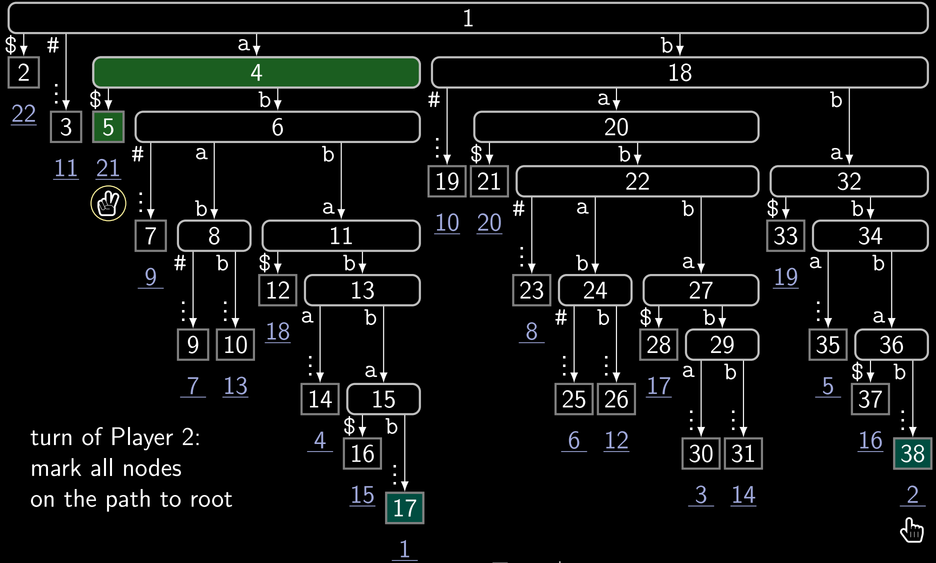
turn of Player 2:
mark all nodes
on the path to root

$T = \mathrm{a}|\mathrm{bbabbabab}$



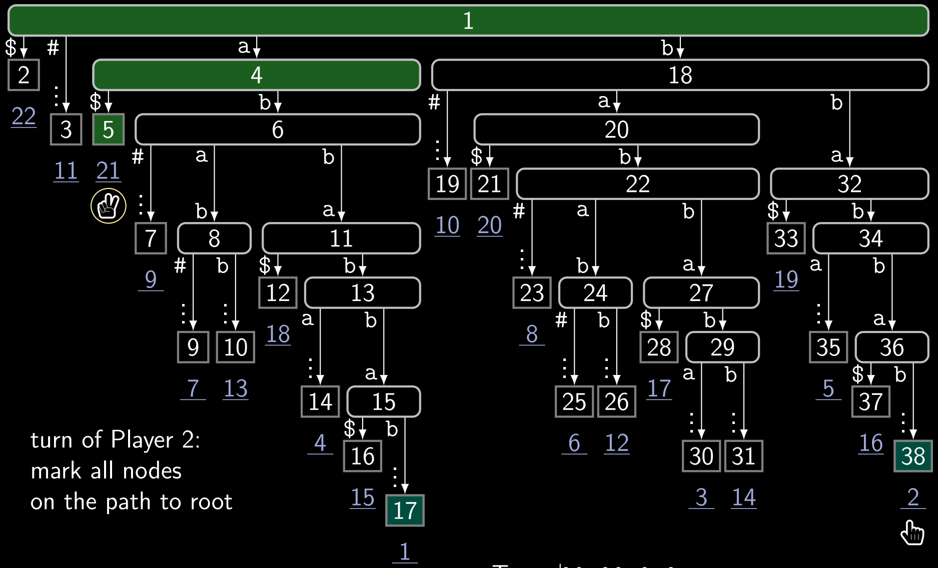
turn of Player 2:
mark all nodes
on the path to root

$T = \mathtt{a|bbabbabab}$



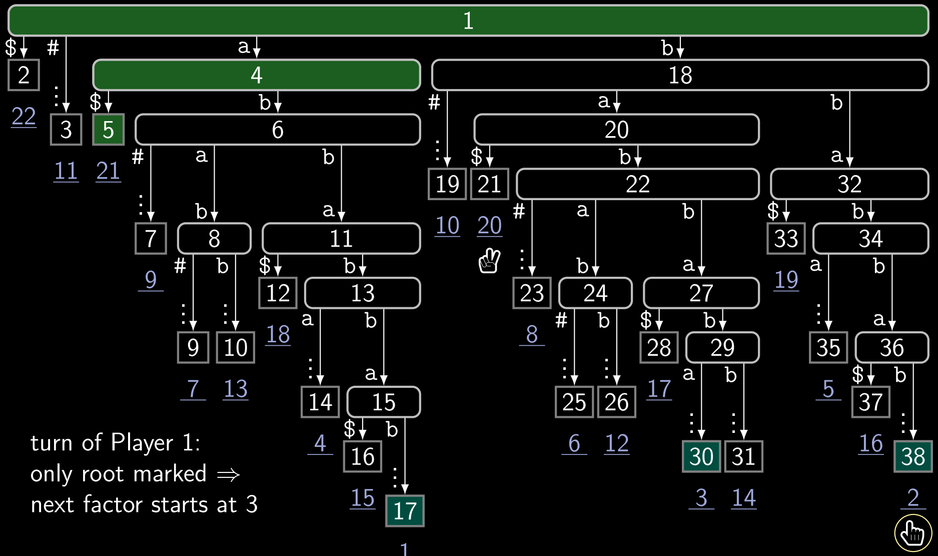
turn of Player 2:
mark all nodes
on the path to root

$T = \mathrm{a}|\mathrm{bbabbabab}$



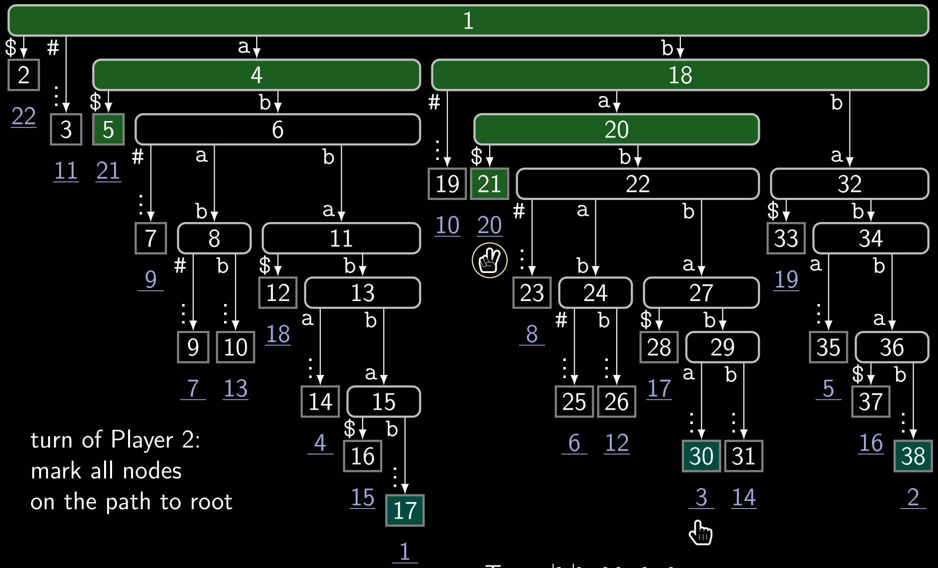
turn of Player 2:
mark all nodes
on the path to root

$T =$ a|bbabbabab



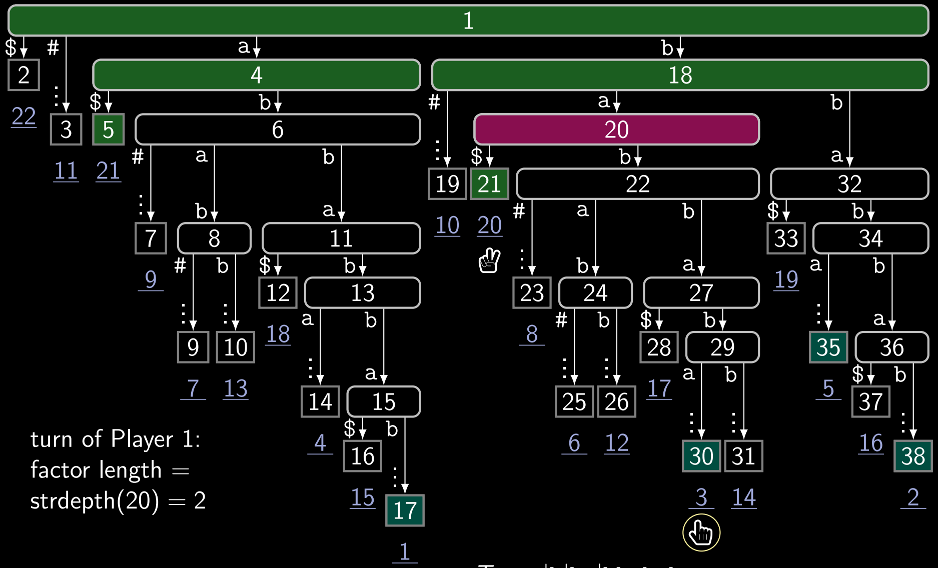
turn of Player 1:
only root marked $\Rightarrow$
next factor starts at 3

$T = \mathtt{a}|\mathtt{b}|\mathtt{babbabab}$



turn of Player 2:
mark all nodes
on the path to root

$T = \mathrm{a}|\mathrm{b}|\mathrm{babbabab}$



turn of Player 1:
factor length =
strdepth(20) = 2

$T = \mathtt{a|b|ba|bbabab}$

analysis

- termination when both players meet (at symbol #)
- Player 2 never marks a node twice $\Rightarrow \mathcal{O}(n)$ node visits
- Player 1 calls $\mathcal{O}(z)$ times strdepth(w), where $z = \#$ factors

find source positions in $\mathcal{O}(z \lg n)$ bits:

- second game, but keeping the red marked nodes
- when Player 2 reaches a red marked node from leaf λ: store there sn(λ)

note: $z = \mathcal{O}(\log_\sigma n) \Rightarrow \mathcal{O}(z \lg n) \subset \mathcal{O}(n \lg \sigma)$
total:

- $\mathcal{O}(n)$ time for reversed LZ with $\sum_{w \in W} \mathrm{strdepth}(w) = n$, where
  - W : all nodes Player 1 queried
- $\mathcal{O}(nt_{\mathsf{SA}})$ time for LPnrF by setting $z \leftarrow n$, and using SA sampling for computing strdepth(w)

open problems

- $\mathcal{O}(n \lg \sigma)$ bits + $\mathcal{O}(n)$ time possible for computing any LP*F table?
  ! Matching statistics can be computed in $\mathcal{O}(n \lg \sigma)$ bits and $\mathcal{O}(n)$ time because the choice for the reference is static, while all LP*F tables need references based on the already processed text.
- LCP array has notion of *irreducible* LCP values with a sum of $\mathcal{O}(n \lg n)$.
  - Are there irreducible LP*F values? If so, what is their sum?

$\mathcal{O}(r)$ words of space, r : runs in the Burrows-Wheeler transform (BWT).

- matching statistics: Bannai,Gagie,I'20
- LPF: Prezza,Rosone'20
- NOV LZ by adaptation of Policriti,Prezza'18:
  - they update the RLBWT of the reversed text while computing a factor
  - instead, keep the RLBWT and update it *after* determining factor
  - adaptation to LPnF seems hard:
    - need to insert characters and undo these insertions in RLBWT

      $\Rightarrow \sum_{i=1}^{n} \text{LPnF}[i] = \mathcal{O}(n^2)$ steps
- for LPnF or LPnrF open