Bidirectional Search in a String with Wavelet Trees

Thomas Schnattinger

Institute of Theoretical Computer Science
University of Ulm, Germany

June 21, 2010

Problem description

Given: string S el_anele_lepanelen$

Find pattern: l e l _ane l e_ l epane l en$

"Forward search" for e el_ane le _ le pane le n$

"Backward search" for e el_an ele _lepan ele n$

Goal: "full text index" supporting *efficient bidirectional search*

Published approaches

Forward search

- Suffix tree (*Weiner P., 1973*)
- Suffix array (*Manber et al., 1990*)
- and several variants...

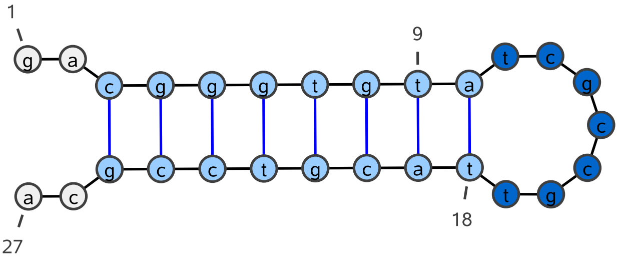
Backward search

- "FM-Index" (*Ferragina et al., 2000*)
- variant, using a "Wavelet Tree" (*Grossi et al., 2003*)

Bidirectional search

- Affix tree (*Stoye J., 1995*)
- Affix array (*Strothmann D., 2007*)

- Search for palindromic patterns in large strings
- e.g.: finding "RNA secondary structure" on the human genome

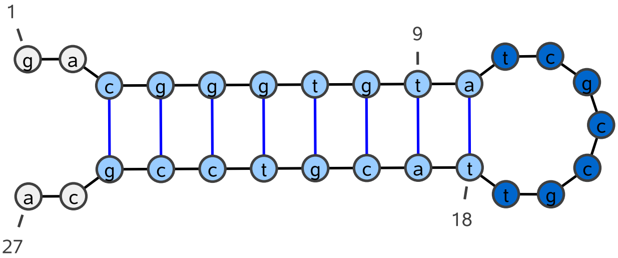


(possible base pairs are *a–t, c–g and g–t*)

. . . ggtgtatcgccgttacgtc . . . ggtgtatcgccgtaacgtc . . .

Motivation

- Search for palindromic patterns in large strings
- e.g.: finding "RNA secondary structure" on the human genome



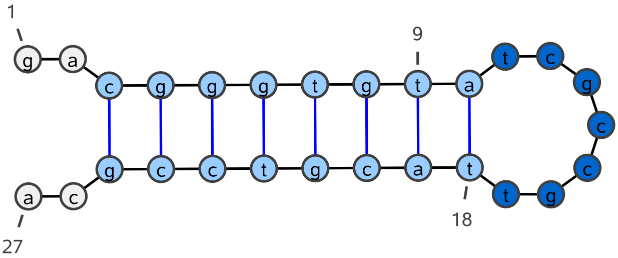
(possible base pairs are *a–t, c–g and g–t*)

`...ggtgtatcgccgttacgtc ... ggtgtatcgccgtaacgtc ...`

`...ggtgtatcgccgttacgtc ... ggtgtatcgccgtaacgtc ...`

Motivation

- Search for palindromic patterns in large strings
- e.g.: finding "RNA secondary structure" on the human genome



(possible base pairs are *a–t, c–g and g–t*)

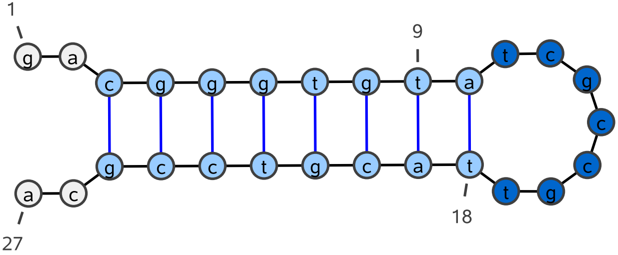
...ggtgtatcgccgttacgtc ... ggtgtatcgccgtaacgtc ...

...ggtgtatcgccgttacgtc ... ggtgtatcgccgtaacgtc ...

...ggtgtatcgccgttacgtc ... ggtgtatcgccgtaacgtc ...

Motivation

- Search for palindromic patterns in large strings
- e.g.: finding "RNA secondary structure" on the human genome



(possible base pairs are *a–t, c–g and g–t*)

. . . ggtgtatcgccgttacgtc . . . ggtgtatcgccgtaacgtc . . .

. . . ggtgtatcgccgttacgtc . . . ggtgtatcgccgtaacgtc . . .

. . . ggtgtatcgccgttacgtc . . . ggtgtatcgccgtaacgtc . . .

. . . ggtgtatcgccgttacgtc . . . ggtgtatcgccgtaacgtc . . .

Some basics

Given: string $S = \texttt{el\_anele\_lepanelen\$}$

$$\begin{aligned} S_1 &= \texttt{el\_anele\_lepanelen\$} \\ S_2 &= \texttt{l\_anele\_lepanelen\$} \\ S_3 &= \texttt{\_anele\_lepanelen\$} \\ &\vdots \\ S_{17} &= \texttt{en\$} \\ S_{18} &= \texttt{n\$} \\ S_{19} &= \$ \end{aligned}$$

Definition (Suffix Array)

The suffix array SA of a string S is an array containg a permutation of the numbers in the interval $[1..n]$ so that $S_{SA[1]} <_{\text{lex}} S_{SA[2]} <_{\text{lex}} \ldots <_{\text{lex}} S_{SA[n]}$.

Suffix Array

i	$SA[i]$	$S_{SA[i]}$
1	19	$
2	3	_anele_lepanelen$
3	9	_lepanelen$
4	4	anele_lepanelen$
5	13	anelen$
6	8	e_lepanelen$
7	1	el_anele_lepanelen$
8	6	ele_lepanelen$
9	15	elen$
10	17	en$
11	11	epanelen$
12	2	l_anele_lepanelen$
13	7	le_lepanelen$
14	16	len$
15	10	lepanelen$
16	18	n$
17	5	nele_lepanelen$
18	14	nelen$
19	12	panelen$

Lexicographical order
$\rightarrow$ ω-**interval**

Every substring ω of S corresponds to some interval

- e-interval [6..11]
- ele-interval [8..9]

Burrows and Wheeler transform

i	$SA[i]$	$BWT[i]$	$S_{SA[i]}$
1	19	n	$el_anele_lepanelen
2	3	l	_anele_lepanelen$el
3	9	e	_lepanelen$el_anele
4	4	_	anele_lepanelen$el_
5	13	p	anelen$el_anele_lep
6	8	l	e_lepanelen$el_anel
7	1	$	el_anele_lepanelen$
8	6	n	ele_lepanelen$el_an
9	15	n	elen$el_anele_lepan
10	17	l	en$el_anele_lepanel
11	11	l	epanelen$el_anele_l
12	2	e	l_anele_lepanelen$e
13	7	e	le_lepanelen$el_ane
14	16	e	len$el_anele_lepane
15	10	_	lepanelen$el_anele_
16	18	e	n$el_anele_lepanele
17	5	a	nele_lepanelen$el_a
18	14	a	nelen$el_anele_lepa
19	12	e	panelen$el_anele_le

Backward search for "ele" (1/3)

i	$BWT[i]$	$S_{SA[i]}$
1	n	$
2	l	_anele_lepanelen$
3	e	_lepanelen$
4	_	anele_lepanelen$
5	p	anelen$
6	l	e _lepanelen$
7	$	e l_anele_lepanelen$
8	n	e le_lepanelen$
9	n	e len$
10	l	e n$
11	l	e panelen$
12	e	l_anele_lepanelen$
13	e	le_lepanelen$
14	e	len$
15	_	lepanelen$
16	e	n$
17	a	nele_lepanelen$
18	a	nelen$
19	e	panelen$

Step 1. Determination of the e-interval

Backward search for “ele” (1/3)

i	$BWT[i]$	$S_{SA[i]}$
1	n	$
2	l	_anele_lepanelen$
3	e	_lepanelen$
4	_	anele_lepanelen$
5	p	anelen$
6	l	e_lepanelen$
7	$	el_anele_lepanelen$
8	n	ele_lepanelen$
9	n	elen$
10	l	en$
11	l	epanelen$
12	e	l_anele_lepanelen$
13	e	le_lepanelen$
14	e	len$
15	_	lepanelen$
16	e	n$
17	a	nele_lepanelen$
18	a	nelen$
19	e	panelen$

Step 1. Determination of the e-interval

trivial:

$i_1 = C[\mathtt{e}] = 6$
$j_1 = C[\mathtt{l}] - 1 = 11$

Backward search for “ele” (2/3)

i	$BWT[i]$	$S_{SA[i]}$
1	n	$
2	l	_anele_lepanelen$
3	e	_lepanelen$
4	_	anele_lepanelen$
5	p	anelen$
6	l	e_lepanelen$
7	$	el_anele_lepanelen$
8	n	ele_lepanelen$
9	n	elen$
10	l	en$
11	l	epanelen$
12	e	l_anele_lepanelen$
13	e	le _lepanelen$
14	e	le n$
15	_	le panelen$
16	e	n$
17	a	nele_lepanelen$
18	a	nelen$
19	e	panelen$

Step 2. Determination of the `le`-interval

(with backward search for `l`)

Backward search for “ele” (2/3)

i	$BWT[i]$	$S_{SA[i]}$
1	n	$
2	l	_anele_lepanelen$
3	e	_lepanelen$
4	_	anele_lepanelen$
5	p	anelen$
6	l	e _lepanelen$
7	$	e l_anele_lepanelen$
8	n	e le_lepanelen$
9	n	e len$
10	l	e n$
11	l	e panelen$
12	e	l_anele_lepanelen$
13	e	le _lepanelen$
14	e	le n$
15	_	le panelen$
16	e	n$
17	a	nele_lepanelen$
18	a	nelen$
19	e	panelen$

Step 2. Determination of the `le`-interval

(with backward search for `l`)

Backward search for “ele” (2/3)

i	$BWT[i]$	$S_{SA[i]}$
1	n	$
2	l	_anele_lepanelen$
3	e	_lepanelen$
4	_	anele_lepanelen$
5	p	anelen$
6	l	e_lepanelen$
7	$	el_anele_lepanelen$
8	n	ele_lepanelen$
9	n	elen$
10	l	en$
11	l	epanelen$
12	e	l_anele_lepanelen$
13	e	le_lepanelen$
14	e	len$
15	_	lepanelen$
16	e	n$
17	a	nele_lepanelen$
18	a	nelen$
19	e	panelen$

Step 2. Determination of the `le`-interval

(with backward search for `l`)

$$i_2 = C[\mathtt{l}] + Occ(\mathtt{l}, i_1 - 1) = 12 + 1 = 13$$

$$j_2 = C[\mathtt{l}] + Occ(\mathtt{l}, j_1) - 1 = 12 + 4 - 1 = 15$$

Backward search for "ele" (3/3)

i	$BWT[i]$	$S_{SA[i]}$
1	n	$
2	l	_anele_lepanelen$
3	e	_lepanelen$
4	_	anele_lepanelen$
5	p	anelen$
6	l	e_lepanelen$
7	$	el_anele_lepanelen$
8	n	ele _lepanelen$
9	n	ele n$
10	l	en$
11	l	epanelen$
12	e	l_anele_lepanelen$
13	e	le _lepanelen$
14	e	le n$
15	_	le panelen$
16	e	n$
17	a	nele_lepanelen$
18	a	nelen$
19	e	panelen$

Step 3. Determination of the ele-interval

(with backward search for e)

Backward search for “ele” (3/3)

i	$BWT[i]$	$S_{SA[i]}$
1	n	$
2	l	_anele_lepanelen$
3	e	_lepanelen$
4	_	anele_lepanelen$
5	p	anelen$
6	l	e_lepanelen$
7	$	el_anele_lepanelen$
8	n	ele _lepanelen$
9	n	ele n$
10	l	en$
11	l	epanelen$
12	e	l_anele_lepanelen$
13	e	le _lepanelen$
14	e	le n$
15	_	le panelen$
16	e	n$
17	a	nele_lepanelen$
18	a	nelen$
19	e	panelen$

Step 3. Determination of the ele-interval (with backward search for e)

Backward search for “ele” (3/3)

i	$BWT[i]$	$S_{SA[i]}$
1	n	$
2	l	_anele_lepanelen$
3	e	_lepanelen$
4	_	anele_lepanelen$
5	p	anelen$
6	l	e_lepanelen$
7	$	el_anele_lepanelen$
8	n	ele_lepanelen$
9	n	elen$
10	l	en$
11	l	epanelen$
12	e	l_anele_lepanelen$
13	e	le_lepanelen$
14	e	len$
15	_	lepanelen$
16	e	n$
17	a	nele_lepanelen$
18	a	nelen$
19	e	panelen$

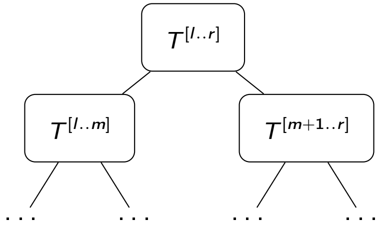
Step 3. Determination of the ele-interval (with backward search for e)

$$i_3 = C[e] + Occ(e, i_2 - 1) = 6 + 2 = 8$$

$$j_3 = C[e] + Occ(e, j_2) - 1 = 6 + 4 - 1 = 9$$

Definition: Wavelet Tree

- Every node corresponds to some alphabet interval $[l..r]$

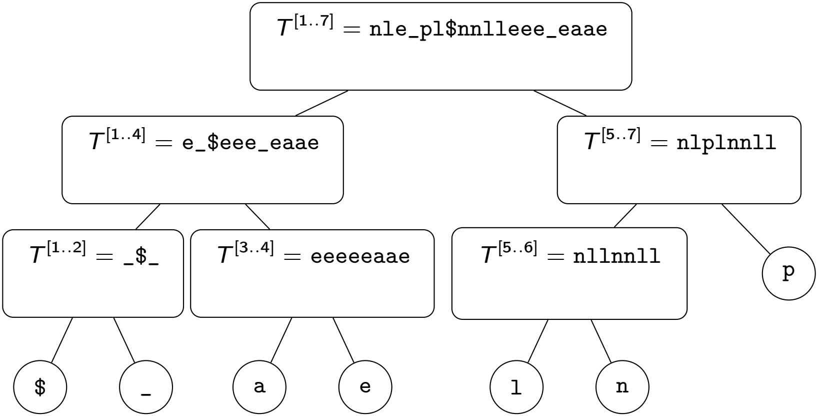


where $m = \lfloor \frac{l+r}{2} \rfloor$.

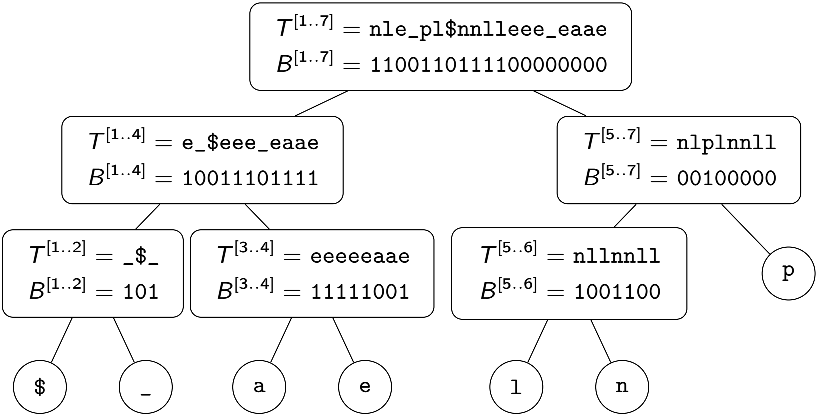
- The root corresponds to the whole alphabet interval $[1..|\Sigma|]$.

- $T^{[l..r]}$ is formed from T by deleting characters that are not in the alphabet interval $[l..r]$.

Wavelet Tree of $T = BWT(S) =$ nle_pl$nnlleee_eaae

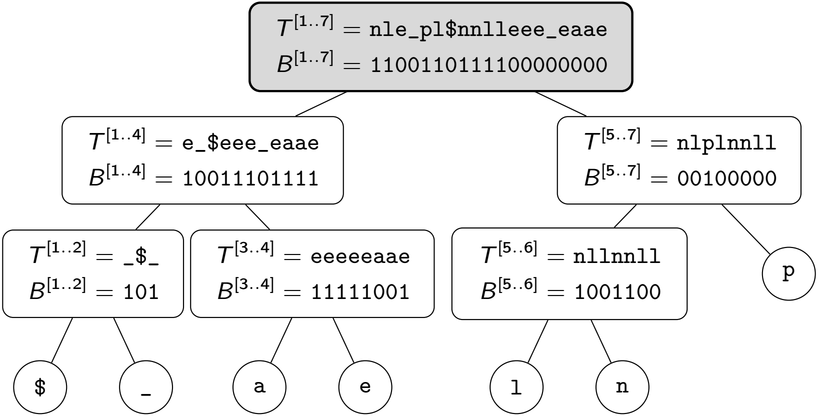


Wavelet Tree of $T = BWT(S) = \texttt{nle\_pl\$nnlleee\_eaae}$



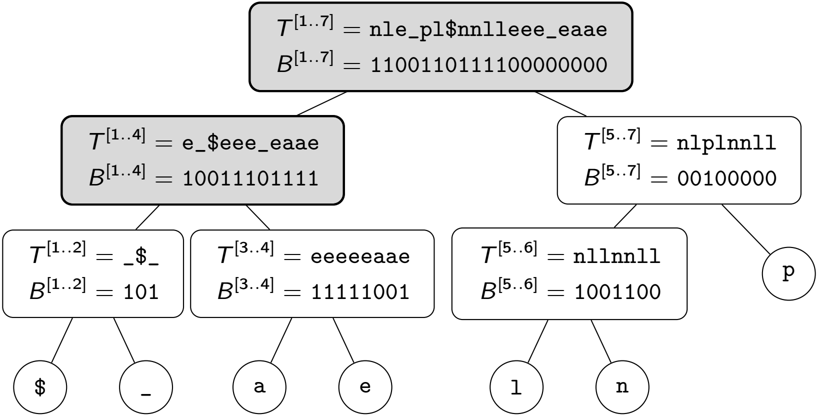
$Occ(e, 16) =$

Wavelet Tree of $T = BWT(S) =$ `nle_pl$nnlleee_eaae`



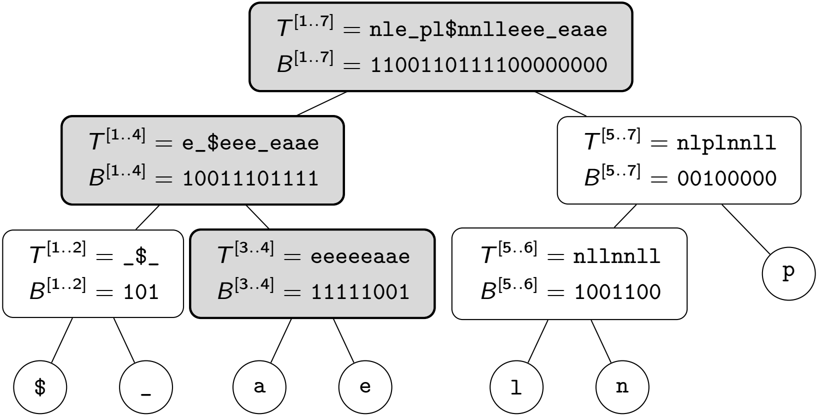
$Occ(e, 16) = Occ'(e, 16, [1..7]) =$

Wavelet Tree of $T = BWT(S) =$ nle_pl$nnlleee_eaae



$$Occ(e, 16) = Occ'(e, 16, [1..7]) = Occ'(e, \underbrace{\text{rank}_0(B^{[1..7]}, 16)}_{=8}, [1..4]) =$$

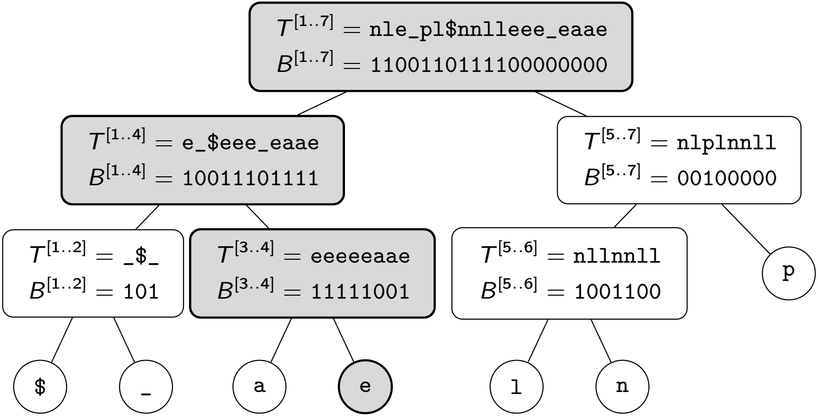
Wavelet Tree of $T = BWT(S) =$ nle_pl$nnlleee_eaae



$$Occ(e, 16) = Occ'(e, 16, [1..7]) = Occ'(e, \underbrace{\text{rank}_0(B^{[1..7]}, 16)}_{=8}, [1..4]) =$$

$$Occ'(e, \underbrace{\text{rank}_1(B^{[1..4]}, 8)}_{=5}, [3..4]) =$$

Wavelet Tree of $T = BWT(S) =$ `nle_pl$nnlleee_eaae`



$$Occ(e,16) = Occ'(e,16,[1..7]) = Occ'(e,\underbrace{\mathrm{rank}_0(B^{[1..7]},16)}_{=8},[1..4]) =$$

$$Occ'(e,\underbrace{\mathrm{rank}_1(B^{[1..4]},8)}_{=5},[3..4]) = Occ'(e,\underbrace{\mathrm{rank}_1(B^{[3..4]},5)}_{=5},[4..4]) = 5$$

Bidirectional search

Idea:

- one wavelet tree for the string $BWT(S)$ for backward search, and
- one wavelet tree for the reverse string $BWT(S^{rev})$ for forward search (by backward searching the reverse text)

Problem:

- There is no efficient mapping of the intervals of these two indexes.
- So we always have to store both intervals; one search step in one index requires adjustment of the other interval.

The bidirectional wavelet-index

i		$S_{SA[i]}$
1	n	$
2	l	_anele_lepanelen$
3	e	_lepanelen$
4	_	anele_lepanelen$
5	p	anelen$
6	l	e _lepanelen$
7	$	e l_anele_lepanelen$
8	n	e le_lepanelen$
9	n	e len$
10	l	e n$
11	l	e panelen$
12	e	l_anele_lepanelen$
13	e	le _lepanelen$
14	e	le n$
15	_	le panelen$
16	e	n$
17	a	nele_lepanelen$
18	a	nelen$
19	e	panelen$

i		$S^{rev}_{SA^{rev}[i]}$
1	e	$
2	l	_elena_le$
3	a	_le$
4	n	a_le$
5	n	apel_elena_le$
6	l	e $
7	p	e l _elena_le$
8	_	e l ena_le$
9	n	e l enapel_elena_le$
10	l	e n a_le$
11	l	e n apel_elena_le$
12	e	l_elena_le$
13	_	le$
14	e	lena_le$
15	e	lenapel_elena_le$
16	e	na_le$
17	e	napel_elena_le$
18	$	nelenapel_elena_le$
19	a	pel_elena_le$

The bidirectional wavelet-index

i		$S_{SA[i]}$
1	n	$
2	l	_anele_lepanelen$
3	e	_lepanelen$
4	_	anele_lepanelen$
5	p	anelen$
6	l	e _lepanelen$
7	$	e l_anele_lepanelen$
8	n	e le_lepanelen$
9	n	e len$
10	l	e n$
11	l	e panelen$
12	e	l_anele_lepanelen$
13	e	le _lepanelen$
14	e	le n$
15	_	le panelen$
16	e	n$
17	a	nele_lepanelen$
18	a	nelen$
19	e	panelen$

i		$S^{rev}_{SA^{rev}[i]}$
1	e	$
2	l	_elena_le$
3	a	_le$
4	n	a_le$
5	n	apel_elena_le$
6	l	e $
7	p	e l _elena_le$
8	_	e l ena_le$
9	n	e l enapel_elena_le$
10	l	e n a_le$
11	l	e n apel_elena_le$
12	e	l_elena_le$
13	_	le$
14	e	lena_le$
15	e	lenapel_elena_le$
16	e	na_le$
17	e	napel_elena_le$
18	$	nelenapel_elena_le$
19	a	pel_elena_le$

The bidirectional wavelet-index

i		$S_{SA[i]}$
1	n	$
2	l	_anele_lepanelen$
3	e	_lepanelen$
4	_	anele_lepanelen$
5	p	anelen$
6	l	e_lepanelen$
7	$	el_anele_lepanelen$
8	n	ele_lepanelen$
9	n	elen$
10	l	en$
11	l	epanelen$
12	e	l_anele_lepanelen$
13	e	le_lepanelen$
14	e	len$
15	_	lepanelen$
16	e	n$
17	a	nele_lepanelen$
18	a	nelen$
19	e	panelen$

i		$S^{rev}_{SA^{rev}[i]}$
1	e	$
2	l	_elena_le$
3	a	_le$
4	n	a_le$
5	n	apel_elena_le$
6	l	e$
7	p	el_elena_le$
8	_	elena_le$
9	n	elenapel_elena_le$
10	l	ena_le$
11	l	enapel_elena_le$
12	e	l_elena_le$
13	_	le$
14	e	lena_le$
15	e	lenapel_elena_le$
16	e	na_le$
17	e	napel_elena_le$
18	$	nelenapel_elena_le$
19	a	pel_elena_le$

The bidirectional wavelet-index

i		$S_{SA[i]}$
1	n	$
2	l	_anele_lepanelen$
3	e	_lepanelen$
4	_	anele_lepanelen$
5	p	anelen$
6	l	e_lepanelen$
7	$	el_anele_lepanelen$
8	n	ele_lepanelen$
9	n	elen$
10	l	en$
11	l	epanelen$
12	e	l_anele_lepanelen$
13	e	le_lepanelen$
14	e	len$
15	_	lepanelen$
16	e	n$
17	a	nele_lepanelen$
18	a	nelen$
19	e	panelen$

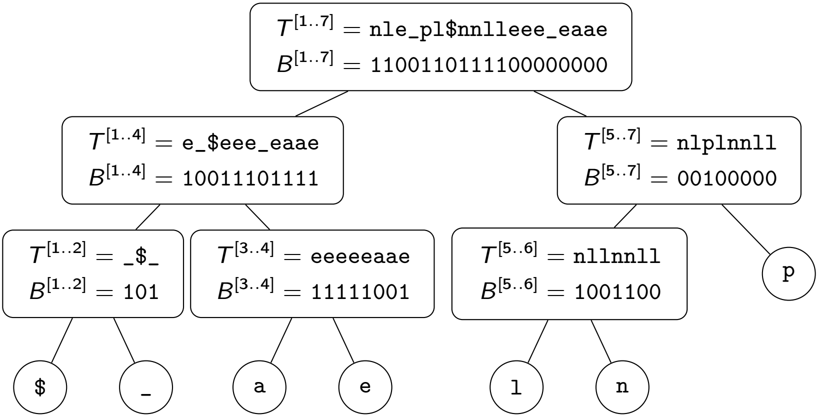
i		$S^{rev}_{SA^{rev}[i]}$
1	e	$
2	l	_elena_le$
3	a	_le$
4	n	a_le$
5	n	apel_elena_le$
6	l	e$
7	p	el_elena_le$
8	_	elena_le$
9	n	elenapel_elena_le$
10	l	ena_le$
11	l	enapel_elena_le$
12	e	l_elena_le$
13	_	le$
14	e	lena_le$
15	e	lenapel_elena_le$
16	e	na_le$
17	e	napel_elena_le$
18	$	nelenapel_elena_le$
19	a	pel_elena_le$

The bidirectional wavelet-index

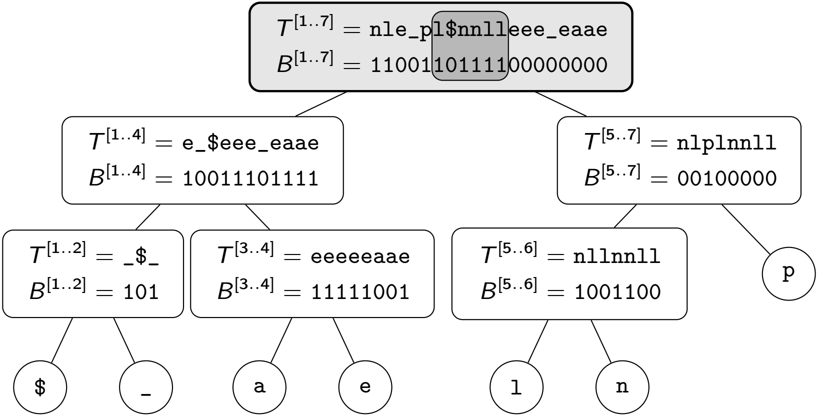
i		$S_{SA[i]}$
1	n	$
2	l	_anele_lepanelen$
3	e	_lepanelen$
4	_	anele_lepanelen$
5	p	anelen$
6	l	e_lepanelen$
7	$	el_anele_lepanelen$
8	n	ele_lepanelen$
9	n	elen$
10	l	en$
11	l	epanelen$
12	e	l_anele_lepanelen$
13	e	le_lepanelen$
14	e	len$
15	_	lepanelen$
16	e	n$
17	a	nele_lepanelen$
18	a	nelen$
19	e	panelen$

i		$S^{rev}_{SA^{rev}[i]}$
1	e	$
2	l	_elena_le$
3	a	_le$
4	n	a_le$
5	n	apel_elena_le$
6	l	e$
7	p	el_elena_le$
8	_	elena_le$
9	n	elenapel_elena_le$
10	l	ena_le$
11	l	enapel_elena_le$
12	e	l_elena_le$
13	_	le$
14	e	lena_le$
15	e	lenapel_elena_le$
16	e	na_le$
17	e	napel_elena_le$
18	$	nelenapel_elena_le$
19	a	pel_elena_le$

The function *getBounds*

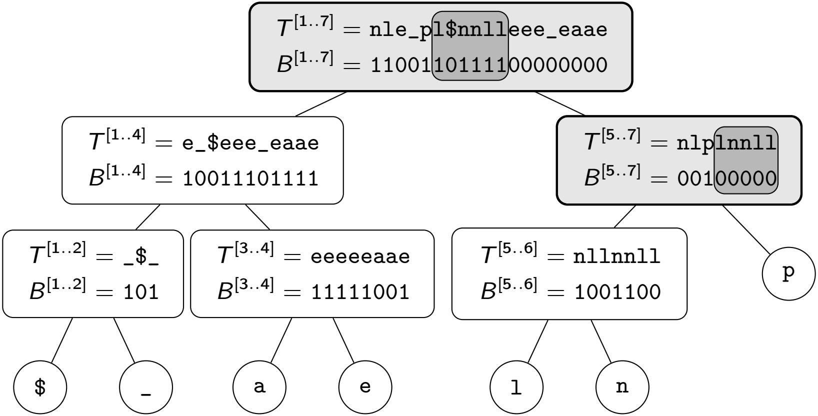


The function *getBounds*



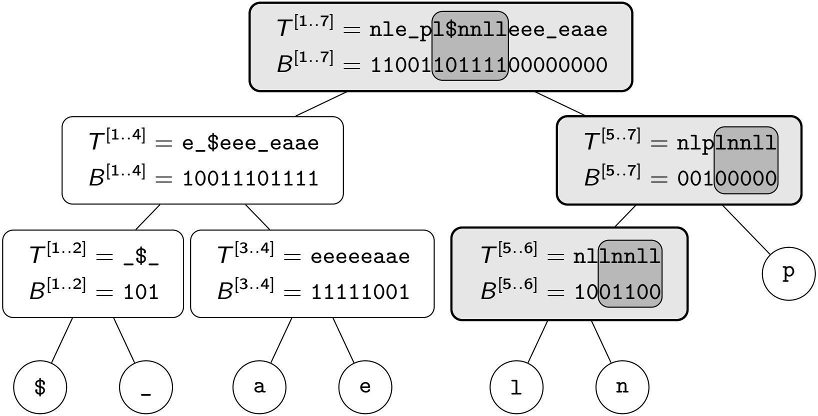
1. *getBounds*([6..11], [1..7], l) → 1 character smaller than l.

The function *getBounds*



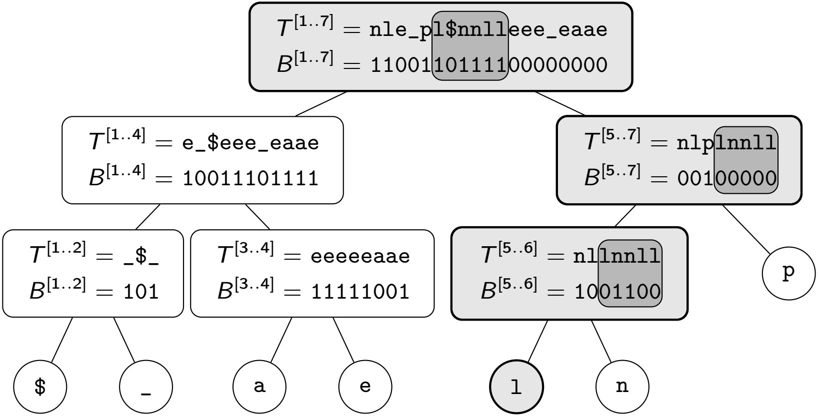
1. *getBounds*([6..11], [1..7], l) → 1 character smaller than l.
2. *getBounds*([4..8], [5..7], l)

The function *getBounds*



1. *getBounds*$([6..11], [1..7], \texttt{l})$ $\rightarrow$ 1 character smaller than l.
2. *getBounds*$([4..8], [5..7], \texttt{l})$
3. *getBounds*$([3..7], [5..6], \texttt{l})$ $\rightarrow$ 2 characters greater than l.

The function *getBounds*



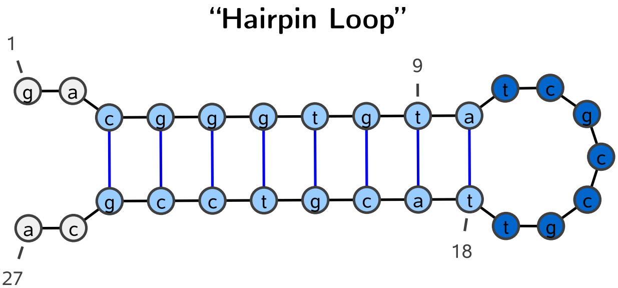
1. *getBounds*([6..11], [1..7], l) → 1 character smaller than l.
2. *getBounds*([4..8], [5..7], l)
3. *getBounds*([3..7], [5..6], l) → 2 characters greater than l.
4. *getBounds*([2..4], [5..5], l)

The bidirectional wavelet-index

i		$S_{SA[i]}$
1	n	$
2	l	_anele_lepanelen$
3	e	_lepanelen$
4	_	anele_lepanelen$
5	p	anelen$
6	l	e_lepanelen$
7	$	el_anele_lepanelen$
8	n	ele_lepanelen$
9	n	elen$
10	l	en$
11	l	epanelen$
12	e	l_anele_lepanelen$
13	e	le_lepanelen$
14	e	len$
15	_	lepanelen$
16	e	n$
17	a	nele_lepanelen$
18	a	nelen$
19	e	panelen$

i		$S^{rev}_{SA^{rev}[i]}$
1	e	$
2	l	_elena_le$
3	a	_le$
4	n	a_le$
5	n	apel_elena_le$
6	l	e$
7	p	el_elena_le$
8	_	elena_le$
9	n	elenapel_elena_le$
10	l	ena_le$
11	l	enapel_elena_le$
12	e	l_elena_le$
13	_	le$
14	e	lena_le$
15	e	lenapel_elena_le$
16	e	na_le$
17	e	napel_elena_le$
18	$	nelenapel_elena_le$
19	a	pel_elena_le$

Experimental comparison (1/2)



(possible base pairs are *a–t*, *c–g* and *g–t*)

Patterns:

- `P1 = (stem:=N{10,50}) (loop:=GGAC) ^stem`
- `P2 = (stem:=N{15,20}) (loop:=N{5}) ^stem`
- `P3 = (stem:=N{15,20}) (loop:=(A|C){15}) ^stem`

Experimental comparison (2/2)

Text:
the first 1 billion characters of the human genome

Results:

index	size	P1	P2	P3
SA, *Occ* + text	4408 MB	41 ms	22208 ms	249 ms
CSA/*Occ* + text	1053 MB	651 ms	345860 ms	6528 ms
BWI	799 MB	79 ms	28373 ms	274 ms

Compare: affix tree (42915 MB), affix array (17166 MB)

End

Thank you for your attention.

Any questions?

Another Comparison

index	**size**	P1	P2	P3
SA, *Occ* + text	4408 MB	< 1s	22s	< 1s
CSA/*Occ* + text	1053 MB	< 1s	5m 46s	7s
BWI	799 MB	< 1s	28s	< 1s
Affix tree	42915 MB	-	-	-
Affix array	17166 MB	-	-	-
RNAmotif	0 MB	9m 34s	15m 59s	1m 16s