Theoretical Informatics and Applications
Theoret. Informatics Appl. **36** (2002) 249–259
DOI: 10.1051/ita:2002012

WORDS OVER AN ORDERED ALPHABET AND SUFFIX PERMUTATIONS *

Jean-Pierre Duval[1] and Arnaud Lefebvre[2]

Abstract. Given an ordered alphabet and a permutation, according to the lexicographic order, on the set of suffixes of a word w, we present in this article a linear time and space method to determine whether a word w' has the same permutation on its suffixes. Using this method, we are then also able to build the class of all the words having the same permutation on their suffixes, first of all the smallest one. Finally, we note that this work can lead to a method for generating a Lyndon word randomly in linear time or for computing the set of Lyndon words of length n.

Mathematics Subject Classification. 68R15.

Introduction

In this paper we consider the characterization of words having a given mapping σ from $[1 \dots n]$ onto $[1 \dots n]$ as suffix permutation according to the lexicographic order. Such characterization, obtained in linear time from σ, gives the way to build the words having σ as suffix permutation. We assume the words with k different letters to be given with the first k letters of the alphabet. This leads to consider the minimum size of the alphabet required to build words for a given σ. Then we observe how to build Lyndon words using previous techniques.

First of all, recall some basic definitions.

Keywords and phrases: Suffix permutation, Lyndon words.

* *This work was partially supported by a NATO grant PST.CLG.977017.*

[1] LIFAR-ABISS, Faculté des Sciences, Université de Rouen, 76821 Mont-Saint-Aignan Cedex, France; e-mail: `jean-pierre.duval@univ-rouen.fr`

[2] ABISS, UMR 6037 du CNRS, Faculté des Sciences, Université de Rouen,
76821 Mont-Saint-Aignan Cedex, France; e-mail: `arnaud.lefebvre@univ-rouen.fr`

Vocabulary and notations

Let Σ be an ordered alphabet. In the examples we consider letters $a < b < c < d < e \ldots$ To a word $w = x_1 \ldots x_n$, of length n, we associate its suffixes $u_i = x_i \ldots x_n$ for $i = 1, \ldots, n$. We call *suffix permutation* of w the permutation σ over $[1 \ldots n]$ characterized by:

$$u_{\sigma^{-1}(1)} < u_{\sigma^{-1}(2)} \ldots < u_{\sigma^{-1}(n)}.$$

In other words, $\sigma(i)$ is the rank of suffix $u_i = x_i \ldots x_n$ in the set of lexicographically ordered suffixes.

Characterization of words according to a given suffix permutation

One can build the suffix tree of a word in linear time [1, 3], then in a second pass, deduce, the suffix permutation in linear time and the Lyndon words factorization [1]; the Lyndon words factorization is also linear, different from the one-pass left-to-right factorization method [2] and from the tree construction in [4].

Here, we deal more precisely with the following questions:
Given a permutation σ over $[1 \ldots n]$, find the set of words having σ as the suffix permutation. Given a word w, is σ its suffix permutation?

Our answer has the following form:
Given a permutation σ over $[1 \ldots n]$, σ is the suffix permutation of a word $w = x_1 \ldots x_n$ if and only if we have:

$$x_{\sigma^{-1}(1)} r_1 x_{\sigma^{-1}(2)} r_2 \ldots r_{n-1} x_{\sigma^{-1}(n)}$$

where each r_i is an inequality relation ($r_i ='\leq'$ or $r_i ='<'$) over letters $x_{\sigma^{-1}(i)}$ and $x_{\sigma^{-1}(i+1)}$.
The sequence of r_i is computed in linear time from σ and σ^{-1} using the BuildRelations *algorithm.*

Examples

Example 0.1. Given the permutation $\sigma = (1, 3, 5, 2, 4)$, a word $w = x_1 \ldots x_5$ has σ as the suffix permutation if and only if we have:

$$x_1 \leq x_4 \leq x_2 < x_5 \leq x_3.$$

The relation, computed in linear time from σ, allows us to verify whether a word admits σ as the suffix permutation. Moreover it allows us to generate the set of words admitting σ as the suffix permutation. Words with k different letters are given using the k first letters of the alphabet.

Example 0.2. The word $w = aabab$ has $\sigma = (1,3,5,2,4)$ as the suffix permutation because:

$$x_1 = a \leq x_4 = a \leq x_2 = a < x_5 = b \leq x_3 = b$$

$$\begin{array}{cccccc} & 1 & 2 & 3 & 4 & 5 \\ & \mathrm{a} & \mathrm{a} & \mathrm{b} & \mathrm{a} & \mathrm{b} \\ \sigma & 1 & 3 & 5 & 2 & 4. \end{array}$$

Example 0.3. The set of words admitting $\sigma = (1,3,5,2,4)$ as the suffix permutation is obtained, in lexicographic order according to $(x_1, x_4, x_2, x_5, x_3)$, with $x_1 \leq x_4 \leq x_2 < x_5 \leq x_3$.

x_1	x_4	x_2	x_5	x_3	w
a	a	a	b	b	$aabab$
a	a	a	b	c	$aacab$
a	a	b	c	c	$abcac$
a	a	b	c	d	$abdac$
a	b	b	c	c	$abcbc$
a	b	b	c	d	$abdbc$
a	b	c	d	d	$acdbd$
a	b	c	d	e	$acebd$

Algorithm

The sequence of r_i associated to a suffix permutation σ is built in linear time from σ and σ^{-1} using the following algorithm:

```
BuildRelations
Input: a permutation $\sigma$ on $[1 \ldots n]$ and $\sigma^{-1}$, $\sigma$ extended by $\sigma(n+1) = 0$
Output: $(r_i)$, $i \in \mathbb{N}$, $1 \leq i < n$, such that $\sigma$ is the suffix permutation
    of a word $w = x_1 \ldots x_n$ if and only if $x_{\sigma^{-1}(1)} r_1 x_{\sigma^{-1}(2)} r_2 \ldots r_{n-1} x_{\sigma^{-1}(n)}$
for $k \leftarrow 1$ to $n-1$
    if $\sigma(\sigma^{-1}(k)+1) < \sigma(\sigma^{-1}(k+1)+1)$ then
        $r_k \leftarrow' \leq'$
    else $r_k \leftarrow' <'$
```

We establish, in Section 1, that the algorithm is correct regarding input and output parameters and in linear time (Th. 1.1).

Alphabet cardinality

The minimum number of distinct letters necessary to build a word according to a given suffix permutation is variable: it is equal to one plus the number of strict inequalities generated by the BuildRelations algorithm (see Prop. 1.6).

We can see, in Section 2, that there is only one permutation of length n that requires at least n distinct letters (Prop. 2.2 and Prop. 2.8).

Lyndon words generation

In the last section we see how to use the method to generate randomly in linear time the Lyndon words, smallest representative in their classes according to the suffix permutation on words. The set of Lyndon words of length n may be generated.

1. Algorithm correctness

In this section we see the correctness of the BuildRelations algorithm.

Theorem 1.1. *The* BuildRelations *algorithm is correct with respect to input and output predicates, and is in linear time in n.*

Linear time achievement of the algorithm BuildRelations is from the single loop line 4. Our main result, Proposition 1.6, is devoted to properties on input and output conditions. As a corollary, this leads to Theorem 1.1. In the sequel we assume:

- σ is a permutation on an integer segment $[1 \dots n]$, extended in $\sigma(n+1) = 0$;
- $r_1 \dots r_{n-1}$ are inequalities according to the conditions of lines 4 to 7 of the algorithm BuildRelations;
- $w = x_1 \dots x_n$ is a word of length n, for $i \in \mathbb{N}$, $1 \leq i \leq n$, $u_i = x_i \dots x_n$ are suffixes, extended in u_{n+1} the empty word.

Lemma 1.2. *The following conditions are equivalent:*

(a) *for $i, j \in \mathbb{N}$, $1 \leq i, j \leq n$, $\sigma(i) < \sigma(j)$*
if and only if $u_i < u_j$;

(a') *σ is the suffix permutation associated to the word w.*

Proof. According to the definition, the suffix permutation satisfies (a). Since there is a unique permutation satisfying (a), it is the suffix permutation. Observe the consistency of the extension on $n+1$; u_{n+1}, the empty word, is smaller than the other suffixes, has rank 0 and $\sigma(n+1) = 0$. □

The following mapping μ from $[1 \dots n]$ to $[0 \dots n+1]$ is associated to σ:

$$\mu = \sigma(\sigma^{-1} + 1), \text{ i.e., for } k \in \mathbb{N}, 1 \leq k \leq n, \mu(k) = \sigma(\sigma^{-1}(k) + 1).$$

The following result is also of use in the third section:

Proposition 1.3. *Mapping μ is a bijection from $[1 \dots n]$ onto $[0 \dots n] - \{\sigma(1)\}$. For $k \in \mathbb{N}, 1 \leq k < n$ we have:*

- *$r_k =$ " $\leq$ " if and only if $\mu(k) < \mu(k+1)$;*
- *$r_k =$ " $<$ " if and only if $\mu(k) > \mu(k+1)$.*

Proof. Mapping σ is a permutation on $[1 \dots n]$ with extension $\sigma(n+1) = 0$. One can easily verify that:

- σ^{-1} defines a permutation on $[1 \dots n]$;

- $(\sigma^{-1}+1)$ defines a bijection from $[1\dots n]$ onto $[2\dots n+1]$;
- σ defines a bijection from $[2\dots n+1]$ onto $[0\dots n]-\{\sigma(1)\}$;
- μ is a bijection from $[1\dots n]$ onto $[0\dots n]-\{\sigma(1)\}$.

In the algorithm BuildRelations line 5 the condition $\sigma(\sigma^{-1}(k)+1) < \sigma(\sigma^{-1}(k+1)+1)$ can be expressed by $\mu(k) < \mu(k+1)$. It follows that if $\mu(k) < \mu(k+1)$ then $r_k =$ " $\leq$ " else $r_k =$ " $<$ ". □

We define predicate $\mathcal{P}$ for $i, j \in \mathbb{N}$, $1 \leq i, j \leq n$ by:

$$\mathcal{P}(i,j) = [x_i < x_j, \text{ or, } [x_i = x_j \text{ and } \sigma(i+1) < \sigma(j+1)]].$$

Lemma 1.4. *The following conditions are equivalent:*

(b) *for* $k \in \mathbb{N}, 1 \leq k < n$, $\mathcal{P}(\sigma^{-1}(k), \sigma^{-1}(k+1))$;

(b') $x_{\sigma^{-1}(1)} r_1 x_{\sigma^{-1}(2)} \dots r_{n-1} x_{\sigma^{-1}(n)}$.

Proof. Let k be an integer, $1 \leq k < n$. We have

$$\mathcal{P}(\sigma^{-1}(k), \sigma^{-1}(k+1)) = [x_{\sigma^{-1}(k)} < x_{\sigma^{-1}(k+1)}, \text{ or,}$$

$$x_{\sigma^{-1}(k)} = x_{\sigma^{-1}(k+1)} \text{ and } \sigma(\sigma^{-1}(k)+1) < \sigma(\sigma^{-1}(k+1)+1)]$$

and then

$$\mathcal{P}(\sigma^{-1}(k), \sigma^{-1}(k+1)) = [x_{\sigma^{-1}(k)} < x_{\sigma^{-1}(k+1)}, \text{ or,}$$

$$x_{\sigma^{-1}(k)} = x_{\sigma^{-1}(k+1)} \text{ and } \mu(k) < \mu(k+1)].$$

According to Proposition 1.3, we have $\mathcal{P}(\sigma^{-1}(k), \sigma^{-1}(k+1))$ implies $x_{\sigma^{-1}(k)} r_k x_{\sigma^{-1}(k+1)}$ and $x_{\sigma^{-1}(k)} r_k x_{\sigma^{-1}(k+1)}$ implies $\mathcal{P}(\sigma^{-1}(k), \sigma^{-1}(k+1))$. It follows that (b) and (b') are equivalent. □

Before proving, in Proposition 1.6, the equivalence of conditions (a) and (b) from Lemmas 1.2 and 1.4, recall for commodity the following properties.

Property 1.5. *Let $\mathcal{R}_1$ and $\mathcal{R}_2$ two relations on X^2 such that with either $\mathcal{R} = \mathcal{R}_1$ or $\mathcal{R} = \mathcal{R}_2$, for each (x,y) in X^2 one and only one of $\mathcal{R}(x,y)$ or $\mathcal{R}(y,x)$ or $x = y$ conditions holds. The following conditions are equivalent:*

(i) *for all (x,y) in X^2, if $\mathcal{R}_1(x,y)$ then $\mathcal{R}_2(x,y)$;*

(ii) *for all (x,y) in X^2, if $\mathcal{R}_2(x,y)$ then $\mathcal{R}_1(x,y)$;*

(iii) *for all (x,y) in X^2, $\mathcal{R}_1(x,y)$ if and only if $\mathcal{R}_2(x,y)$.*

Proof. (i) implies (ii). Let x and y in X such that $\mathcal{R}_2(x,y)$. Neither $x = y$, nor $\mathcal{R}_1(y,x)$ since it should imply $x = y$ or $\mathcal{R}_2(y,x)$ contradicting $\mathcal{R}_2(x,y)$; it follows that $\mathcal{R}_1(x,y)$. Then (i) implies (ii). Similarly (ii) implies (i). Thus (i), (ii) and (iii) are equivalent. □

We can check that $\mathcal{P}$ is a total ordering and, then, that we are in the conditions of Property 1.5.

Proposition 1.6. *Let $r_1 \dots r_{n-1}$ be the inequalities produced by the algorithm* BuildRelations. *The following conditions are equivalent:*

(a) *for $i, j \in \mathbb{N}$, $1 \le i, j \le n$, $\sigma(i) < \sigma(j)$ if and only if $u_i < u_j$ (i.e. σ is the suffix permutation associated to w);*

(b) *for $1 \le k < n$, $\mathcal{P}(\sigma^{-1}(k), \sigma^{-1}(k+1))$ (i.e. word $w = x_1 \dots x_n$ and inequalities $r_1 \dots r_{n-1}$ satisfy $x_{\sigma^{-1}(1)} r_1 x_{\sigma^{-1}(2)} \dots r_{n-1} x_{\sigma^{-1}(n)}$);*

(c) *for $i, j \in \mathbb{N}$, $1 \le i, j \le n$, $\sigma(i) < \sigma(j)$ if and only if $\mathcal{P}(i, j)$.*

Proof. (a) implies (b).

Assume condition (a) holds. Let k be an integer, $1 \le k < n$. Let $i = \sigma^{-1}(k)$ and $j = \sigma^{-1}(k+1)$. Then, we have $k = \sigma(i) < \sigma(j) = k+1$. Thus from (a) we deduce $u_i < u_j$. Two cases are to be considered:

- either $x_i < x_j$, it follows that $\mathcal{P}(i, j)$;
- or $x_i = x_j$ and $u_{i+1} < u_{j+1}$, from condition (a) we have $\sigma(i+1) < \sigma(j+1)$, it follows that $\mathcal{P}(i, j)$.

In both cases $\mathcal{P}(i, j)$, *i.e.*, $\mathcal{P}(\sigma^{-1}(k), \sigma^{-1}(k+1))$. This proves (a) implies (b).

(b) implies (c).

Assume condition (b) holds. Let $i, j \in \mathbb{N}$, $1 \le i, j \le n$ with $\sigma(i) < \sigma(j)$. Let $m = \sigma(j) - \sigma(i)$, $k = \sigma(i)$ and $i_0 = \sigma^{-1}(k)$, $i_1 = \sigma^{-1}(k+1)$, … ,$i_m = \sigma^{-1}(k+m)$. Note that $i_0 = i$ and $i_m = j$. From condition (b) we have $\mathcal{P}(i_0, i_1)$, $\mathcal{P}(i_1, i_2)$, … , $\mathcal{P}(i_{m-1}, i_m)$. Since $\mathcal{P}$ is a transitive relation, we have $\mathcal{P}(i_0, i_m)$, *i.e.*, $\mathcal{P}(i, j)$. Condition: for $i, j \in \mathbb{N}$, $1 \le i, j \le n$, $\sigma(i) < \sigma(j)$, implies $\mathcal{P}(i, j)$ holds. According to Property 1.5 it follows that condition (c) holds. This proves (b) implies (c).

(c) implies (a).

Assume condition (c) holds. Suppose there are $i, j \in \mathbb{N}$, $1 \le i, j \le n$ with $\sigma(i) < \sigma(j)$ and $u_i \ge u_j$. Take $i, j \in \mathbb{N}$, $1 \le i, j \le n$, with the extra condition $i + j$ is maximal with $\sigma(i) < \sigma(j)$ and $u_i \ge u_j$. From condition (c) and $\sigma(i) < \sigma(j)$, we have $\mathcal{P}(i, j)$. There is an alternative:

- either $x_i < x_j$ then $u_i < u_j$ which contradicts $u_i \ge u_j$;
- or $x_i = x_j$ and $\sigma(i+1) < \sigma(j+1)$. There are three cases depending on whether $i = n$ or not, $j = n$ or not:
  - $i = n$ and $j < n$. It follows that $u_{i+1} = \varepsilon$ and $u_{j+1} \ne \varepsilon$. Since $x_i = x_j$ we have $u_i < u_j$ which contradicts $u_i \ge u_j$;
  - $i < n$ and $j = n$. It follows that $\sigma(i+1) > 0$ and $\sigma(j+1) = 0$ contradicting the assumption $\sigma(i+1) < \sigma(j+1)$;
  - $i < n$ and $j < n$. It follows that $\sigma(i+1) < \sigma(j+1)$. From the assumption $i + j$ maximal defecting condition $u_i < u_j$, we have $u_{i+1} < u_{j+1}$. From $x_i = x_j$ and $u_{i+1} < u_{j+1}$ follows that $u_i < u_j$ contradicting $u_i \ge u_j$.

In both cases it leads to $u_i < u_j$ in contradiction with $u_i \ge u_j$. It follows that for i, $j \in \mathbb{N}$, $1 \le i, j \le n$, $\sigma(i) < \sigma(j)$ implies $u_i < u_j$. According to Property 1.5, condition (a) holds. This proves (c) implies (a).

Conditions (a), (b) and (c) are equivalent. This concludes the proof of Proposition 1.6. □

We are now able to prove Theorem 1.1.

Proof of Theorem 1.1. Mappings σ and σ^{-1} are in direct access, constant time. Steps 4 to 7 are in linear time in n. From Proposition 1.6 conditions (a) and (b) are equivalent, *i.e.*, σ is the suffix permutation for $w = x_1 \dots x_n$ if and only if condition $x_{\sigma^{-1}(1)} r_1 x_{\sigma^{-1}(2)} \dots r_{n-1} x_{\sigma^{-1}(n)}$ holds. □

2. Suffix permutation and alphabet cardinality

In this section we see how the mapping $\mu = \sigma(\sigma^{-1} + 1)$, associated to a permutation σ, allows us to determine directly the minimum number of different letters required to build a word w with suffix permutation σ (Prop. 2.2). We can deduce that n different letters can be required for a word of length n (Prop. 2.4) in a single case (Prop. 2.8). We say that an mapping μ from $[1 \dots n]$ to $[0 \dots n]$ is μ-valid if and only if there exists a permutation σ on $[1 \dots n]$ such that $\mu = \sigma(\sigma^{-1} + 1)$. We deduce the conditions such that μ is μ-valid, we define the links between μ and σ (Lem. 2.6). Then we deduce a method to build a μ-valid mapping.

2.1. Minimum number of letters according to a given suffix permutation

The number of different letters required to build a word admitting a given permutation σ over $[1 \dots n]$ as suffix permutation is from 1 to n. It is always possible to use n different letters for a suffix permutation over $[1 \dots n]$ but it is not always necessary.

Example 2.1. Consider $\sigma = (6, 5, 4, 3, 2, 1)$. The word $w' = fedcba$, built with 6 different letters $a < b < c < d < e < f$, and the word $w'' = a^6$, have the same suffix permutation σ.

The maximum number of n different letters can be required in some cases: we can verify this, for a given suffix permutation, by studying $\mu = \sigma \circ (\sigma^{-1} + 1)$.

Proposition 2.2. *Let σ be a permutation over $[1 \dots n]$ extended by $\sigma(n+1) = 0$, and $\mu = \sigma \circ (\sigma^{-1} + 1)$. The minimum number of different letters required to build a word admitting σ as the suffix permutation is equal to:*

$$1 + \operatorname{card}\{k \mid 1 \leq k < n \text{ and } \mu(k) > \mu(k+1)\}\cdot$$

Proof. A word $w = x_1 \dots x_n$ admitting σ as suffix permutation satisfies the relation

$$x_{\sigma^{-1}(1)} r_1 x_{\sigma^{-1}(2)} \dots r_{\sigma^{-1}(n-1)} x_n$$

where $r_1 \dots r_{n-1}$ is the sequence built by the algorithm BuildRelations.

The minimum number of different letters is then equal to 1 plus the number of strict inequalities in $r_1 \dots r_{n-1}$. According to the algorithm it is equal to:

$$1 + \mathrm{card}\{k \mid 1 \leq k < n \text{ and } \mu(k) > \mu(k+1)\}\cdot$$

□

Example 2.3. Consider $\sigma = (2, 3, 1, 4)$. We have $\sigma^{-1} = (3, 1, 2, 4)$, $\sigma^{-1} + 1 = (4, 2, 3, 5)$ and $\mu = (4, 3, 1, 0)$. The words $x_1 \dots x_n$ admitting σ as the suffix permutation have to satisfy $x_3 < x_1 < x_2 < x_4$. Only one word satisfies this criterion, it is $w = bcad$.

2.2. Suffix permutation needing at least n distinct letters

More generally, given n, there is a unique permutation which needs at least n distinct letters.

Proposition 2.4. *Let σ be the permutation over $[1 \dots n]$ defined for k, $0 \leq k \leq n-1$ by:*

$$\sigma(n-k) = \begin{cases} n - \frac{k}{2} & \text{if } k \text{ is even} \\ \frac{k+1}{2} & \text{if } k \text{ is odd.} \end{cases}$$

The only word, admitting σ as the suffix permutation, is written with the n letters of the ordered alphabet $\{c_1 < c_2 < \dots < c_n\}$ by $w = x_1 \dots x_n$ with $x_{\sigma^{-1}(n-k)} = c_{n-k}$.

Proof. One can verify that for $i = 1$ to n:

$$\sigma \circ (\sigma^{-1} + 1)(i) = \begin{cases} n - i + 1 & \text{for } i = 1 \text{ to } m \\ n - i & \text{for } i = m + 1 \text{ to } n \end{cases}$$

with $m = \frac{n}{2}$ for n even, $m = \frac{n-1}{2}$ for n odd.

The mapping $\sigma \circ (\sigma^{-1} + 1)$ is strictly decreasing, consequently the word w admitting σ as the suffix permutation verifies:

$$x_{\sigma^{-1}(1)} r_1 x_{\sigma^{-1}(2)} \dots r_{\sigma^{-1}(n-1)} x_n.$$

It is written with n different letters: $x_{n-k} = c_{\sigma(n-k)}$. □

Remark. The permutation σ is built over $[1 \dots n]$ by following the next two rules:

(1) going over, two by two, from position $n-1$ to 1 if n is even (resp. $n-1$ to 2 if n is odd), and numbering them (in an increasing manner) from 1 to $\frac{n}{2}$ (resp. from 1 to $\frac{n-1}{2}$);

(2) going over, two by two, from position n to 2 if n is even (resp. n to 1 if n is odd), and numbering them (in a decreasing manner) from n to $\frac{n}{2}+1$ (resp. from n to $\frac{n+1}{2}$).

Example 2.5. For $n = 6$:

- step (1): $\sigma(5) = 1$, $\sigma(3) = 2$, $\sigma(1) = 3$;
- step (2): $\sigma(6) = 6$, $\sigma(4) = 5$, $\sigma(2) = 4$.

One can easily verify that the only word admitting σ as suffix permutation is $w = cdbeaf$.

2.3. Suffix permutation according to a given number of letters

Lemma 2.6. *Let μ be an mapping from $[1 \dots n]$ to $[0 \dots n]$. The following two conditions (i) and (ii) are equivalent:*

(i) *there exists a permutation σ over $[1 \dots n]$ extended by $\sigma(n+1) = 0$ such that $\mu = \sigma \circ (\sigma^{-1} + 1)$ (i.e. is μ-valid);*

(ii) $0 \in \mu[1 \dots n]$*, μ is injective and* $\{\mu^{-1}(0), \mu^{-2}(0), \dots, \mu^{-n}(0)\} = [1 \dots n]$.

Then if (i) holds, then μ and σ satisfy the following conditions:

(i_1) $\sigma(1) = [0 \dots n] - \mu[1 \dots n]$*;*

(i_2) $\sigma(n+1) = 0$*, and, for each* $i \in \mathbb{N}$, $1 \le i \le n$*:* $\sigma(i+1) = \mu(\sigma(i))$*;*

(i_3) $\sigma(n+1) = 0$*, and, for each* $i \in \mathbb{N}$, $1 \le i \le n$*:* $\sigma(i) = \mu^{-1}(\sigma(i+1))$*;*

(i_4) $\sigma(n+1) = 0$*, and, for each* $i \in \mathbb{N}$, $1 \le i \le n$*:* $\sigma(i) = \mu^{-n+i-1}(0)$.

Proof. From condition (i), one can verify that conditions (i_1), (i_2), (i_3) and (i_4) hold. Thus (ii) holds. From condition (ii), let σ^{-1} be defined by the condition (i_4). Then one can verify that σ^{-1} is a permutation on $[1 \dots n]$, and, $\mu = \sigma(\sigma^{-1} + 1)$. Then (i) and the conditions (i_1), (i_2), (i_3) and (i_4) hold. □

Example 2.7. Consider $\mu = (4, 3, 0, 5, 2)$; we necessarily have

$$x_{\sigma^{-1}(1)} < x_{\sigma^{-1}(2)} < x_{\sigma^{-1}(3)} \le x_{\sigma^{-1}(4)} < x_{\sigma^{-1}(5)}.$$

We need, at least, four different letters (one "a", one "b", two "c" and one "d") to build a suffix permutation σ such that $\mu = \sigma \circ (\sigma^{-1} + 1)$:

	1	2	3	4	5
μ	4	3	0	5	2
σ	1	4	5	2	3
σ^{-1}	1	4	3	2	5.

The word $w = acdbc$ admits σ as the suffix permutation.

From Lemma 2.6(ii) one obtain randomly a μ-valid mapping from $[1 \dots n]$ onto $[0 \dots n]$ by choosing randomly in $[1 \dots n]$ the successive values (all different) $j_1, \dots, j_n$ for $\mu^{-1}(0), \mu^{-2}(0), \dots, \mu^{-n}(0)$. Then μ is defined by:

$$\mu(j_1) = 0, \mu(j_2) = j_1, \dots, \mu(j_n) = j_{n-1}.$$

The following algorithm gives a method to generate a μ-valid mapping randomly.

```
$\mu$-VALID
Input: $\mu$ array of $n$ integers
          (Initially, each column is free)
Output: $\mu$ containing a $\mu$-valid mapping
$i \leftarrow 0$
while there exists a free column in $\mu$ do
    $j \leftarrow$ choose randomly the index of a free column
              (column $j$ is no more free)
    $\mu[j] \leftarrow i$
    $i \leftarrow j$
```

As an application of Lemma 2.6, we consider the case of a μ-valid mapping requiring n different letters.

Proposition 2.8. *The only permutation σ over $[1\ldots n]$ requiring a minimum of n distinct letters to write a word w admitting σ as the suffix permutation is:*

$$\sigma(n-k) = \begin{cases} n - \frac{k}{2} & \textit{if } k \textit{ is even} \\ \frac{k+1}{2} & \textit{if } k \textit{ is odd.} \end{cases}$$

Proof. If σ requires n distinct letters, the associated function $\mu = \sigma \circ (\sigma^{-1} +1)$ is strictly decreasing in $[0\ldots n]$. The only decreasing function satisfying the conditions of Lemma 2.6 is:

- $\mu = (n, n-1, \ldots, \frac{n}{2}+1, \frac{n}{2}-1, \ldots, 0)$ if n is even;
- $\mu = (n, n-1, \ldots, \frac{n+1}{2}+1, \frac{n+1}{2}-1, \ldots, 0)$ if n is odd.

One can verify that the presence of $\frac{n}{2}$ for n even ($\frac{n+1}{2}$ for n odd) implies a cycle in $\frac{n}{2}$: either $\mu(\frac{n}{2}) = \frac{n}{2}$, or, $\mu(\frac{n}{2}) = \frac{n}{2}+1$ and $\mu(\frac{n}{2}+1) = \frac{n}{2}$ (resp. $\frac{n+1}{2}$ for n odd). The associated mapping σ is then the only one requiring n distinct letters. □

3. Lyndon words

In this section we apply the previous techniques to the construction of Lyndon words.

A Lyndon word is a word that is smaller than each of its proper suffixes. This property is characterized here by $\sigma(1) = 1$, then by the fact that the associated mapping μ is an injective mapping with no cycle from $[1\ldots n]$ onto $[0\ldots n] - \{1\}$. Constructing a μ-valid mapping (resp. a permutation σ on $[1\ldots n]$) leads to a Lyndon word if and only if column 1 is the last to be filled (resp. $\sigma(1) = 1$). Thus we generate a Lyndon word of length n randomly, and in linear time, by generating either σ or μ.

Two words are σ-equivalent if they have the same suffix permutation σ. We have seen how to build the smallest representative word of a given σ-class using the BuildRelations algorithm. The σ-classes of Lyndon words are characterized by $\sigma(1) = 1$. Applying this techniques by choosing randomly a permutation σ with $\sigma(1) = 1$ we obtain a random generation of each σ-classes of Lyndon words and the construction of their smallest representative words.

The following algorithm describes this method.

```
RandomLyndonWord
Input: $n$ the length of the desired Lyndon word
Output: $w = x_1 \ldots x_n$ a Lyndon word
Random generation of $\sigma$ on $[1 \ldots n]$ with $\sigma(1) = 1$
Generation of $r_1 \ldots r_{n-1}$ with the BuildRelations algorithm
$x_{\sigma^{-1}(1)} \leftarrow a$
for $k \leftarrow 1$ to $n-1$ do
    if $r_k =' \leq '$ then
        $x_{\sigma^{-1}(k+1)} \leftarrow x_{\sigma^{-1}(k)}$
    else $x_{\sigma^{-1}(k+1)} \leftarrow next(x_{\sigma^{-1}(k)})$
(next(c) is the letter following letter c in the ordered alphabet)
```

We assume the random generation of an integer is performed in constant time, thus the method is linear in time in n.

To generate the smallest representative word for each σ-classes of Lyndon words of length n, we can compute the set of permutations σ on $[1 \ldots n]$ with $\sigma(1) = 1$ and generate the smallest words associated to each σ.

4. Conclusion

In order to continue this work, we have to answer the following question: how generating efficiently a μ-valid mapping with exactly m decreasing steps $\mu(k) > \mu(k+1)$, leading (according to Prop. 2.4) to words with at least $m+1$ different letters?

References

[1] M. Crochemore, C. Hancart and T. Lecroq, *Algorithmique du texte.* Vuibert (2001).

[2] J.-P. Duval, Factorizing Words over an Ordered Alphabet. *J. Algorithms* **4** (1983) 363-381.

[3] E.M. McCreight, A Space-Economical Suffix Tree Construction Algorithm. *J. Algorithms* **23** (1976) 262-272.

[4] C. Hohlweg and C. Reutenauer, *Lyndon words, permutations and trees,* Rapport interne 2002-017. Université Louis Pasteur de Strasbourg.

Communicated by J. Berstel.
Received January, 2002. Accepted October, 2002.