Tutorial Week 7

Definition 1. *The suffix table* SUF *of a string y of length n gives us the order of the suffixes of y, such that* $\mathrm{SUF}[i] < \mathrm{SUF}[i+1]$. *The* LCP *array corresponding to y gives us the lengths of the longest common prefixes of consecutive elements in the suffix table:*

$$\mathrm{LCP}[i] = |\mathrm{lcp}(y[\mathrm{SUF}[i-1]..n-1], y[\mathrm{SUF}[i]..n-1])|.$$

Together the two form the suffix array. If by $\mathrm{First}_k(u)$ *we denote the prefix of length k of u, when $|u| \geq k$, and u, otherwise, then by $R_k[i]$ we denote the rank of* $\mathrm{First}_k(y[i..n-1])$ *inside the ordered list of all* $\mathrm{First}_k(u)$, *where u is a non-empty suffix of y.*

Exercise 1. *Consider the following list of sequences: ababa, abcacabb, abcacababc, and abacabacab. For each of them, using the prefix doubling technique find out the corresponding ranks. Next, find out their corresponding suffix array consisting of the suffix table* SUF *and the corresponding* LCP *table, respectively.*

What is the complexity of the algorithm?

Solution:

i/r	0	1	2	3	4
$y[i]$	a	b	a	b	a
$R_1[i]$	0	1	0	1	0
$R_2[i]$	1	2	1	2	0
$R_4[i]$	2	4	1	3	0
$\mathrm{SUF}[r]$	4	2	0	3	1
$\mathrm{LCP}[r]$	0	1	3	0	2

i/r	0	1	2	3	4	5	6	7
$y[i]$	a	b	c	a	c	a	b	b
$R_1[i]$	0	1	2	0	2	0	1	1
$R_2[i]$	0	4	5	1	5	0	3	2
$R_4[i]$	1	5	7	2	6	0	4	3
$\mathrm{SUF}[r]$	5	0	3	7	6	1	4	2
$\mathrm{LCP}[r]$	0	2	1	0	1	1	0	2

i/r	0	1	2	3	4	5	6	7	8	9
$y[i]$	a	b	c	a	c	a	b	a	b	c
$R_1[i]$	0	1	2	0	2	0	1	0	1	2
$R_2[i]$	0	3	5	1	5	0	2	0	3	4
$R_4[i]$	2	6	9	3	8	0	4	1	5	7
SUF$[r]$	5	7	0	3	6	8	1	9	4	2
LCP$[r]$	0	2	3	1	0	1	2	0	1	2

i/r	0	1	2	3	4	5	6	7	8	9
$y[i]$	a	b	a	c	a	b	a	c	a	b
$R_1[i]$	0	1	0	2	0	1	0	2	0	1
$R_2[i]$	0	3	1	4	0	3	1	4	0	2
$R_4[i]$	1	4	2	6	1	4	2	5	0	3
$R_8[i]$	2	7	4	9	1	6	3	8	0	5
SUF$[r]$	8	4	0	6	2	9	5	1	7	3
LCP$[r]$	0	2	6	1	4	0	1	5	0	3

The algorithm takes linear time to find the array SUF using the skew algorithm and linear time to compute the array LCP. Please note that when computing the SUF$[r]$ we look at the positions of the ranks in the word! Furthermore, when computing the LCP$[r]$ we look at the $r-1$ and r positions in the array SUF and not in the word! ■

Exercise 2. *Consider now the same list of sequences: ababa, abcacabb, abcacababc, and abacabacab. For each of them, using the skew algorithm find out the corresponding ranks, considering the t and s sequences. Next, find out their corresponding suffix array consisting of the suffix table* SUF *and the corresponding* LCP *table (obviously, these are the same as for the previous exercise).*

What is the complexity of the algorithm?

Solution: First we look at the string *ababa*.

i	0	1	2	3	4
$y[i]$	a	b	a	b	a

Rank t		Rank s	i	Suff(1230)
0	a	0	4	0
1	aba	1	0	1230
2	ba	2	3	230
3	bab	3	1	30

Rank	j	$(y[j], s[j+1])$
0	2	$(a, 2)$

i	0	1	2	3	4
$y[i]$	a	b	a	b	a
$r[i]$	2	4	1	3	0
SUF$[r]$	4	2	0	3	1
LCP$[r]$	0	1	3	0	2

Next we look at the string $abcacabb$.

i	0	1	2	3	4	5	6	7
$y[i]$	a	b	c	a	c	a	b	b

Rank t		Rank s	i	Suff(013452)
0	abc	0	0	013452
1	aca	1	3	13452
2	b	2	7	2
3	bb	3	6	3452
4	bca	4	1	452
5	cab	5	4	52

Rank	j	$(y[j], s[j+1])$
0	2	$(a, 3)$
1	5	$(c, 1)$

i	0	1	2	3	4	5	6	7
$y[i]$	a	b	c	a	c	a	b	b
$r[i]$	1	5	7	2	6	0	4	3
SUF$[r]$	5	0	3	7	6	1	4	2
LCP$[r]$	0	2	1	0	1	1	0	2

Next we look at the string $abcacababc$.

i	0	1	2	3	4	5	6	7	8	9
$y[i]$	a	b	c	a	c	a	b	a	b	c

Rank t		Rank s	i	Suff(0124350)	Rank	j	$(y[j], s[j+1])$
0	abc	0	7	0	0	5	$(a, 2)$
1	aca	1	0	0124350	1	8	$(b, 4)$
2	bab	2	3	124350	2	2	$(c, 1)$
3	bca	3	6	24350			
4	c	4	1	350			
5	cab	5	9	4350			
		6	4	50			

i	0	1	2	3	4	5	6	7	8	9
$y[i]$	a	b	c	a	c	a	b	a	b	c
$r[i]$	2	6	9	3	8	0	4	1	5	7
SUF$[r]$	5	7	0	3	6	8	1	9	4	2
LCP$[r]$	0	2	3	1	0	1	2	0	1	2

Finally, we look at the string *abacabacab*.

i	0	1	2	3	4	5	6	7	8	9
$y[i]$	a	b	a	c	a	b	a	c	a	b

Rank t		Rank s	i	Suff(0412304)	Rank	j	$(y[j], s[j+1])$
0	aba	0	4	04	0	8	$(a, 2)$
1	aca	1	0	0412304	1	2	$(a, 4)$
2	b	2	6	12304	2	5	$(b, 1)$
3	bac	3	9	2304			
4	cab	4	1	304			
		5	7	4			
		5	3	412304			

i	0	1	2	3	4	5	6	7	8	9
$y[i]$	a	b	a	c	a	b	a	c	a	b
$r[i]$	2	7	4	9	1	6	3	8	0	5
SUF$[r]$	8	4	0	6	2	9	5	1	7	3
LCP$[r]$	0	2	6	1	4	0	1	5	0	3

The algorithm takes linear time to find the array SUF using the skew algorithm and linear time to compute the array LCP. Please note that when computing the SUF$[r]$ we look at the positions of the ranks in the word! Furthermore, when computing the LCP$[r]$ we look at the $r-1$ and r positions in the array SUF and not in the word! ■