A note on the Burrows-Wheeler transformation

Maxime Crochemore   Jacques Désarménien
Dominique Perrin

May 15, 2004

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 [2]. 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 shitfs of w in alphabetic order as the rows $w_1, w_2, \ldots, w_n$ 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 [11] 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 [10] that this transformation was related with notions in combinatorics on words such as Sturmian words. Similar considerations were developped in [1] in a different context. The results presented here are also close to the ones of [4].

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 [9]).

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 $w = a_1a_2 \cdots a_n$ be a word of length n on the alphabet A. The *Parikh vector* of a word w on the alphabet A is the integer vector $v = (n_1, n_2, \ldots, n_k)$ where n_i is the number of occurrences of the i-th letter of A in w. We suppose w to be primitive, i.e. that w is not a power of another word. Let $w_1, w_2, \ldots, w_n$ be the sequence of conjugates of w in increasing alphabetic order. Let b_i denote the last letter of w_i, for $i = 1, \ldots, n$. Then the Burrows-Wheeler transform of w is the word $T(w) = b_1 b_2 \cdots b_n$.

Example 1 Let $w = abracadabra$. 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 $T(w) = rdarcaaaabb$.

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 $z = c_1 c_2 \cdots c_n$ is the nondecreasing rearrangement of w (and of $T(w)$).

Let σ be the permutation of the set $\{1, \ldots, n\}$ such that $\sigma(i) = j$ iff $w_j = a_i a_{i+1} \cdots a_{i-1}$. 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 $1 \le i \le n$

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

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

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

Indeed, $b_{\sigma(j)}$ is the last letter of $w_{\sigma(j)} = a_j a_{j+1} \cdots a_{j-1}$, 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 $\bmod n$. Actually, π is just the permutation obtained by writing σ as a word and interpreting it as an n-cycle. Thus, we have also $\sigma(i) = \pi^{i-1}(1)$ and

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

Example 1 (continued)

We have, written as a cycle

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

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

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

Thus the permutation π transforms the last column of the array of conjugates of w into the first one. Actually, it can be noted that π 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 σ results from the suffix array of w which can be computed in linear time [3] on a fixed alphabet. The corresponding result on the alphabet of integers is a more recent result. It has been proved independently by three groups of researchers, [7], [8] and [6].

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 $y = b_1 b_2 \cdots b_n$, denoted $\mathrm{rank}(i, y)$ is the number of occurrences of the letter b_i in $b_1 b_2 \cdots b_i$.

We observe that for each index i, and for the aforementioned words $y = b_1 b_2 \cdots b_n$ and $z = c_1 c_2 \cdots c_n$

$$\mathrm{rank}(i, z) = \mathrm{rank}(\pi(i), y). \tag{5}$$

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

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

Hence, the number of occurrences of c_i in $c_1 c_2 \cdots c_i$ is equal to the number of occurrences of $b_{\pi(i)} = c_i$ in $b_1 b_2 \cdots b_{\pi(i)}$.

To obtain w from $T(w) = b_1 b_2 \cdots b_n$, we first compute $z = c_1 c_2 \cdots c_n$ 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 $\mathrm{rank}(j, y) = \mathrm{rank}(i, z)$. This defines the permutation π, from which σ can be reconstructed. An algorithm computing π from $y = T(w)$ is represented below.

```
PERMUTATION(b1 b2 ··· bn)
c ← SORT(b1 b2 ··· bn)
for i ← 1 to n do
    if i = 1 or c_{i−1} ≠ c_i then
        j ← 0
    do  j ← j + 1
    while b_j ≠ c_i
    π(i) ← j
return π
```

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 $z = c_1 c_2 \cdots c_n$ and π by Formula (3). The algorithm allowing to recover w is represented below.

```
WORD(z, π)
j ← 1
a_1 ← c_1
for i ← 2 to n do
    j ← π(j)
    a_j ← c_j
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 π (this is the computation described in [1]).

4 Descents of permutations

A descent of a permutation π is an index i such that $\pi(i) > \pi(i+1)$. We denote by $\mathrm{des}(\pi)$ the set of descents of the permutation π. 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 π 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 π 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 $\mathrm{des}(\pi) = \{7, 8, 9\}$.

Let us fix an ordered alphabet A with k elements for the rest of the paper. Let w be a word and $v = (n_1, n_2, \ldots, n_k)$ be the Parikh vector of w. We say that v is *positive* if $n_i > 0$ for $i = 1, 2, \ldots, k$. We denote by $\rho(v)$ the set of integers $\rho(v) = \{n_1, n_1 + n_2, \ldots, n_1 + \cdots + n_{k-1}\}$. 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 $\mathrm{des}(\pi) \subset \rho(v)$.

Example 1 (continued) The Parikh vector of the word $w = abracadabra$ 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 $v = (n_1, n_2, \cdots, n_k)$ with $n = n_1 + \cdots + n_k$, the map $w \mapsto \pi = P(w)$ 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 $\{1, 2, \ldots, n\}$ such that $\rho(v)$ contains $\mathrm{des}(\pi)$.*

This result is actually a particular case of a result stated in [9] and essentially due to I. Gessel and C. Reutenauer [5]. The complete result ([9], Theorem 11.6.1 p. 378) establishes a bijection between words of type λ and pairs (π, E) where π is a permutation of type λ and E is a subset of $\{1, 2, \ldots, n-1\}$ with at most $k-1$ elements containing $\mathrm{des}(\pi)$. 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. λ has only one part) and π is circular. We illustrate the general case with an example for the sake of clarity. For example, the word $w = abaab$ has the nonincreasing factorization in Lyndon words $w = (ab)(aab)$. Thus w has type $(3, 2)$. The corresponding permutation of type $(3, 2)$ is $\pi = (24)(013)$. We have $\mathrm{des}(\pi) = \{3\}$ which is actually included in $\rho(v) = \{3, 5\}$.

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 $\{1, 2, \ldots, n\}$ having one descent.

In the general case, another possible formulation is the following. Let us say that a word $b_1 b_2 \cdots b_n$ is *co-Lyndon* if the permutation π 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 π is shown on the right.

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

References

[1] Hideo Bannai, Shunsuke Inenaga, Ayumi Shinohara, and Masayuki Takeda. Inferring strings from graphs and arrays. In Branislav Rovan and Peter Vojtáš, editors, *Mathematical Foundations of Computer Science 2003*, volume 2747 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[2] Michael Burrows and David J. Wheeler. A block sorting data compression algorithm. Technical report, Digital System Research Center, 1994.

[3] Maxime Crochemore and Wojciech Rytter. *Jewels of Strigology*. World Scientific, 2002.

[4] Jean-Pierre Duval and Arnaud Lefebvre. Words over an ordered alphabet and suffix permutations. *Theoretical Informatics and Applications*, 36:249–260, 2002.

[5] 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.

[6] Juha Kärkkäinen and Peter Sanders. Simple linear work suffix array construction. In Jos C. M. Beaten, Jan Karel Lenstra, Joachim Parrow, and Gerhard J. Woeginger, editors, *Proc. 30th International Colloquium on Automata, Languages and Programming (ICALP '03)*, volume 2619 of *Lecture Notes in Computer Science*, pages 943–955. Springer-Verlag, 2003.

[7] Dong Kyue Kim, Jeong Seop Sim, Heejin Park, and Kunsoo Park. Linear-time construction of suffix arrays. In Ricardo Baeza-Yates, Edgar Chávez, and Maxime Crochemore, editors, *Combinatorial Pattern Matching*, volume 2676 of *Lecture Notes in Computer Science*, pages 186–199. Springer-Verlag, 2003.

[8] Pang Ko and Srinivas Aluru. Space efficient linear time construction of suffix arrays. In Ricardo Baeza-Yates, Edgar Chávez, and Maxime Crochemore, editors, *Combinatorial Pattern Matching*, volume 2676 of *Lecture Notes in Computer Science*, pages 200–210. Springer-Verlag, 2003.

[9] M. Lothaire. *Algebraic Combinatorics on Words*. Cambridge University Press, Cambridge, 2002.

[10] Sabrina Mantaci, Antonio Restivo, and Marinella Sciortino. The Burrows-Wheeeler transform and Sturmian words. *Information Processing Letters*, 86:241–246, 2003.

[11] Giovanni Manzini. An analysis of the Burrows-Wheeler transform. *J. ACM*, 48(3):407–430, 2001.