

# An Efficient Shift Rule for the Prefer-Max De Bruijn Sequence<sup>☆</sup>

Gal Amram<sup>a</sup>, Yair Ashlagi<sup>a</sup>, Amir Rubin<sup>a</sup>, Yotam Svoray<sup>a</sup>, Moshe Schwartz<sup>b</sup>,  
Gera Weiss<sup>a</sup>

<sup>a</sup>*Department of Computer Science, Ben-Gurion University of The Negev*

<sup>b</sup>*Department of Electrical and Computer Engineering, Ben-Gurion University of The Negev*

---

## Abstract

A shift rule for the prefer-max De Bruijn sequence is formulated, for all sequence orders, and over any finite alphabet. An efficient algorithm for this shift rule is presented, which has linear (in the sequence order) time and memory complexity.

**Keywords:** De Bruijn sequence, Ford sequence, prefer-max sequence, shift rule  
**2010 MSC:** 94A55, 05C45, 05C38

---

## 1. Introduction

A  $k$ -ary De Bruijn sequence of order  $n$  (denoted  $(n, k)$ -DB), is a string  $\langle v_i \rangle_{i=0}^{k^n-1} \triangleq v_0 v_1 \dots v_{k^n-1}$  over the alphabet  $[k] \triangleq \{0, 1, \dots, k-1\}$ , i.e.,  $v_i \in [k]$  for all  $i \in \mathbb{Z}_{k^n}$ , such that all  $n$ -substrings  $v_i v_{i+1} \dots v_{i+n-1}$  are distinct (note that  $i \in \mathbb{Z}_{k^n}$  means that indices are taken modulo  $k^n$ ). Discovered as early as 1894 by Flye-Sainte Marie [4], they have been rediscovered over the years many times, among them, by De Bruijn [2], whose name they now carry.

Many  $(n, k)$ -DB sequences exist, their exact number given by  $((k-1)!)^{k^{n-1}} \cdot k^{k^{n-1}-n}$ , as proved in [16]. Of these, a particular sequence stands out, featuring in many past works. Consider first the binary case,  $k = 2$ , start the sequence with

---

<sup>☆</sup>This work was supported in part by the Israeli Science Foundation (ISF) under grant no. 130/14.

*Email addresses:* galamra@cs.bgu.ac.il (Gal Amram),  
ashlagi@post.bgu.ac.il (Yair Ashlagi), amirrub@cs.bgu.ac.il (Amir Rubin),  
ysavorai@post.bgu.ac.il (Yotam Svoray), schwartz@ee.bgu.ac.il (Moshe Schwartz), geraw@cs.bgu.ac.il (Gera Weiss)

$0^n$ , and add bit by bit, always preferring to append a 1, unless it creates an  $n$ -string that has already been seen previously. After obtaining a sequence of length  $2^n$ , move the  $0^n$  prefix to the end of the sequence. The result is an  $(n, 2)$ -DB sequence dubbed the prefer-one sequence, sometimes also called a Ford sequence due to [5], even though already discovered in [12]. See also the survey [7] and all the references therein. By complementing all the bits, we obtain the prefer-min  $(n, k)$ -DB sequence. In the non-binary case, we can replace the prefer-one rule by a prefer-max (assuming a lexicographical order of the alphabet), resulting in the lexicographically largest  $(n, k)$ -DB sequence, and symmetrically (by complementation), a prefer-min  $(n, k)$ -DB sequence which is the lexicographically smallest  $(n, k)$ -DB sequence.

The greedy bit-by-bit algorithm of [12] (also called by Knuth a “granddaddy” algorithm [10, Sec. 7.2.1.1]) is certainly an inefficient way of generating the prefer-max sequence. Several suggestions have been made since to improve the efficiency of the sequence construction. Fredricksen and Kessler [8], and Fredricksen and Maiorana [9] showed that the prefer-max sequence is in fact a concatenation of certain (Lyndon) words. While seemingly an inefficient way to generate the prefer-max sequence, a later careful analysis [14] has shown that this decomposition allows us to generate the sequence of length  $k^n$  in  $O(k^n)$  time. Another efficient block concatenation construction was suggested in [13].

However, another equally important way of generating sequences is of interest, namely, by using a shift rule. It is well known that any  $(n, k)$ -DB sequence  $\langle v_i \rangle_{i=0}^{k^n-1}$  can be generated by a finite shift register (FSR) of order  $n$ , i.e., there exists a shift-rule function  $f: [k]^n \rightarrow [k]$  such that  $v_{i+n} = f(v_i, v_{i+1}, \dots, v_{i+n-1})$  for all  $i \in \mathbb{Z}_{k^n}$ . Several efficiently computable shift rules for De Bruijn sequences are known (see [15] for a comprehensive list). However, with a single exception, they only generate non prefer-max sequences, and the only exception [6], addresses only the generation of binary prefer-one sequences.

The main contribution of this paper is an efficient shift-rule function for the  $(n, k)$  prefer-max De Bruijn sequence. The shift rule, given in Algorithm 1, is based on the decomposition of the prefer-max sequence found by [9], and operates in time  $O(n)$ . This closes a gap in the literature, since while efficient constructions for the entire prefer-max sequence are known, an efficient shift rule is only known in the binary case.

The paper is organized as follows. In Section 2 we provide the necessary notation used throughout the paper, and recall some relevant results. In Section 3 we provide a mathematical expression for the shift rule. We proceed in Section 4

to devise an efficient algorithm that implements the shift rule. We conclude in Section 5 with a short discussion of the results.

## 2. Preliminaries

Let us start by reviewing the necessary definitions and previous results, before presenting the new results. To avoid trivialities, we assume throughout the paper that  $n, k \geq 2$ . With our alphabet symbols  $[k]$  we associate a lexicographical order,  $0 < 1 < \dots < k-1$ . This order is extended in the natural way to all finite strings from  $[k]^*$  by defining  $x < y$  if either  $x$  is a prefix of  $y$ , or there exist (possibly empty)  $u, v, w \in [k]^*$  and two letters  $\sigma, \sigma' \in [k]$ ,  $\sigma < \sigma'$ , such that  $x = u\sigma v$  and  $y = u\sigma'w$ .

Given a string  $v \triangleq \sigma_0\sigma_1\dots\sigma_{n-1}$ , with  $\sigma_i \in [k]$ , we define the *rotation* operator,  $\mathcal{R}: [k]^n \rightarrow [k]^n$  as  $\mathcal{R}(v) \triangleq \sigma_1\sigma_2\dots\sigma_{n-1}\sigma_0$ . We say that two strings  $v, u \in [k]^n$  are *cyclically equivalent* if there exists  $i \in \mathbb{Z}$  such that  $v = \mathcal{R}^i(u)$ . The equivalence classes under  $\mathcal{R}$  are called *necklaces*. The number of necklaces, denoted by  $Z_k(n)$ , is known to be  $Z_k(n) \triangleq \frac{1}{n} \sum_{d|n} \phi(d) k^{n/d}$ , where  $\phi$  is Euler's  $\phi$ -function (also the number of cycles in the pure cycling FSR, see [17]). Let  $v \in [k]^n$  be a string. The cyclic order of  $v$ , denoted  $o(v)$ , is the smallest positive integer  $o(v) \in \mathbb{N}$  such that  $\mathcal{R}^{o(v)}(v) = v$  or, alternatively, it is the number of elements in the necklace containing  $v$ . If  $o(v) = |v|$  we say that  $v$  is *primitive*. For any  $v \in [k]^n$  there is a unique primitive word  $w \in [k]^d$ , for some  $d|n$ , such that  $v = w^{n/d}$ . This  $w$  is called the *root* of  $v$ .

A primitive word that is lexicographically least in its necklace is called a *Lyndon word*. If  $L \in [k]^+$  is a Lyndon word, we shall also find it useful to define  $L^m$  as an *expanded Lyndon word*, for all  $m \in \mathbb{N}$ . Additionally, we can arrange all the expanded Lyndon words of length  $n$  in increasing lexicographical order

$$L_0^{r_0} < L_1^{r_1} < \dots < L_{Z_k(n)-1}^{r_{Z_k(n)-1}}.$$

The main result of [8, 9] (rephrased to simplify the presentation) is that the prefer-min  $(n, k)$ -DB sequence is in fact the concatenation  $L_0L_1\dots L_{Z_k(n)-1}$ . We shall use this fact later on, and call it the FKM factorization. We also comment that by complementing all the letters via  $\psi: [k] \rightarrow [k]$ ,  $\psi(i) \triangleq k-1-i$ , for all  $i \in [k]$ , the prefer-min  $(n, k)$ -DB sequence becomes the prefer-max  $(n, k)$ -DB sequence, and vice versa. We extend  $\psi$  to operate on strings in the natural way, i.e., applying it to all letters of the word.

**Example 1.** Fix  $n = 2$  and  $k = 3$ . We then have the following lexicographical order of expanded Lyndon words,

$$00 < 01 < 02 < 11 < 12 < 22$$

hence

$$L_0 = 0 \quad L_1 = 01 \quad L_2 = 02 \quad L_3 = 1 \quad L_4 = 12 \quad L_5 = 2,$$

and indeed the prefer-min  $(2, 3)$ -DB sequence is  $L_0 L_1 L_2 L_3 L_4 L_5 = 001021122$ . After complementing each symbol we obtain  $\psi(L_0 L_1 L_2 L_3 L_4 L_5) = 221201100$ , which is the prefer-max  $(2, 3)$ -DB sequence.

### 3. Shift-Rule Construction

In this section we state and prove our shift-rule construction. For ease of presentation, we work with the prefer-min sequence, while remembering that by simply complementing symbols with  $\psi$ , this is equivalent to working with the prefer-max sequence.

We first require a definition that distinguishes another necklace member that is not necessarily the expanded Lyndon word  $L_i^{r_i}$  defined above.

**Definition 2.** A word  $v \in [k]^n$  is a necklace *head*, tested by the predicate  $\text{head}(v)$ , if we can write  $v$  as  $v = (k-1)^t w \sigma$ , where  $w \in [k]^{n-t-1}$ ,  $\sigma \in [k-1]$  (i.e.,  $\sigma \neq k-1$ ), and  $\mathcal{R}^t(v) = w \sigma (k-1)^t$  is an expanded Lyndon word.

We briefly note that the necklace containing the single string  $(k-1)^n$  does not formally have a necklace head, whereas all other necklaces have a unique necklace head. Additionally, by the above definition, if  $(k-1)^t w \sigma$  is a necklace head, either  $w = \varepsilon$  is empty, or it does not start with the letter  $k-1$ .

We now define our shift rule. Traditionally, a shift rule is a function that takes  $n$  consecutive symbols in the sequence (i.e., the current state of an FSR generating the sequence) and its output is the next symbol. However, we will find it more convenient to define a shift rule as providing the next state of the FSR. Specifically, let  $\langle v_i \rangle_{i=0}^{k^n-1}$  be the prefer-min  $(n, k)$ -DB sequence. A shift rule for the sequence is a function  $f: [k]^n \rightarrow [k]^n$  such that  $f(v_i v_{i+1} \dots v_{i+n-1}) = v_{i+1} v_{i+2} \dots v_{i+n}$ , for all  $i \in \mathbb{Z}_{k^n}$ .

**Definition 3.** Let  $\text{next}: [k]^n \rightarrow [k]^n$  be defined by

$$\text{next}(\sigma w) \triangleq \begin{cases} w(\sigma + 1) & \text{if } \sigma \neq k - 1 \text{ and } \text{head}(w\sigma), \\ w(\min S) & \text{if } \sigma = k - 1 \text{ and} \\ & S \triangleq \{\sigma' \in [k - 1] : \text{head}(w\sigma')\} \neq \emptyset, \\ w\sigma & \text{otherwise,} \end{cases}$$

where  $\sigma \in [k]$  and  $w \in [k]^{n-1}$ .

The main result of this section is the following theorem.

**Theorem 4.** *next is a shift rule for the prefer-min  $(n, k)$ -DB sequence.*

In order to prove Theorem 4 we state a sequence of lemmas, building up to the main result.

**Lemma 5.** *A string  $v \in [k]^+$  is an expanded Lyndon word if and only if  $v \leq \mathcal{R}^i(v)$  for all  $i \in \mathbb{Z}$  (i.e., it is lexicographically least in its necklace).*

*Proof.* Consider the (unique) decomposition  $v = w^t$ , where  $w$  is primitive. Note that  $\mathcal{R}^i(v) = (\mathcal{R}^i(w))^t$ . Thus,  $v \leq \mathcal{R}^i(v)$  if and only if  $w \leq \mathcal{R}^i(w)$ , which holds for all  $i \in \mathbb{Z}$  if and only if  $w$  is a Lyndon word.  $\square$

A first step we take is showing that increasing the rightmost letter that is not  $k - 1$  in an expanded Lyndon word maintains the expanded Lyndon property.

**Lemma 6.** *If  $w\sigma(k - 1)^t \in [k]^n$  is an expanded Lyndon word and  $\sigma \in [k - 1]$  then  $w(\sigma + 1)(k - 1)^t$  is also an expanded Lyndon word.*

*Proof.* Let  $v \triangleq w\sigma(k - 1)^t \in [k]^n$  and  $v' \triangleq w(\sigma + 1)(k - 1)^t$ . Assume to the contrary that  $v'$  is not an expanded Lyndon word. Hence, by Lemma 5, there exists a non-trivial rotation amount  $1 \leq i \leq n - 1$  such that  $\mathcal{R}^i(v') < v'$ .

Let us write  $v' = w_1x$  such that  $\mathcal{R}^i(v') = xw_1$ . Because  $v$  is an expanded Lyndon word and  $\sigma < k - 1$ , the first letter of  $w$  is not  $k - 1$ . Since  $\mathcal{R}^i(v') = xw_1 < w_1x$ , we also have that the first letter of  $x$  is not  $k - 1$ . It then follows that  $w_1$  is a non-empty prefix of  $w$ , and we may write  $w = w_1w_2$ . We now have

$$\mathcal{R}^i(v') = w_2(\sigma + 1)(k - 1)^tw_1 < w_1w_2(\sigma + 1)(k - 1)^t = v'.$$

But then we have

$$\mathcal{R}^i(v) = w_2\sigma(k - 1)^tw_1 < w_1w_2\sigma(k - 1)^t = v, \quad (1)$$

since reducing  $\sigma + 1$  to  $\sigma$  on the left-hand side occurs strictly closer to the word's beginning (most-significant symbol) compared with the same symbol reduction on the right-hand side. By Lemma 5, (1) contradicts the fact that  $v$  is an expanded Lyndon word.  $\square$

We now turn, in the following lemmas, to consider connections between successive expanded Lyndon words,  $L_i^{r_i}$  and  $L_{i+1}^{r_{i+1}}$ .

**Lemma 7.** *Let  $L_i^{r_i} = w\sigma(k-1)^t \in [k]^n$  be the  $i$ th expanded Lyndon word in increasing lexicographical order where  $\sigma \neq k-1$ . Then, the  $(i+1)$ th expanded Lyndon word,  $L_{i+1}^{r_{i+1}}$ , is  $w(\sigma+1)x$  where  $x \in [k]^t$  is the lexicographically smallest word for which  $w(\sigma+1)x$  is an expanded Lyndon word.*

*Proof.* By Lemma 6,  $w(\sigma+1)(k-1)^t$  is an expanded Lyndon word, i.e.,  $w(\sigma+1)(k-1)^t = L_j^{r_j}$  for some  $j > i$ . It then follows that  $L_{i+1}^{r_{i+1}}$  must be of the form  $w(\sigma+1)x$  as claimed.  $\square$

The following lemma combines the shift rule function, next, and the lexicographical order of expanded Lyndon words.

**Lemma 8.**  $\text{next}^{|L_i|}(L_i^{r_i}) = L_{i+1}^{r_{i+1}}$ , for all  $i \in [Z_k(n) - 2]$ .

*Proof.* Since  $i \in [Z_k(n) - 2]$ ,  $L_i^{r_i}$  is not the lexicographically last expanded Lyndon word and not the one before it, i.e.,

$$L_i^{r_i} \neq (k-1)^n, \quad L_i^{r_i} \neq (k-2)(k-1)^{n-1}. \quad (2)$$

We can therefore write  $L_i = w\sigma(k-1)^t$ ,  $\sigma \in [k-1]$ , so  $L_i^{r_i} = w\sigma(k-1)^t L_i^{r_i-1}$ . Using these notations

$$\text{next}^{|L_i|}(L_i^{r_i}) = \text{next}^{|w|+1+t}(w\sigma(k-1)^t L_i^{r_i-1}).$$

Our proof proceeds by establishing the following three facts:

- (a)  $\text{next}^{|w|}(w\sigma(k-1)^t L_i^{r_i-1}) = \sigma(k-1)^t L_i^{r_i-1} w$
- (b)  $\text{next}(\sigma(k-1)^t L_i^{r_i-1} w) = (k-1)^t L_i^{r_i-1} w(\sigma+1)$
- (c)  $\text{next}^t((k-1)^t L_i^{r_i-1} w(\sigma+1)) = L_i^{r_i-1} w(\sigma+1)x$ , such that  $x \in [k]^t$  is the lexicographically smallest word for which  $L_i^{r_i-1} w(\sigma+1)x$  is an expanded Lyndon word.

Combining the three facts together, we get that  $\text{next}^{|L_i|}(L_i^{r_i}) = L_i^{r_i-1}w(\sigma+1)x$ , and by Lemma 7, we get the desired.

We first prove step (a). We contend that this step's claim holds since in the first  $|w|$  applications of  $\text{next}$  only the third case of the definition of  $\text{next}$  (cf. Definition 3) takes place. To prove this, we need to show that for any decomposition  $w = w_1\tau w_2$ ,  $\tau \in [k]$ ,  $w_1, w_2 \in [k]^*$ , we have

$$\text{next}(\tau w_2 \sigma(k-1)^t L_i^{r_i-1} w_1) = w_2 \sigma(k-1)^t L_i^{r_i-1} w_1 \tau.$$

Hence, we need to show that  $w_2 \sigma(k-1)^t L_i^{r_i-1} w_1 \tau$  is not a necklace head, and that if  $\tau = k-1$  then there is no  $\sigma' \in [k-1]$  such that  $w_2 \sigma(k-1)^t L_i^{r_i-1} w_1 \sigma'$  is a necklace head.

For the first condition, assume to the contrary that the predicate  $\text{head}(w_2 \sigma(k-1)^t L_i^{r_i-1} w_1 \tau)$  is true. By definition, there should exist an integer  $0 \leq m \leq |w_2|$  such that  $w_2 = (k-1)^m w_3$  and

$$\mathcal{R}^m(w_2 \sigma(k-1)^t L_i^{r_i-1} w_1 \tau) = w_3 \sigma(k-1)^t L_i^{r_i-1} w_1 \tau (k-1)^m$$

is an expanded Lyndon word. However, we note that

$$w_3 \sigma(k-1)^t L_i^{r_i-1} w_1 \tau (k-1)^m = \mathcal{R}^{|w_1|+1+m}(L_i^{r_i}).$$

Since  $0 < |w_1| + 1 + m < |L_i|$ , this contradicts the cyclic order of  $L_i^{r_i}$ .

As for the second condition, where  $\tau = k-1$ , assume to the contrary that for some  $\sigma' \in [k-1]$ , the string  $w_2 \sigma(k-1)^t L_i^{r_i-1} w_1 \sigma'$  is a necklace head. Again, there should exist an integer  $0 \leq m \leq |w_2|$ , such that  $w_2 = (k-1)^m w_3$ , and

$$\mathcal{R}^m(w_2 \sigma(k-1)^t L_i^{r_i-1} w_1 \sigma') = w_3 \sigma(k-1)^t L_i^{r_i-1} w_1 \sigma' (k-1)^m \quad (3)$$

is an expanded Lyndon word. Thus, on the right-hand side of (3), the rightmost letter that is not  $k-1$ , is  $\sigma'$ . By repeated applications of Lemma 6, we get that we can replace  $\sigma'$  by  $k-1$  and still have an expanded Lyndon word, i.e.,

$$w_3 \sigma(k-1)^t L_i^{r_i-1} w_1 (k-1)^{m+1} = \mathcal{R}^{|w_1|+1+m}(L_i^{r_i})$$

is an expanded Lyndon word. As in the previous case, this contradicts the cyclic order of  $L_i$ .

The proof of step (b) is simpler. We need to show that we fall under the first case in the definition of next (cf. Definition 3), i.e., that  $(k-1)^t L_i^{r_i-1} w \sigma$  is a necklace head. That is indeed true since

$$\mathcal{R}^t((k-1)^t L_i^{r_i-1} w \sigma) = L_i^{r_i-1} w \sigma (k-1)^t = L_i^{r_i}$$

is an expanded Lyndon word.

Finally, we address step (c), where we need to prove that  $\text{next}^t((k-1)^t L_i^{r_i-1} w(\sigma+1)) = L_i^{r_i-1} w(\sigma+1)x$ , such that  $x \in [k]^t$  is the lexicographically smallest word for which  $L_i^{r_i-1} w(\sigma+1)x$  is an expanded Lyndon word. Note that by (2),  $(k-1)^t L_i^{r_i-1} w(\sigma+1) \neq (k-1)^n$ , so by Lemma 7 such an  $x$  exists. Additionally, for any  $0 \leq i < t$  we have that  $\text{next}^i((k-1)^t L_i^{r_i-1} w(\sigma+1)) = (k-1)^i w'$ , thus we never fall within the first case of next.

Next, we show that for any  $0 \leq i < t$ ,  $j = t - i - 1$ ,  $x = x_1 \tau x_2$ ,  $x_1 \in [k]^i$ ,  $\tau \in [k]$ , we have that

$$\text{next}((k-1)^{j+1} L_i^{r_i-1} w(\sigma+1)x_1) = (k-1)^j L_i^{r_i-1} w(\sigma+1)x_1 \tau.$$

We distinguish between two cases depending on  $\tau$ . For the first case, let  $\tau = k-1$ . We contend that we do not fall within the second case of next. Assume to the contrary that there is some  $\sigma' \in [k-1]$  such that  $(k-1)^j L_i^{r_i-1} w(\sigma+1)x_1 \sigma'$  is a necklace head. Thus,  $L_i^{r_i-1} w(\sigma+1)x_1 \sigma' (k-1)^j$  is an expanded Lyndon word. Looking at its suffix of length  $t$ , we get

$$x_1 \sigma' (k-1)^j < x_1 (k-1) x_2 = x_1 \tau x_2 = x,$$

which is a contradiction to the minimality of  $x$ .

Now, for the case where  $\tau \in [k-1]$ . By the definition of  $x$  we know that  $L_i^{r_i-1} w(\sigma+1)x = L_i^{r_i-1} w(\sigma+1)x_1 \tau x_2$  is an expanded Lyndon word. Using Lemma 6, we get that  $L_i^{r_i-1} w(\sigma+1)x_1 \tau (k-1)^j$  is also an expanded Lyndon word. Therefore,  $(k-1)^j L_i^{r_i-1} w(\sigma+1)x_1 \tau$  is a necklace head. Left to be shown is that

$$\tau = \tau_{\min} \triangleq \min \left\{ \tau' \in [k] : \text{head}((k-1)^j L_i^{r_i-1} w(\sigma+1)x_1 \tau') \right\}.$$

Assuming to the contrary that  $\tau_{\min} < \tau$ , then  $(k-1)^j L_i^{r_i-1} w(\sigma+1)x_1 \tau_{\min}$  is a necklace head, implying that  $L_i^{r_i-1} w(\sigma+1)x_1 \tau_{\min} (k-1)^j$  is an expanded Lyndon word. As in the previous case, since

$$x_1 \tau_{\min} (k-1)^j < x_1 \tau x_2 = x,$$



we get a contradiction to the minimality of  $x$ .  $\square$

Lemma 8 does not apply to the penultimate expanded Lyndon word, for which, by simple inspection of the definition of  $\text{next}$  we state

$$\text{next}(L_{Z_k(n)-2}^{r_{Z_k(n)-2}}) = \text{next}((k-2)(k-1)^{n-1}) = (k-1)^n. \quad (4)$$

We are now in a position to prove the main result.

*Proof of Theorem 4.* As a first technical step it is easy to verify that  $\text{next}$  is a shift rule generating some sequence, since indeed for every  $\sigma w$ ,  $\sigma \in [k]$ ,  $w \in [k]^{n-1}$ , we have  $\text{next}(\sigma w) = w\sigma'$  for some  $\sigma' \in [k]$ .

In the next step, let us examine an unknown sequence  $\langle v_i \rangle_{i=0}^{k^n-1}$ , that is initialized with  $v_0 \dots v_{n-1} = 0^n$ , and whose following symbols are generated using  $\text{next}$ . We define the numbers  $s_i \triangleq \sum_{j=0}^{i-1} |L_j|$ , for all  $0 \leq i \leq Z_k(n) - 1$ . We prove by induction that for all  $i \in [Z_k(n) - 1]$ ,  $v_{s_i} v_{s_i+1} \dots v_{s_i+n-1} = L_i^{r_i}$ . The proof is immediate since the induction base is our initialization of  $v_0 \dots v_{n-1} = 0^n = L_0^{r_0}$ , and the induction step is provided by Lemma 8, since

$$v_{s_{i+1}} \dots v_{s_{i+1}+n-1} = \text{next}^{|L_i|}(v_{s_i} \dots v_{s_i+n-1}) = \text{next}^{|L_i|}(L_i^{r_i}) = L_{i+1}^{r_{i+1}}.$$

By this induction, we already have the prefix of the generated sequence to be  $L_0 L_1 \dots L_{Z_k(n)-2}$ , but we are missing the last Lyndon word,  $L_{Z_k(n)-1} = k - 1$ . This is easily taken care of, since by (4),

$$\begin{aligned} v_{s_{Z_k(n)-2}+1} \dots v_{s_{Z_k(n)-2}+n} &= \text{next}(v_{s_{Z_k(n)-2}} \dots v_{s_{Z_k(n)-2}+n-1}) \\ &= \text{next}(L_{Z_k(n)-2}^{r_{Z_k(n)-2}}) = \text{next}((k-2)(k-1)^{n-1}) \\ &= (k-1)^n, \end{aligned}$$

namely, the last symbol is the last Lyndon word,

$$v_{s_{Z_k(n)-2}+n} = v_{s_{Z_k(n)-1}} = v_{k^n-1} = k - 1 = L_{Z_k(n)-1}.$$

We also observe that the shift rule wraps around the end of the sequence. Indeed, by a simple inspection of Definition 3, for every  $1 \leq i \leq n$ ,

$$\begin{aligned} \text{next}(v_{k^n-i} \dots v_{k^n-1} v_0 \dots v_{n-1-i}) &= \text{next}((k-1)^i 0^{n-i}) \\ &= (k-1)^{i-1} 0^{n-i+1} \\ &= v_{k^n-i+1} \dots v_{k^n-1} v_0 \dots v_{n-i}. \end{aligned}$$

As the final step in the proof, by FKM [8, 9] this sequence is exactly the prefer-min  $(n, k)$ -DB sequence.  $\square$

We conclude this section by reminding the reader that in order to generate the prefer-max  $(n, k)$ -DB sequence (instead of the prefer-min one), all that is required is to start the FSR with  $(k - 1)^n$ , and to use the shift rule  $\psi^{-1} \circ \text{next} \circ \psi$ , where  $\psi$  is the complement function defined in Section 2, and  $\circ$  denotes function composition.

#### 4. Efficient Shift-Rule Algorithm

Algorithms for implementing shift-rules for the prefer-min (or prefer-max)  $(n, k)$ -DB sequences are known [12, 5], however, they are highly inefficient. The main result of this section is an efficient algorithm that implements the shift rule we presented in the previous section. By quick inspection, the claim hinges on an efficient implementation of the head predicate, as well as finding  $\min S$  in the second case of next.

Our algorithm uses two key components. The first, is the renowned factorization due to Chen, Fox, and Lyndon [1], namely, that every word  $w \in [k]^+$  has a unique decomposition  $w = w_0 w_1 \dots w_{m-1}$ , such that  $w_i$  is a Lyndon word for all  $0 \leq i \leq m - 1$ , and  $w_0 \geq w_1 \geq \dots \geq w_{m-1}$ . We shall call this the CFL factorization of  $w$ . The second key component is due to Duval [3], who showed that this unique decomposition may be computed for all  $w \in [k]^+$  in  $O(|w|)$  time and memory.

The algorithm we propose for computing next is the following:

---

**Algorithm 1**  $\text{next}(\sigma w)$ .

---

```

1: if  $((\sigma < k - 1) \wedge \text{head}(w\sigma))$  then
2:   return  $w(\sigma + 1)$ 
3: else if  $\sigma w = (k - 1)^n$  then
4:   return  $(k - 1)^{n-1}0$ 
5: else if  $((\sigma = k - 1) \wedge \text{head}(w(k - 2)))$  then
6:   let  $t \in \mathbb{Z}$ ,  $w' \in [k]^*$  be such that  $w = (k - 1)^t w'$ , and  $w' \notin (k - 1)[k]^*$ 
7:   let  $w' = w'_0 \cdots w'_{m-1}$  be the CFL factorization of  $w'$ 
8:    $w'_m \leftarrow \varepsilon$ 
9:    $s \leftarrow \min \{1 \leq i \leq m : w'_i \text{ is a (possibly empty) proper prefix of } w'_0\}$ 
10:   $p \leftarrow |w'_s w'_{s+1} \cdots w'_m|$ 
11:   $\sigma' \leftarrow w'_{0,p}$ 
12:  if  $\text{head}(w\sigma')$  then
13:    return  $w\sigma'$ 
14:  else
15:    return  $w(\sigma' + 1)$ 
16:  end if
17: else
18:   return  $w\sigma$ 
19: end if

```

---

We comment that in line 11 of the algorithm,  $w'_{0,p}$  denotes the  $p$ th letter of  $w'_0$ , where letter indexing starts from 0.

The main theorem of the section is the following.

**Theorem 9.** *Algorithm 1 correctly computes the shift rule  $\text{next}$  from Definition 3 in  $O(n)$  time and memory.*

In order to prove this theorem we require some auxiliary lemmas. First, we address the efficiency of computing the predicate  $\text{head}$ .

**Lemma 10.** *For any  $w \in [k]^n$  it is possible to compute  $\text{head}(w)$  in  $O(n)$  time and memory.*

*Proof.* Let  $i \in \mathbb{Z}$  be the largest integer such that  $(k - 1)^i$  is a prefix of  $w$ . We apply Duval's algorithm [3] to  $\mathcal{R}^i(w)$  to obtain its CFL factorization  $\mathcal{R}^i(w) = w_0 w_1 \cdots w_{m-1}$ . Then  $\text{head}(w)$  is true if and only if  $w_0 = w_{m-1}$ .  $\square$

We continue with three lemmas that concern the CFL factorization of a certain prefix of expanded Lyndon words.

**Lemma 11.** *Let  $v = w(k-1)^t \in [k]^n$  be an expanded Lyndon word,  $w \in [k]^*[k-1]$ , and let  $w_0 \cdots w_{m-1} = w$  be the CFL factorization of  $w$ . Then  $w_i$  is a prefix of  $w_0$ , for all  $0 \leq i \leq m-1$ .*

*Proof.* The claim holds trivially for  $w_0$ . Assume to the contrary that  $w_i$ , for some  $1 \leq i \leq m-1$ , is not a prefix of  $w_0$  (and in particular, they are not equal). Due to the CFL factorization,  $w_0 > w_i$ . Since both  $w_0$  and  $w_i$  are not empty, we can write  $w_0 = x\tau y$ , and  $w_i = x\tau'z$ , with  $x, y, z \in [k]^*$  and  $\tau, \tau' \in [k]$ ,  $\tau > \tau'$ . Thus,  $\mathcal{R}^{|w_0|+\cdots+|w_{i-1}|}(v)$ , that starts with  $x\tau'$ , is lexicographically smaller than  $v$ , that starts with  $x\tau$ . This, by Lemma 5, contradicts the assumption that  $v$  is an expanded Lyndon word.  $\square$

**Lemma 12.** *Let  $v = w(k-1)^t \in [k]^n$  be an expanded Lyndon word,  $w \in [k]^*[k-1]$ , and let  $w = w_0 \cdots w_{m-1}$  be the CFL factorization of  $w$ . Then, for any  $1 \leq i \leq m-1$ ,  $w_i \dots w_{m-1}$  is a proper prefix of  $w$ .*

*Proof.* Let us fix any  $1 \leq i \leq m-1$ . We first prove by induction on  $j \leq m-1$  that  $w_i \dots w_j$  is a prefix of  $w$ . The base case holds by Lemma 11. For the induction step, assume that  $w_i \dots w_j$  is a prefix of  $w$ , and we shall prove that the same holds for  $w_i \dots w_{j+1}$ .

Let  $w_i \dots w_j x$  be a prefix of  $w$ , where  $x \in [k]^*$ ,  $|x| = |w_{j+1}|$ . If  $w_{j+1} < x$ , then

$$\begin{aligned} \mathcal{R}^{|w_0|+\cdots+|w_{i-1}|}(v) &= w_i \dots w_{m-1} \sigma(k-1)^t w_0 \dots w_{i-1} \\ &< w_0 \dots w_{m-1} \sigma(k-1)^t = v, \end{aligned}$$

which contradicts Lemma 5. Thus,  $w_{j+1} \geq x$ . Recall, by Lemma 11,  $w_{j+1}$  is a prefix of  $w_0$ . If  $w_{j+1} > x$ , then  $x$  is strictly smaller than a prefix of  $w_0$  of the same length, which, again, contradicts Lemma 5 in a similar manner.

We conclude that  $w_{j+1} = x$ , and hence,  $w_i \dots w_{j+1}$  is a prefix of  $w$ . Therefore, by induction  $w_i \dots w_{m-1}$  is a prefix of  $w$ . Finally, since  $i \geq 1$  and  $w_0 \neq \varepsilon$ ,  $w_i \dots w_m$  is a proper prefix of  $w$ .  $\square$

**Lemma 13.** *Let  $v = w(k-1)^t \in [k]^n$  be an expanded Lyndon word,  $w \in [k]^*[k-1]$ , and let  $w = w_0 \dots w_{m-1}$  be the CFL factorization of  $w$ . If, for some  $1 \leq i \leq m-1$ ,  $w_i$  is a proper prefix of  $w_0$ , then  $w_i \dots w_{m-1}$  is a proper prefix of  $w_0$ .*

*Proof.* Assume to the contrary that  $w_i \dots w_{m-1}$  is not a proper prefix of  $w_0$ . By Lemma 12,  $w_i \dots w_{m-1}$  is a prefix of  $w$ . Therefore, we get that  $w_0$  is a prefix of

$w_i \dots w_{m-1}$ . Let us then write  $w_0 = w_i \dots w_j u$ , where  $u \in [k]^+$  is a non-empty prefix of  $w_{j+1}$ . Hence,  $u \leq w_{j+1}$ . However,  $u$  is a non-empty proper suffix of  $w_0$ , and since  $w_0$  is a Lyndon word, by [11, Prop. 5.1.2],  $w_0 < u$ . Therefore,  $w_0 < u \leq w_{j+1}$  which contradicts the properties of the CFL factorization.  $\square$

We are now ready to prove the main theorem.

*Proof of Theorem 9.* The time and memory complexity of Algorithm 1 follow from Lemma 10 and Duval's algorithm [3]. The rest of the proof is concerned with the correctness of the algorithm.

We argue that Algorithm 1 computes the function next. We consider the three cases of Definition 3 separately. First, if  $\sigma \in [k-1]$  and  $\text{head}(w\sigma)$ , the algorithm returns  $w(\sigma+1)$  in line 2 as required by the first case of Definition 3.

Now, assume the input  $\sigma w$  falls within the third case of Definition 3. If  $\sigma < k-1$  the claim is obvious as the condition in line 1 does not hold. If  $\sigma = k-1$ , then we have  $S \triangleq \{\sigma'' \in [k-1] : \text{head}(w\sigma'')\}$ . By Lemma 6,  $S \neq \emptyset$  if and only if  $\text{head}(w(k-2))$  holds. Thus, line 1 correctly checks whether the second case of Definition 3 applies. We therefore reach line 18 exactly when the third case of Definition 3 applies, and correctly return  $w\sigma$ .

We are left with the second case of Definition 3, where  $\sigma = k-1$  and  $S \neq \emptyset$ . First, the special case of  $\sigma w = (k-1)^n$ , is handled correctly in line 4. Otherwise,  $w$  contains some letter other than  $k-1$ , and  $w'$  is well defined.

Consider the CFL factorization,  $w' = w'_0 w'_1 \dots w'_{m-1}$ , and define the auxiliary  $w'_m \triangleq \varepsilon$ . Additionally, define  $u \triangleq w'_i \dots w'_m$ , such that  $w'_i$  is the first (possibly empty) proper prefix of  $w'_0$ , and define  $p \triangleq |u|$ . Finally, we set  $\sigma' \triangleq w'_{0,p}$  to be the  $p$ -th letter of  $w'_0$ . Note that all these definitions agree with the algorithm.

We contend that

$$\sigma' \leq \min S \leq \sigma' + 1, \quad (5)$$

and thus lines 12-16 of the algorithm provide the correct answer.

We start by proving the lower bound of (5). If  $\sigma' = 0$  we are done. Otherwise,  $\sigma' > 0$  and we shall prove that  $w'(\sigma'-1)(k-1)^t$  is not an expanded Lyndon word. Using Lemma 13, we can conclude that  $u\sigma'$  is a prefix of  $w'_0$ , and therefore also a prefix of  $w'(\sigma'-1)(k-1)^t$ . Additionally, note that  $u(\sigma'-1)$  is a subword of  $w'(\sigma'-1)(k-1)^t$ . Therefore,  $w'(\sigma'-1)(k-1)^t$  is strictly greater than one of its rotations. By Lemma 5, we conclude  $w'(\sigma'-1)(k-1)^t$  is not an expanded Lyndon word.

We turn to prove the upper bound of (5). By line 5 of the algorithm, we already know that  $k-2 \in S$ , hence, if  $\sigma' \geq k-3$  we have nothing further to

prove. Otherwise,  $\sigma' \leq k - 4$  and we shall prove that  $w'(\sigma' + 1)(k - 1)^t$  is an expanded Lyndon word. In fact, we prove a stronger property, namely, that  $w'(\sigma' + 1)(k - 1)^t$  is a Lyndon word. We accomplish this by taking any partition  $w'(\sigma' + 1)(k - 1)^t = xy$ ,  $x, y \in [k]^+$ , and proving that  $xy < yx$ .

First, the claim is trivial if  $y$  is a suffix of  $(k - 1)^t$ . In addition, if  $y = (\sigma' + 1)(k - 1)^t$ , we note that  $w'$  starts with a letter  $\tau < (\sigma' + 1)$ . This holds since  $w'(k - 2)(k - 1)^t$  is an expanded Lyndon word and  $\sigma'$  appears somewhere in it, hence, by Lemma 5, we conclude that  $w'$  starts with a letter  $\tau \leq \sigma'$ . It is left to prove the case where  $y = zw'_{j+1} \dots w'_{m-1}(\sigma' + 1)(k - 1)^t$ , where  $z$  is a nonempty suffix of  $w'_j$ . We consider two cases, depending on whether  $z = w'_j$ .

If  $z \neq w'_j$ , i.e.,  $z$  is a non-empty proper suffix of  $w'_j$ , then by [11, Prop. 5.1.2],  $w'_j < z$ . Hence,  $z > v$  where  $v$  is the prefix of  $w'_j$  of length  $|z|$ . However, by Lemma 11,  $v$  is also a prefix of  $w'_0$ . Thus we get that the prefix of  $w'(\sigma' + 1)(k - 1)^t$  of length  $|y|$  is strictly smaller than  $y$ , since the former starts with  $v$ , while the latter starts with  $z > v$ , and so

$$w'(\sigma' + 1)(k - 1)^t = xy < yx,$$

as desired.

If  $z = w'_j$  then we may write  $y = w'_j w'_{j+1} \dots w'_{m-1}(\sigma' + 1)(k - 1)^t$ . Recall that  $i$  is defined as the smallest integer such that  $w'_i$  is a (possibly empty) proper prefix of  $w'_0$ . First, consider the case of  $j \geq i$ . By Lemma 13,  $w'_i \dots w'_{m-1}$  is a proper prefix of  $w'_0$ . By definition, within  $w'_0$ , the prefix  $w'_i \dots w'_{m-1}$  is followed by the letter  $\sigma'$ . Thus,  $w'_j \dots w'_{m-1} \sigma'$  is a subword of  $w'_0$ . Let  $v$  be the prefix of  $w'_0$  of length  $|w'_j \dots w'_{m-1} \sigma'|$ . Since  $w'_0$  is a Lyndon word,  $v \leq w'_j \dots w'_{m-1} \sigma'$ . Therefore,  $v < w'_j \dots w'_{m-1}(\sigma' + 1)$ , and as a consequence, the prefix of  $w'(\sigma' + 1)(k - 1)^t$  of length  $|y|$  is strictly smaller than  $y$ . Thus, again,

$$w'(\sigma' + 1)(k - 1)^t = xy < yx,$$

as desired.

The final case left to consider is the case of  $1 < j < i$  (note that  $j > 1$  since  $x \neq \varepsilon$ ). By Lemma 11 and by the choice of  $i$  as the first proper prefix of  $w'_0$ ,  $w'_0 = w'_1 = \dots = w'_j = \dots = w'_{i-1}$ . Hence, we can write:

$$y = (w'_0)^r w'_i \dots w'_{m-1}(\sigma' + 1)(k - 1)^t,$$

and

$$xy = w'(\sigma' + 1)(k - 1)^t = (w'_0)^{r'+r} w'_i \dots w'_{m-1}(\sigma' + 1)(k - 1)^t,$$

where  $r' \geq 1$  and  $r' + r = i$ . Notice now that  $w'_i \dots w'_{m-1} \sigma'$  is a prefix of  $w'_0$ , and therefore  $(w'_0)^r w'_i \dots w'_{m-1} \sigma'$  is a prefix of  $(w'_0)^{r+1} = w'_0 w'_1 \dots w'_r$ , hence also a prefix of  $w'_0 \dots w'_{m-1} (\sigma' + 1)(k - 1)^t$ . In addition,  $(w'_0)^r w'_i \dots w'_{m-1} (\sigma' + 1)$  is a prefix of  $y$ .

Comparing the prefixes of  $xy$  and  $yx$ , we have

$$(w'_0)^r w'_i \dots w'_{m-1} \sigma' < (w'_0)^r w'_i \dots w'_{m-1} (\sigma' + 1),$$

and it follows that

$$w'(\sigma' + 1)(k - 1)^t = xy < yx,$$

which completes the proof.  $\square$

## 5. Discussion

In this paper we studied the well known prefer-min and prefer-max  $(n, k)$ -DB sequences. We completed a gap in the literature by presenting a shift-rule for the sequences, as well as an efficient algorithm computing this shift rule. The algorithm receives as input a sub-sequence of  $n$  symbols, and determines the next symbol in  $O(n)$  time and memory.

The shift rule we presented may be seen as an extension to the binary shift rule presented in [6]. Indeed, if we set  $k = 2$  in our algorithm, the second case of Definition 3 becomes degenerate, we are left with the algorithm of [6]. This also explains the main difficulty in our solution, which is finding  $\min S$  efficiently. The crux of solving this difficulty is the proof that we only need to choose between two carefully chosen values.

## References

- [1] K. T. Chen, R. H. Fox, R. C. Lyndon, Free differential calculus, IV, *Annals of Math.* 68 (1958) 81–95.
- [2] N. G. de Bruijn, A combinatorial problem, *Koninklijke Nederlandse Akademie v. Wetenschappen* 49 (1946) 758–764.
- [3] J. P. Duval, Factorizing words over an ordered alphabet, *J. of Algorithms* 4 (1983) 363–381.
- [4] C. Flye-Sainte Marie, Solution to problem number 58, *L'Intermédiaire des Mathématiciens* (1894) 107–110.

- [5] L. R. Ford, A cyclic arrangement of  $m$ -tuples, Tech. Rep. P-1071, RAND Corp. (1957).
- [6] H. M. Fredricksen, Generation of the Ford sequence of length  $2^n$ ,  $n$  large, J. Combin. Theory 12 (1972) 153–154.
- [7] H. M. Fredricksen, A survey of full length nonlinear shift register cycle algorithms, SIAM Rev. 24 (2) (1982) 195–221.
- [8] H. M. Fredricksen, I. J. Kessler, Lexicographic compositions and De Bruijn sequences, J. Combin. Theory 22 (1977) 17–30.
- [9] H. M. Fredricksen, J. Maiorana, Necklaces of beads in  $k$  colors and  $k$ -ary De Bruijn sequences, Discrete Math. 23 (1978) 207–210.
- [10] D. E. Knuth, The Art of Computer Programming Volume 4a: Combinatorial Algorithms, Addison Wesley, 2011.
- [11] M. Lothaire, Combinatorics on Words, Cambridge University Press, 1997.
- [12] M. H. Martin, A problem in arrangements, Bull. Amer. Math. Soc. 40 (1934) 859–864.
- [13] A. Ralston, A new memoryless algorithm for De Bruijn sequences, J. of Algorithms 2 (1) (1981) 50–62.
- [14] F. Ruskey, C. Savage, T. M. Y. Wang, Generating necklaces, J. of Algorithms 13 (3) (1992) 414–430.
- [15] J. Sawada, A. Williams, D. Wong, A surprisingly simple De Bruijn sequence construction, Discrete Math. 339 (2016) 127–131.
- [16] T. van Aardenne-Ehrenfest, N. G. de Bruijn, Circuits and trees in oriented linear graphs, Simon Stevin: Wis- en Natuurkundig Tijdschrift 28 (1951) 203–217.
- [17] J. H. van Lint, R. M. Wilson, A Course in Combinatorics, 2nd Edition, Cambridge Univ. Press, 2001.