

# A note on the Burrows-Wheeler transformation

Maxime Crochemore      Jacques Désarménien  
Dominique Perrin

July 25, 2003

## Abstract

We relate the Burrows-Wheeler transformation with a result in combinatorics on words known as the Gessel-Reutenauer transformation.

## 1 Introduction

The Burrows-Wheeler transformation is a popular method used for text compression [3]. The rough idea is to encode a text in two passes. In the first pass, the text  $w$  is replaced by a text  $T(w)$  of the same length obtained as follows: list the cyclic shifts of  $w$  in alphabetic order as the rows  $w_1, w_2, \dots, w_n$  of an array. Then  $T(w)$  is the last column of the array. In a second pass, a simple encoding allows to compress  $T(w)$ , using a simple method like run-length or move-to-front encoding. Indeed, adjacent rows will often begin by a long common prefix and  $T(w)$  will therefore have long runs of identical symbols. For example, in a text in english, most rows beginning with 'nd' will end with 'a'. We refer to [9] for a complete presentation of the algorithm and an analysis of its performances. It was remarked recently by S. Mantacci, A. Restivo and M. Sciortino [8] that this transformation was related with notions in combinatorics on words such as Sturmian words. Similar considerations were developed in [2] in a different context. The results presented here are also close to the ones of [5].

In this note, we study the transformation from the combinatorial point of view. We show that the Burrows-Wheeler transformation is a particular case of a bijection due to I.M. Gessel and C. Reutenauer which allows the enumeration of permutations by descents and cyclic type (see [7]).

The paper is organized as follows. In the first section, we describe the Burrows-Wheeler transformation. The next section describes the inverse of the transformation with some emphasis on the computational aspects. The last section is devoted to the link with the Gessel-Reutenauer correspondance.

## 2 The Burrows-Wheeler transformation

The principle of the method is very simple. We consider an ordered alphabet  $A$ . Let  $w = a_1 a_2 \dots a_n$  be a word of length  $n$  on the alphabet  $A$ . We suppose  $w$

to be primitive, i.e. that  $w$  is not a power of another word. Let  $w_1, w_2, \dots, w_n$  be the sequence of conjugates of  $w$  in increasing alphabetic order. Let  $b_i$  denote the last letter of  $w_i$ , for  $i = 1, \dots, n$ . Then the Burrows-Wheeler transform of  $w$  is the word  $T(w) = b_1 b_2 \dots b_n$ .

**Example 1** Let  $w = \text{abracadabra}$ . The list of conjugates of  $w$  sorted in alphabetical order is represented below.

	1	2	3	4	5	6	7	8	9	10	11
1	a	a	b	r	a	c	a	d	a	b	r
2	a	b	r	a	a	b	r	a	c	a	d
3	a	b	r	a	c	a	d	a	b	r	a
4	a	c	a	d	a	b	r	a	a	b	r
5	a	d	a	b	r	a	a	b	r	a	c
6	b	r	a	a	b	r	a	c	a	d	a
7	b	r	a	c	a	d	a	b	r	a	a
8	c	a	d	a	b	r	a	a	b	r	a
9	d	a	b	r	a	a	b	r	a	c	a
10	r	a	a	b	r	a	c	a	d	a	b
11	r	a	c	a	d	a	b	r	a	a	b

The word  $T(w)$  is the last column of the array. Thus  $T(w) = \text{rdarcaaabb}$ .

It is clear that  $T(w)$  depends only on the conjugacy class of  $w$ . Therefore, in order to study the correspondance  $w \mapsto T(w)$ , we may suppose that  $w$  is a Lyndon word, i.e. that  $w = w_1$ . Let  $c_i$  denote the first letter of  $w_i$ . Thus the word  $z = c_1 c_2 \dots c_n$  is the nondecreasing rearrangement of  $w$  (and of  $T(w)$ ).

Let  $\sigma$  be the permutation of the set  $\{1, \dots, n\}$  such that  $\sigma(i) = j$  iff  $w_j = a_i a_{i+1} \dots a_{i-1}$ . In other terms,  $\sigma(i)$  is the rank in the alphabetic order of the  $i$ -th circular shift of the word  $w$ .

**Example 1** (continued)

We have

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 1 & 3 & 7 & 11 & 4 & 8 & 5 & 9 & 2 & 6 & 10 \end{pmatrix}$$

By definition, we have for each index  $i$  with  $1 \leq i \leq n$

$$a_i = c_{\sigma(i)}. \tag{1}$$

We also have the following formula expressing  $T(w)$  using  $\sigma$

$$b_i = a_{\sigma^{-1}(i)-1} \tag{2}$$

Indeed,  $b_{\sigma(j)}$  is the last letter of  $w_{\sigma(j)} = a_j a_{j+1} \dots a_{j-1}$ , whence  $b_{\sigma(j)} = a_{j-1}$  which is equivalent to the above formula.

Let  $\pi = P(w)$  be the permutation defined by  $\pi(i) = \sigma(\sigma^{-1}(i) + 1)$  where the addition is to be taken mod  $n$ . Actually,  $\pi$  is just the permutation obtained

by writing  $\sigma$  as a word and interpreting it as an  $n$ -cycle. Thus, we have also  $\sigma(i) = \pi^{i-1}(1)$  and

$$a_i = c_{\pi^{i-1}(1)} \quad (3)$$

**Example 1** (continued)

We have, written as a cycle

$$\pi = ( 1 \ 3 \ 7 \ 11 \ 4 \ 8 \ 5 \ 9 \ 2 \ 6 \ 10 )$$

and as an array  $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 3 & 6 & 7 & 8 & 9 & 10 & 11 & 5 & 2 & 1 & 4 \end{pmatrix}$

Substituting in Formula (2) the value of  $a_i$  given by Formula (1), we obtain  $b_i = c_{\sigma(\sigma^{-1}(i)-1)}$  which is equivalent to

$$c_i = b_{\pi(i)} \quad (4)$$

Thus the permutation  $\pi$  transforms the last column of the array of conjugates of  $w$  into the first one. Actually, it can be noted that  $\pi$  transforms any column of this array into the following one.

The computation of  $T(w)$  from  $w$  can be done in linear time. Indeed, provided  $w$  is chosen as a Lyndon word, the order between the conjugates is the same as the order between the corresponding suffixes. The computation of the permutation  $\sigma$  results from the suffix array of  $w$  which can be computed in linear time [4] on a fixed alphabet. The corresponding result on the alphabet of integers is a more recent result (see [1]).

### 3 Inverse transformation

We now show how  $w$  can be recovered from  $T(w)$ . For this, we introduce the following notation. The rank of  $i$  in the word  $y = b_1b_2 \cdots b_n$ , denoted  $\text{rank}(i, y)$  is the number of occurrences of the letter  $b_i$  in  $b_1b_2 \cdots b_i$ .

We observe that for each index  $i$ , and for the aforementioned words  $y = b_1b_2 \cdots b_n$  and  $z = c_1c_2 \cdots c_n$

$$\text{rank}(i, z) = \text{rank}(\pi(i), y). \quad (5)$$

Indeed, we first note that for two words  $u, v$  of the same length and any letter  $a$ , one has  $au < av \Leftrightarrow ua < va \Leftrightarrow u < v$ . Thus for all indices  $i, j$

$$i < j \text{ and } c_i = c_j \Rightarrow \pi(i) < \pi(j). \quad (6)$$

Hence, the number of occurrences of  $c_i$  in  $c_1c_2 \cdots c_n$  is equal to the number of occurrences of  $b_{\pi(i)} = c_i$  in  $b_1b_2 \cdots b_n$ .

To obtain  $w$  from  $T(w) = b_1b_2 \cdots b_n$ , we first compute  $z = c_1c_2 \cdots c_n$  by rearranging the letters  $b_i$  in nondecreasing order. Property (5) shows that  $\pi(i)$  is the index  $j$  such that  $c_i = b_j$  and  $\text{rank}(j, y) = \text{rank}(i, z)$ . This defines the permutation  $\pi$ , from which  $\sigma$  can be reconstructed. An algorithm computing  $\pi$  from  $y = T(w)$  is represented below.

```

PERMUTATION( $b_1 b_2 \dots b_n$ )
1  $c \leftarrow \text{SORT}(b_1 b_2 \dots b_n)$ 
2 for  $i \leftarrow 1$  to  $n$  do
3     if  $i = 1$  or  $c_{i-1} \neq c_i$  then
4          $j \leftarrow 0$ 
5     do  $j \leftarrow j + 1$ 
6     while  $b_j \neq c_i$ 
7      $\pi(i) \leftarrow j$ 
8 return  $\pi$ 

```

This algorithm can be optimized to a linear-time algorithm by storing the first position of each symbol in the word  $z$ .

Finally  $w$  can be recovered from  $z = c_1 c_2 \dots c_n$  and  $\pi$  by Formula (3). The algorithm allowing to recover  $w$  is represented below.

```

WORD( $z, \pi$ )
1  $j \leftarrow 1$ 
2  $a_1 \leftarrow c_1$ 
3 for  $i \leftarrow 2$  to  $n$  do
4      $j \leftarrow \pi(j)$ 
5      $a_j \leftarrow c_j$ 
6 return  $w$ 

```

The computation of  $w$  is not possible without the Parikh vector or equivalently the word  $z$ . One can however always compute the word  $w$  on the smallest possible alphabet associated with permutation  $\pi$  (this is the computation described in [2]).

## 4 Descents of permutations

A descent of a permutation  $\pi$  is an index  $i$  such that  $\pi(i) > \pi(i+1)$ . We denote by  $\text{des}(\pi)$  the set of descents of the permutation  $\pi$ . It is clear by Property (6) that if  $i$  is a descent of  $P(w)$ , then  $c_i \neq c_{i+1}$ . Thus, the number of descents of  $\pi$  is at most equal to  $k-1$  where  $k$  is the number of symbols appearing in the word  $w$ .

**Example 1** (continued) The descents of  $\pi$  appear in boldface.

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & \mathbf{7} & \mathbf{8} & \mathbf{9} & 10 & 11 \\ 3 & 6 & 7 & 8 & 9 & 10 & 11 & 5 & 2 & 1 & 4 \end{pmatrix}$$

Thus  $\text{des}(\pi) = \{7, 8, 9\}$ .

Let us fix an ordered alphabet  $A$  with  $k$  elements for the rest of the paper. The Parikh vector of a word  $w$  on the alphabet  $A$  is the integer vector  $v = (n_1, n_2, \dots, n_k)$  where  $n_i$  is the number of occurrences of the  $i$ -th letter of  $A$  in  $w$ . We say that  $v$  is *positive* if  $n_i > 0$  for  $i = 1, 2, \dots, k$ . We denote by  $\rho(v)$  the

set of integers  $\rho(v) = \{n_1, n_1 + n_2, \dots, n_1 + \dots + n_{k-1}\}$ . When  $v$  is positive,  $\rho(v)$  has  $k - 1$  elements. Let  $\pi = P(w)$  and let  $v$  be the Parikh vector of  $w$ . It is clear by Formula 6 that we have the inclusion  $\text{des}(\pi) \subset \rho(v)$ .

**Example 1** (continued) The Parikh vector of the word  $w = \text{abracadabra}$  is  $v = (5, 2, 1, 1, 2)$  and  $\rho(v) = \{5, 7, 8, 9\}$ .

The following statement results from the preceding considerations.

**Theorem 1** *For any positive vector  $v = (n_1, n_2, \dots, n_k)$ , the map  $w \mapsto P(w)$  is one to one from the set of conjugacy classes of primitive words of length  $n$  on  $A$  with Parikh vector  $v$  onto the set of cyclic permutations on  $\{1, 2, \dots, n\}$  such that  $\rho(v)$  contains  $\text{des}(\pi)$ .*

This result is actually a particular case of a result stated in [7] and essentially due to I. Gessel and C. Reutenauer [6]. The complete result ([7], Theorem 11.6.1 p. 378) establishes a bijection between words of type  $\lambda$  and pairs  $(\sigma, E)$  where  $\sigma$  is a permutation of type  $\lambda$  and  $E$  is a subset of  $\{1, 2, \dots, n-1\}$  with at most  $k-1$  elements containing  $\text{des}(\pi)$ . The type of a word  $w$  of length  $n$  is the partition of  $n$  realized by the length of the factors of its nonincreasing factorization in Lyndon words. The type of a permutation is the partition resulting of the length of its cycles. Thus, Theorem 1 corresponds to the case where  $w$  is a Lyndon word (i.e.  $\lambda$  has only one part) and  $\sigma$  is circular.

We may observe that when the alphabet is binary, i.e. when  $k = 2$ , the result takes a simpler form: the map  $w \mapsto P(w)$  is one-to-one from the set of primitive binary words of length  $n$  onto the set of circular permutations on  $\{1, 2, \dots, n\}$  having one descent.

In the general case, another possible formulation is the following. Let us say that a word  $b_1 b_2 \dots b_n$  is *co-Lyndon* if the permutation  $\pi$  built by Algorithm PERMUTATION is an  $n$ -cycle. It is clear that the map  $w \mapsto T(w)$  is one-to-one from the set of Lyndon words of length  $n$  on  $A$  onto the set of co-Lyndon words of length  $n$  on  $A$ .

The properties of co-Lyndon words have never been studied and this might be an interesting direction of research.

**Example 2** The following array shows the correspondance between Lyndon and co-Lyndon words of length 5 on  $\{a, b\}$ . The permutation  $\pi$  is shown on the right.

Lyndon	co-Lyndon	
<i>aaaab</i>	<i>baaaa</i>	(12345)
<i>aaabb</i>	<i>baaba</i>	(12354)
<i>aabab</i>	<i>bbaaa</i>	(13524)
<i>aabbb</i>	<i>babba</i>	(12543)
<i>ababb</i>	<i>bbbaa</i>	(14253)
<i>abbbb</i>	<i>bbbba</i>	(15432)

## References

- [1] Ricardo Baeza-Yates, Edgar Chávez, and Maxime Crochemore, editors. *Combinatorial Pattern Matching*, volume 2676 of *Lecture Notes in Computer Science*. Springer verlag, 2003.
- [2] Hideo Bannai, Shunsuke Inanaga, Ayumi Shinohara, and Masayuki Takeda. Inferring strings from graphs and arrays, 2003. MFCS 2003.
- [3] Michael Burrows and David J. Wheeler. A block sorting data compression algorithm. Technical report, Digital System Research Center, 1994.
- [4] Maxime Crochemore and Wojciech Rytter. *Jewels of Stringology*. World Scientific, 2002.
- [5] Jean-Pierre Duval and Arnaud Lefebvre. Words over an ordered alphabet and suffix permutations. *Theoretical Informatics and Applications*, 36:249–260, 2002.
- [6] Ira M. Gessel and Christophe Reutenauer. Counting permutations with given cycle structure and descent set. *J. Combin. Theory Ser. A*, 64(2):189–215, 1993.
- [7] M. Lothaire. *Algebraic combinatorics on words*. Cambridge University Press, Cambridge, 2002. With a preface by Jean Berstel and Dominique Perrin.
- [8] Sabrina Mantaci, Antonio Restivo, and Marinella Sciortino. The Burrows-Wheeler transform and sturmian words. *Information Processing Letters*, 86:241–246, 2003.
- [9] Giovanni Manzini. An analysis of the Burrows-Wheeler transform. *J. ACM*, 48(3):407–430, 2001.