Genomics

Freie Universität Berlin, Institut für Informatik
Knut Reinert, Peter Robinson, Sebastian Bauer
Wintersemester 2012/2013

5. Übungsblatt (**mit Lösung**) vom 24. November 2012
Diskussion am 29. November 2012

---

1. Name two applications of the Burrows-Wheeler transform. What ist the actual benefit of the Burrows-Wheeler transform in these application (in comparision to other methods)?

   Solution See script.

2. Given the Burrows-Wheeler transform $L =$ `ammnnb$aaaa`. (without dot)

   1. Decode the original text.
   2. Formulate an algorithm that efficiently counts the number of occurrences of a pattern in the original text (without decoding the original text). Describe all of the used data structures.
   3. Illustrate how your algorithm works by searching the pattern $P =$ `ana`.

   Solution

   (a) We begin with the last character `$` and second last character `a` in the first rows of F and L. We iteratively determine preceeding characters using the L-to-F mapping and get the original text T=`bananamama$`.

$$\begin{aligned} LF(i) &= C(L[i]) + Occ(L[i], i) \\ T[n-i] &= F[LF^i(1)] \end{aligned}$$

   with LF^i denoting the i-th function composition with itself.

$a \in \Sigma$	`$`	`a`	`b`	`m`	`n`
n_a	1	5	1	2	2
$C(a)$	0	1	6	7	9

i	$F[i]$	$L[i]$	$Occ(L[i], i)$
1	`$`	`a`	1
2	`a`	`m`	1
3	`a`	`m`	2
4	`a`	`n`	1
5	`a`	`n`	2
6	`a`	`b`	1
7	`b`	`$`	1
8	`m`	`a`	2
9	`m`	`a`	3
10	`n`	`a`	4
11	`n`	`a`	5

   (b) Algorithm **count** computes the number of occurrences of $P[1..m]$ in $T[1..n]$:

   // **count(P[1..m])**
   $i = m, a = 1, b = n;$
   while $((a < b) \ \wedge \ (i \geq 1))$ do
   $\quad c = P[i];$
   $\quad a = C(c) + \mathsf{Occ}(c, a-1) + 1;$
   $\quad b = C(c) + \mathsf{Occ}(c, b);$
   $\quad i = i - 1;$
   od
   if $(b < a)$ then return "not found";
   $\quad$ else return "found $(b - a + 1)$ occurrences";
   fi

(c)

α	\$	a	b	m	n
$C[\alpha]$	0	1	6	7	9

- Start with the whole interval $(a_4, b_4) = (1, 11)$.
- Compute L-to-F mappings of the first and last occurrence of a in L $\rightarrow$ $(a_3, b_3) = (2, 6)$.
- Compute L-to-F mappings of the first and last occurrence of n in $L[2..6]$ $\rightarrow$ $(a_2, b_2) = (10, 11)$
- Compute L-to-F mappings of the first and last occurrence of a in $L[10..11]$ $\rightarrow$ $(a_1, b_1) = (5, 6)$

3. For the text **tacaacaatacaagag** construct the BWT and the arrays C and Occ. Use them to search for the pattern **aca**.

$a \in \Sigma$	\$	a	c	g	t
n_a	1	9	3	2	2
$C(a)$	0	1	10	13	15

i	$F[i]$																$L[i]$	$Occ(L[i], i)$	$A[i]$	Marked
1	\$	t	a	c	a	a	c	a	a	t	a	c	a	a	g	a	g	1	17	*
2	a	a	c	a	a	t	a	c	a	a	g	a	g	\$	t	a	c	1	4	
3	a	a	g	a	g	\$	t	a	c	a	a	c	a	a	t	a	c	2	12	
4	a	a	t	a	c	a	a	g	a	g	\$	t	a	c	a	a	c	3	7	
5	a	c	a	a	c	a	a	t	a	c	a	a	g	a	g	\$	t	1	2	
6	a	c	a	a	g	a	g	\$	t	a	c	a	a	c	a	a	t	2	10	
7	a	c	a	a	t	a	c	a	a	g	a	g	\$	t	a	c	a	1	5	*
8	a	g	\$	t	a	c	a	a	c	a	a	t	a	c	a	a	g	2	15	
9	a	g	a	g	\$	t	a	c	a	a	c	a	a	t	a	c	a	2	13	*
10	a	t	a	c	a	a	g	a	g	\$	t	a	c	a	a	c	a	3	8	
11	c	a	a	c	a	a	t	a	c	a	a	g	a	g	\$	t	a	4	3	
12	c	a	a	g	a	g	\$	t	a	c	a	a	c	a	a	t	a	5	11	
13	c	a	a	t	a	c	a	a	g	a	g	\$	t	a	c	a	a	6	6	
14	g	\$	t	a	c	a	a	c	a	a	t	a	c	a	a	g	a	7	16	
15	g	a	g	\$	t	a	c	a	a	c	a	a	t	a	c	a	a	8	14	
16	t	a	c	a	a	c	a	a	t	a	c	a	a	g	a	g	\$	1	1	*
17	t	a	c	a	a	g	a	g	\$	t	a	c	a	a	c	a	a	9	9	*

First, we count the number of occurrences

- Start with the whole interval $(a_4, b_4) = (1, 17)$.
- Compute L-to-F mappings of the first and last occurrence of a in L $\rightarrow$ $(a_3, b_3) = (2, 10)$.
- Compute L-to-F mappings of the first and last occurrence of c in $L[2 \ldots 10]$ $\rightarrow$ $(a_2, b_2) = (11, 13)$
- Compute L-to-F mappings of the first and last occurrence of a in $L[11 \ldots 13]$ $\rightarrow$ $(a_1, b_1) = (5, 7)$

Second, we decode the text starting at each of the interval's rows (5,6, and 7 here). If we decode the text until the beginning, we can conclude the location of the pattern from the number of decoded characters. This, however, would mean in average a running time proportional to the length of the text. For instance, to determine the location of the match defined by row 6, we would need to apply LF 9 times.

If, on the other hand, we would know the position of each character in the text, there would be nothing left to do. Note that the suffix array of the text gives exactly this (denoted as $A[i]$ in the table). Thus, as row 6 is the 6th lexicographical smallest suffix of the text, we can conclude from $A[6] = 10$ that the pattern must match at position 10.

However, storing the entire suffix array takes additional space. In practice, only a sampled suffix array is stored, meaning that we have to decode the text (i.e., we have to apply LF) until we reach a row with known position. Thus, to achive an $O(m)$[1] running time of the entire match procedure, we need to have access to the suffix array sampled using an $O(1)$ interval.

[1] Recall that m is the length of the pattern.

In the table, we specify, which position are known via an asterisk symbol (every 4th position was sampled). Thus, as the $A[6]$ is unknown, we apply LF, which brings us to row 17. As we know the position of that suffix represented by this row and we applied LF only once, the position of our match must be $A[17] + 1 = 9 + 1 = 10$.