Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 332 (2005) 567–572
Theoretical Computer Science
www.elsevier.com/locate/tcs

Note

A note on the Burrows–Wheeler transformation

Maxime Crochemore, Jacques Désarménien, Dominique Perrin*

Université de Marne la Vallée, Blvd Descartes, Champs sur Marne, F-77454 Marne la Vallée, France

Received 26 July 2003; received in revised form 12 July 2004; accepted 4 November 2004
Communicated by G. Ausiello

Abstract

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

Keywords: Text compression; Combinatorics on words

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 shifts 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 Mantaci et al. [10] that this transformation was related with notions in combinatorics on words such as Sturmian words. Similar considerations were developed in [1] in a different context. The results presented here are also close to the ones of [4].

* Corresponding author.
E-mail address: perrin@univ-mlv.fr (D. Perrin).

0304-3975/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2004.11.014

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 Gessel and Reutenauer which allows the enumeration of permutations by descents and cyclic type (see [9]).

The paper is organized as follows. In Section 2, we describe the Burrows–Wheeler transformation. Section 3 describes the inverse of the transformation with some emphasis on the computational aspects. Section 4 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 ith 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 $T(w) = b_1b_2\cdots b_n$.

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 $z = c_1c_2\cdots c_n$ 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 $w_j = a_ia_{i+1}\cdots a_{i-1}$. In other terms, $\sigma(i)$ is the rank in the alphabetic order of the ith 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 $1 \leqslant i \leqslant n$

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

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 mod 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, [6–8].

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 $z = c_1 c_2 \cdots c_n$

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

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 $\text{rank}(j, y) = \text{rank}(i, z)$. This defines the permutation π, from which σ can be reconstructed. An algorithm computing π from $y = T(w)$ is represented below.

```
PERMUTATION(b_1 b_2 ··· b_n)
1  c ← SORT(b_1 b_2 ··· b_n)
2  for i ← 1 to n do
3        if i = 1 or c_{i−1} ≠ c_i then
4              j ← 0
5        do    j ← j + 1
6        while b_j ≠ c_i
7        π(i) ← j
8  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)
1  j ← 1
2  a_1 ← c_1
3  for i ← 2 to n do
4        j ← π(j)
5        a_j ← 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 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 $\text{des}(\pi)$ 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 Gessel and 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 of an arbitrary word 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 $\pi = (35)(124)$. Actually, the permutation π is obtained as follows. Its cycles correspond to the Lyndon factors of w. The letters are replaced by the rank in the lexicographic order of the cyclic iterates of the conjugates. In our example, we obtain

$$\begin{array}{lllllll} 1 & a & a & b & a & a & b & \cdots \\ 2 & a & b & a & a & b & a & \cdots \\ 3 & a & b & a & b & a & b & \cdots \\ 4 & b & a & a & b & a & a & \cdots \\ 5 & b & a & b & a & b & a & \cdots \end{array}$$

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 $k = 2$, Theorem 1 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 of an arbitrary alphabet, 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] H. Bannai, S. Inenaga, A. Shinohara, M. Takeda, Inferring strings from graphs and arrays, in: B. Rovan, P. Vojtáš (Eds.), Mathematical Foundations of Computer Science 2003, Lecture Notes in Computer Science, Vol. 2747, Springer, Berlin, 2003.

[2] M. Burrows, D.J. Wheeler, A block sorting data compression algorithm, Tech. Report, Digital System Research Center, 1994.

[3] M. Crochemore, W. Rytter, Jewels of Strigology, World Scientific, Singapore, 2002.

[4] J.-P. Duval, A. Lefebvre, Words over an ordered alphabet and suffix permutations, Theor. Inform. Appl. 36 (2002) 249–260.

[5] I.M. Gessel, C. Reutenauer, Counting permutations with given cycle structure and descent set, J. Combin. Theory Ser. A 64 (2) (1993) 189–215.

[6] J. Kärkkäinen, P. Sanders, Simple linear work suffix array construction, in: J.C.M. Beaten, J.K. Lenstra, J. Parrow, G.J. Woeginger (Eds.), in: Proc. 30th Internat. Coll. on Automata, Languages and Programming (ICALP '03), Lecture Notes in Computer Science, Vol. 2619, Springer, Berlin, 2003, pp. 943–955.

[7] D.K. Kim, J.S. Sim, H. Park, K. Park, Linear-time construction of suffix arrays, in: Ricardo Baeza-Yates, E. Chávez, Maxime Crochemore (Eds.), Combinatorial Pattern Matching, Lecture Notes in Computer Science, Vol. 2676, Springer, Berlin, 2003, pp. 186–199.

[8] P. Ko, S. Aluru, Space efficient linear time construction of suffix arrays, in: Ricardo Baeza-Yates, E. Chávez, Maxime Crochemore (Eds.), Combinatorial Pattern Matching, Lecture Notes in Computer Science, Vol. 2676, Springer, Berlin, 2003, pp. 200–210.

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

[10] S. Mantaci, A. Restivo, M. Sciortino, The Burrows–Wheeler transform and Sturmian words, Inform. Process. Lett. 86 (2003) 241–246.

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