Discrete Mathematics 339 (2016) 127–131

Contents lists available at ScienceDirect

Discrete Mathematics

journal homepage: www.elsevier.com/locate/disc

A surprisingly simple de Bruijn sequence construction

Joe Sawada [a], Aaron Williams [b], Dennis Wong [c]

[a] School of Computer Science, University of Guelph, Canada

[b] Division of Science, Mathematics, and Computing, Bard College at Simon's Rock, USA

[c] Department of Mathematics, Computer Science and Information Systems, Northwest Missouri State University, USA

ARTICLE INFO

Article history:
Received 27 September 2014
Received in revised form 2 August 2015
Accepted 3 August 2015
Available online 27 August 2015

Keywords:
De Bruijn sequence
Universal cycle
Necklaces
Shift Gray code
CAT algorithm
Generate
Shift rule
Successor rule

ABSTRACT

Pick any length n binary string $b_1b_2\cdots b_n$ and remove the first bit b_1. If $b_2b_3\cdots b_n1$ is a necklace, then append the complement of b_1 to the end of the remaining string; otherwise append b_1. By repeating this process, eventually all 2^n binary strings will be visited cyclically. This shift rule leads to a new de Bruijn sequence construction that can be generated in $O(1)$-amortized time per bit.

1. A new de Bruijn sequence construction

A *de Bruijn sequence* of order n is a cyclic sequence of length 2^n where each substring of length n is a unique binary string. As an example, the cyclic sequence 0000100110101111 of length 16 is a de Bruijn sequence for $n = 4$. The 16 unique substrings of length 4 when considered cyclically are:

0000, 0001, 0010, 0100, 1001, 0011, 0110, 1101, 1010, 0101, 1011, 0111, 1111, 1110, 1100, 1000.

As illustrated in this example, a de Bruijn sequence of order n induces a very specific type of cyclic order of the length n binary strings: the length $n-1$ suffix of a given binary string is the same as the length $n-1$ prefix of the next string in the ordering.

The number of unique de Bruijn sequences for a given n is $2^{2^{n-1}-n}$ [3]; however, only a few efficient constructions are known. In particular, there are

- a shift generation approach based on primitive polynomials by Golomb [9],
- three different algorithms to generate the lexicographically smallest de Bruijn sequence (also known as the Ford sequence): a Lyndon word concatenation algorithm by Fredricksen and Maiorana [8], a successor rule approach by Fredricksen [5], and a block concatenation algorithm by Ralston [12],
- a lexicographic composition concatenation algorithm by Fredricksen and Kessler [7],
- three different pure cycle concatenation algorithms by Fredricksen [6], Etzion and Lempel [4], and Huang [10] respectively, and

E-mail addresses: jsawada@uoguelph.ca (J. Sawada), haron@uvic.ca (A. Williams), cwong@uoguelph.ca (D. Wong).

http://dx.doi.org/10.1016/j.disc.2015.08.002
0012-365X/

▷ cool-lex based constructions by Sawada, Stevens and Williams [13] and Sawada, Williams and Wong [14].

Each algorithm requires only $O(n)$ space and generates their de Bruijn sequences in $O(n)$ time per bit, except the pure cycle concatenation algorithm by Etzion and Lempel which requires $O(n^2)$ space. The Lyndon word concatenation algorithm and the cool-lex based approaches achieve $O(1)$-amortized time per bit. There also exist interesting greedy constructions including the "prefer-1" and "prefer-opposite" approaches by Martin [11] and Alhakim [1]; however, they require $\Omega(2^n)$ space. Finding Euler cycles in the de Bruijn graph is an approach that will find all de Bruijn sequences for a given n, but again, storing the graph requires $\Omega(2^n)$ space.

In this paper, a novel shift-based construction for producing a new de Bruijn sequence is presented. It is based on testing whether or not a given string is a necklace. A *necklace* is the lexicographically smallest string in an equivalence class of strings under rotation. The new construction is based on the following function over binary strings, where $\bar{b}$ denotes the complement of the bit b:

$$f(b_1b_2\cdots b_n) = \begin{cases} b_2b_3\cdots b_n\overline{b_1} & \text{if } b_2b_3\cdots b_n1 \text{ is a necklace;} \\ b_2b_3\cdots b_nb_1 & \text{otherwise.} \end{cases}$$

As an illustration, successive applications of this rule for $n = 5$ starting with the string 00000 (the underlined strings will be discussed later) produce the following listing:

00000, <u>00001</u>, 00011, 00111, 01111, 11111, 11110, 11101,
11011, 10111, 01110, 11100, 11001, 10011, 00110, 01100,
11000, 10001, <u>00010</u>, 00101, 01011, 10110, 01101, 11010,
10101, 01010, 10100, 01001, 10010, <u>00100</u>, <u>01000</u>, <u>10000</u>.

Observe that every binary string of length 5 is visited exactly once and that by applying one more application of the rule, we return to the first string 00000. This property holds in general for all $n > 1$. This leads to the following theorem, where $\mathbf{B}(n)$ denotes the set of binary strings with length n.

Theorem 1. *The shift rule f induces a cyclic ordering on $\mathbf{B}(n)$.*

Before proving this theorem, observe that a de Bruijn sequence results from concatenating the first bit of each string in the exhaustive listing for $\mathbf{B}(n)$ produced by repeatedly applying f. We denote this de Bruijn sequence by $\mathbf{dB}(n)$. As an example, $\mathbf{dB}(5)$ is:

$$00000111110111\mathbf{1}00110001011010100 1.$$

The above sequence is different from all other known constructions. In particular, a reversed rotation of the well known Ford sequence (the lexicographically smallest de Bruijn sequence) differs at the 14th bit:

$$00000111110111\mathbf{0}10111001010011000 1.$$

Since a membership tester for necklaces can be implemented in $O(n)$ time [2], $\mathbf{dB}(n)$ can be generated in $O(n)$ time per bit. However, by studying the properties of this de Bruijn sequence, a slightly more sophisticated approach will generate the sequence in $O(1)$-amortized time per bit.

The rest of the paper is outlined as follows. In Section 2, we present some definitions and notation used later in the paper. In Section 3, we prove Theorem 1, which leads to a new de Bruijn sequence construction. Then in Section 4, we present an algorithm that produces this de Bruijn sequence in $O(1)$-amortized time per bit.

The main results of this paper are also found in Wong's Ph.D. thesis [15].

2. Definitions and notation

Consider a binary string $\alpha = b_1b_2\cdots b_n$. The string α is said to be *periodic* if there exists some shorter string β such that $\alpha = \beta^t$ for some $t > 1$, where the exponent t denotes the number of repeated concatenations. A string that is not periodic is *aperiodic*. The longest aperiodic prefix of α is denoted by $ap(\alpha)$.

A left rotation of α is $b_2b_3\cdots b_nb_1$ and is denoted by $\mathrm{LR}(\alpha)$. Let $\mathrm{LR}^r(\alpha)$ denote the string that results from applying a left rotation r times to α. Thus $\mathrm{LR}^r(\alpha) = b_{r+1}b_{r+2}\cdots b_nb_1b_2\cdots b_r$ when $0 \leq r < n$. The set of strings rotationally equivalent to α is denoted by $\mathbf{Rots}(\alpha)$, and the set of all length n binary necklaces is denoted by $\mathbf{N}(n)$.

A string β is *reachable* from α if β can be obtained from α by repeatedly applying the shift rule f.

3. Proof of Theorem 1

The proof for Theorem 1 is done in two steps. First, we show that the function f is a bijection. Then, we show that every string is reachable from 0^n by applying f.

Lemma 1. *The function f is a bijection.*

Proof. Since the domain and the range of f are the same, it suffices to show that f is onto. Consider a binary string $\beta = b_2 \cdots b_n b_1$. If $b_2 \cdots b_n 1$ is a necklace, then $f(\overline{b_1} b_2 \cdots b_n) = \beta$. Otherwise $f(b_1 b_2 \cdots b_n) = \beta$. □

Since f is a bijection, its inverse is well defined as follows:

$$f^{-1}(b_1 b_2 \cdots b_n) = \begin{cases} \overline{b_n} b_1 b_2 \cdots b_{n-1} & \text{if } b_1 b_2 \cdots b_{n-1} 1 \text{ is a necklace;} \\ b_n b_1 b_2 \cdots b_{n-1} & \text{otherwise.} \end{cases}$$

Now, revisit the example listing of $\mathbf{B}(5)$ given in Section 1. Observe that given any necklace $\alpha \in \mathbf{N}(5)$, the sequence of strings α, $\mathrm{LR}(\alpha)$, $\mathrm{LR}^2(\alpha)$, $\mathrm{LR}^3(\alpha)$ and $\mathrm{LR}^4(\alpha)$ appears as a subsequence in the listing. As an example, the underlined strings are $\mathbf{Rots}(00001)$ and appear in the order described. This property is the key to proving that all strings are reachable from 0^n.

Lemma 2. *Let $\alpha \in \mathbf{N}(n)$ and $\beta = b_1 b_2 \cdots b_n \in \mathbf{Rots}(\alpha)$. Then β is reachable from α.*

Proof. Apply induction on the number of 0s of α. In the base case, $\alpha = 1^n$ is the only string with zero 0s and the only string in $\mathbf{Rots}(\alpha)$ is 1^n. Inductively, assume that for $\alpha \in \mathbf{N}(n)$ with $0 \leq k < n$ 0s, each string $\beta \in \mathbf{Rots}(\alpha)$ is reachable from α. Consider $\alpha \in \mathbf{N}(n)$ with $k+1$ 0s. The set $\mathbf{Rots}(\alpha)$ contains all the left rotations of α. We now show by induction that $\mathrm{LR}^{r+1}(\alpha)$ is reachable from $\mathrm{LR}^r(\alpha)$ where $r = \{0, 1, \ldots, n-2\}$.

> In the base case, when $r = 0$, $\mathrm{LR}^0(\alpha) = \alpha$ and is reachable from α. Inductively, assume $\mathrm{LR}^t(\alpha)$ is reachable from α where $0 \leq t < n-1$. Consider $\mathrm{LR}^{t+1}(\alpha) = b_1 b_2 \cdots b_n$. If $b_1 b_2 \cdots b_{n-1} 1 \notin \mathbf{N}(n)$, then $f^{-1}(b_1 b_2 \cdots b_n) = b_2 b_3 \cdots b_n b_1 = \mathrm{LR}^t(\alpha)$. Otherwise, $b_n = 0$ and $f^{-1}(b_1 b_2 \cdots b_n) = 1 b_1 b_2 \cdots b_{n-1}$ because $b_1 b_2 \cdots b_n \notin \mathbf{N}(n)$ but $b_1 b_2 \cdots b_{n-1} 1 \in \mathbf{N}(n)$. Now observe that $b_1 b_2 \cdots b_{n-1} 1 \in \mathbf{N}(n)$ has k 0s and it is the necklace representative of $1 b_1 b_2 \cdots b_{n-1}$. Thus by the (external) inductive hypothesis, the string $1 b_1 b_2 \cdots b_{n-1}$ is reachable from $b_1 b_2 \cdots b_{n-1} 1$. Finally, $f^{-1}(b_1 b_2 \cdots b_{n-1} 1) = 0 b_1 b_2 \cdots b_{n-1} = \mathrm{LR}^t(\alpha)$, and thus $\mathrm{LR}^{t+1}(\alpha)$ is reachable from $\mathrm{LR}^t(\alpha)$.

Since $\mathrm{LR}^{r+1}(\alpha)$ is reachable from $\mathrm{LR}^r(\alpha)$, by transitivity, each $\beta \in \mathbf{Rots}(\alpha)$ is reachable from α. □

Lemma 3. *Each string $\alpha \in \mathbf{B}(n)$ is reachable from 0^n.*

Proof. Apply induction on the number of 1s of α. In the base case, the only string with zero 1s is 0^n, which is reachable from 0^n. Inductively, assume any string with $0 \leq k < n$ 1s is reachable from 0^n. Consider a string β with $k+1$ 1s and assume $\beta \in \mathbf{Rots}(\alpha)$ where $\alpha = b_1 b_2 \cdots b_n$ is a necklace. Note that $b_n = 1$ since α is a necklace with at least one 1. By Lemma 2 β is reachable from α. Clearly α is reachable from $\alpha' = f^{-1}(\alpha) = 0 b_1 b_2 \cdots b_{n-1}$. Since α' has k 1s, it is reachable from 0^n by the inductive assumption. Thus, by transitivity, β is reachable from 0^n. □

Together, Lemmas 1 and 3 prove Theorem 1.

4. Generating the de Bruijn sequence efficiently

As mentioned earlier, our de Bruijn sequence $\mathbf{dB}(n)$ is obtained by concatenating the first bit of each string in an exhaustive listing for $\mathbf{B}(n)$ produced by repeatedly applying f. Since a membership tester for necklaces can be implemented in $O(n)$ time [2], $\mathbf{dB}(n)$ can be generated in $O(n)$ time per bit. However, by studying the strings of the form $b_1 b_2 \cdots b_n$ such that $b_2 b_3 \cdots b_n 1$ is a necklace, we can improve the algorithm to run in $O(1)$-amortized time per bit.

In Table 1 we list the binary strings of length 6 obtained by starting from 000000 and successively applying the function f a total of $2^6 - 1$ times. Each row ends with a string $b'_1 b'_2 \cdots b'_n$ such that $b'_2 b'_3 \cdots b'_n 1$ is a necklace, and hence when the function f is applied to this final string, it will complement the final bit after rotation. This means that the first string $\alpha = b_1 b_2 \cdots b_n$ in each row has the property that $b_1 b_2 \cdots b_{n-1} 1$ is a necklace. Observe there are $2|\mathbf{N}(6)| - 2 = 2(14) - 2 = 26$ rows in this table. We will prove this observation for all n later in this section. The value $g(\alpha)$ corresponds to the number of strings in each row. Formally, $g(\alpha)$ is the number of successive applications of the function f starting from α until a bit gets complemented by the function f. Let $f^j(\alpha)$ be the string obtained from j successive applications of the shift rule f starting with $\alpha = b_1 b_2 \cdots b_n$. Recall that $\mathrm{LR}^j(\alpha) = b_{j+1} b_{j+2} \cdots b_n b_1 b_2 \cdots b_j$. Then $g(\alpha)$ corresponds to the value smallest value j such that $f^j(\alpha) \neq \mathrm{LR}^j(\alpha)$.

In the proof of the following lemma, the Kleene star operator b^* denotes 0 or more concatenations of b.

Table 1
The cyclic order of $\mathbf{B}(6)$ induced by the function f starting from 000000. The rows break down the order based on when f complements the last bit after a rotation. The value $g(\alpha)$ corresponds to the number of strings in each row, and **dB** is the concatenation of the first bits of the strings in each row. Concatenating together the strings in the column **dB** gives $\mathbf{dB}(6)$.

i	$\alpha, f(\alpha), f(f(\alpha)), \ldots$	$g(\alpha)$	**dB**
1	000000	1	0
2	000001	1	0
3	000011	1	0
4	000111	1	0
5	001111	1	0
6	011111	1	0
7	111111	1	1
8	111110, 111101, 111011, 110111, 101111	5	11111
9	011110, 111100, 111001, 110011, 100111	5	01111
10	001110, 011100, 111000, 110001, 100011	5	00111
11	000110	1	0
12	001101	1	0
13	011011, 110110, 101101	3	011
14	011010, 110100, 101001, 010011, 100110	5	01101
15	001100, 011000, 110000, 100001	4	0011
16	000010	1	0
17	000101	1	0
18	001011	1	0
19	010111, 101110, 011101, 111010, 110101, 101011	6	010111
20	010110, 101100, 011001, 110010, 100101	5	01011
21	001010	1	0
22	010101, 101010	2	01
23	010100, 101000, 010001, 100010	4	0101
24	000100	1	0
25	001001, 010010, 100100	3	001
26	001000, 010000, 100000	3	001

Lemma 4. *Let $\alpha = b_1 b_2 \cdots b_n$ and $b_1 b_2 \cdots b_{n-1} 1 \in \mathbf{N}(n)$, and g be the function on α which computes the minimum nonnegative value j such that $f^j(\alpha) \neq \mathrm{LR}^j(\alpha)$. Then:*

$$g(\alpha) = \begin{cases} 1 & \text{if } b_2 b_3 \cdots b_n 1 \in \mathbf{N}(n), \\ |ap(\alpha)| & \text{if } b_2 b_3 \cdots b_n 1 \notin \mathbf{N}(n) \text{ and } \alpha \in \mathbf{N}(n), \\ n - q & \text{if } b_2 b_3 \cdots b_n 1 \notin \mathbf{N}(n) \text{ and } \alpha \notin \mathbf{N}(n), \end{cases}$$

where q is the largest value such that α has suffix 0^q.

Proof. By the definition of f, clearly $g(\alpha) = 1$ when $b_2 b_3 \cdots b_n 1 \in \mathbf{N}(n)$. Otherwise consider two cases depending on whether or not α is a necklace.

Case 1: α is a necklace. If α is periodic with $p = |ap(\alpha)|$, then the substring $b_{p+1} b_{p+2} \cdots b_{2p}$ is lexicographically smaller than $b_i b_{i+1} \cdots b_p$ for all $i > 1$. Thus $f^j(\alpha) = \mathrm{LR}^j(\alpha)$ when $j < p$. Now consider $f^{p-1}(\alpha)$, observe that $f^{p-1}(\alpha) = 1 ap(\alpha)^{p-1} b_1 b_2 \cdots b_{p-1}$ and $ap(\alpha)^{p-1} b_1 b_2 \cdots b_{p-1} 1 = \alpha \in \mathbf{N}(n)$. Thus $f^p(\alpha) \neq \mathrm{LR}^p(\alpha)$ and $g(\alpha) = p = |ap(\alpha)|$. When α is aperiodic, let γ be the lexicographically least maximal substring of the form $0^* 1^*$ within α. Clearly $b_1 b_2 \cdots b_{|\gamma|} = \gamma$ because $\alpha \in \mathbf{N}(n)$. Also since $b_2 b_3 \cdots b_n 1 \notin \mathbf{N}(n)$, there exists some substring β within $b_{|\gamma|+1} b_{|\gamma|+2} \cdots b_n$ which is lexicographically smaller than $b_i b_{i+1} \cdots b_{|\gamma|+i-1}$ for all $1 < i < |\gamma|$. Thus $f^j(\alpha) = \mathrm{LR}^j(\alpha)$ when $j < |\gamma|$. Then notice that $f^{|\gamma|}(\alpha)$ contains the suffix γ which is strictly the lexicographically least maximal substring in α. Thus $f^j(\alpha) = \mathrm{LR}^j(\alpha)$ when $j < n$. Now observe that $f^{n-1}(\alpha) = b_n b_1 b_2 \cdots b_{n-1}$ and $b_1 b_2 \cdots b_{n-1} 1 = \alpha \in \mathbf{N}(n)$, thus $f^n(\alpha) \neq \mathrm{LR}^n(\alpha)$ and $g(\alpha) = n = |ap(\alpha)|$.

Case 2: α is not a necklace. Since $b_1 b_2 \cdots b_{n-1} 1 \in \mathbf{N}(n)$ but $\alpha \notin \mathbf{N}(n)$, α ends with the suffix 0^q with $q \geq 1$. Similar to the argument of Case 1, let γ be the lexicographically least maximal substring of the form $0^* 1^*$ within α. Clearly $b_1 b_2 \cdots b_{|\gamma|} = \gamma$ because $b_1 b_2 \cdots b_{n-1} 1 \in \mathbf{N}(n)$ and $b_n = 0$. The string $f^j(\alpha) = \mathrm{LR}^j(\alpha)$ when $j < |\gamma|$ since $b_{i+1} b_{i+2} \cdots b_{|\gamma|}$ is lexicographically larger than the suffix $0^q b_i$ for all $1 \leq i \leq |\gamma|$. Now consider $f^{|\gamma|}(\alpha)$; it contains the suffix $0^q \gamma$ which will be the unique lexicographically least maximal substring of the form $0^* 1^*$ in α (by definition of γ). Thus $f^j(\alpha) = \mathrm{LR}^j(\alpha)$ when $j < n - q$. Now observe that $f^{n-q-1}(\alpha) = b_{n-q} 0^q \gamma b_{|\gamma|+1} b_{|\gamma|+2} \cdots b_{n-q-1}$ and note that $0^q \gamma b_{|\gamma|+1} b_{|\gamma|+2} \cdots b_{n-q-1} 1$ is a necklace because it ends with 1 and as mentioned earlier $0^q \gamma$ is the unique lexicographically least maximal substring of the form $0^* 1^*$ within α. Thus $f^{n-q}(\alpha) \neq \mathrm{LR}^{n-q}(\alpha)$ and $g(\alpha) = n - q$. □

Still focusing on Table 1, note that the concatenation of the first bit of each string in each row is highlighted in the final column labeled **dB**. By concatenating all the strings together in this final column we obtain $\mathbf{dB}(6)$. Also observe that the strings in each row of Table 1 are obtained by repeatedly applying a left rotation starting from the initial string α. Thus, given the value $g(\alpha)$, we can output the string in the column **dB** in constant time per bit. This leads to an optimized algorithm to generate $\mathbf{dB}(n)$ given in Algorithm 1. A complete C implementation of the algorithm is given in Wong's Ph.D. thesis [15].

Algorithm 1 Optimized shift-based algorithm to generate **dB**(n) in $O(1)$-amortized time per bit.

```
1: procedure FastDeBruijn
2:     b_1 b_2 ··· b_n ← 0^n
3:     do
4:         j ← g(b_1 b_2 ··· b_n)
5:         Print(b_1 b_2 ··· b_j)
6:         b_1 b_2 ··· b_n ← f(b_j b_{j+1} ··· b_n b_1 b_2 ··· b_{j−1})
7:     while b_1 b_2 ··· b_n ≠ 0^n
```

4.1. Analysis

To analyze this optimized algorithm we first need to consider how often a bit gets complemented by the function f. Recall that **N**(n) denotes the set of binary necklaces of length n; we use $N(n)$ for the size of this set. It is well known that

$$N(n) = \frac{1}{n} \sum_{d|n} \phi(d)\, 2^{n/d},$$

where ϕ is Euler's totient function.

Lemma 5. *There are $2N(n) - 2$ binary strings of the form $b_1 b_2 \cdots b_n$ such that $b_2 b_3 \cdots b_n 1$ is a necklace.*

Proof. With the exception of the necklace of all zeros, every binary necklace of length n ends with 1. For each of these $N(n) - 1$ necklaces of the form $b_2 b_3 \cdots b_n 1$, we can assign b_1 to be either 0 or 1. □

Theorem 2. *The algorithm FastDeBruijn generates the de Bruijn sequence* **dB**(n) *in $O(1)$-amortized time per bit.*

Proof. The functions f and g can be computed in $O(n)$ time by applying a standard membership tester for necklaces [2] which can easily be modified to return $|ap(\alpha)|$. Thus, each iteration of the **do/while** loop requires $O(n)$ time. From Lemma 5 there are $2N(n) - 2$ iterations of the **do/while** loop. Thus, the overall running time will be proportional to $O(nN(n)) = \Theta(2^n)$. □

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments in improving the presentation of this paper. The first author's research was supported by NSERC grant 400673.

References

[1] A. Alhakim, A simple combinatorial algorithm for de Bruijn sequences, Amer. Math. Monthly 117 (8) (2010) 728–732.
[2] K.S. Booth, Lexicographically least circular substrings, Inform. Process. Lett. 10 (4/5) (1980) 240–242.
[3] N.G. de Bruijn, A combinatorial problem, Indag. Math. (N.S.) 49 (1946) 758–764.
[4] T. Etzion, A. Lempel, Algorithms for the generation of full-length shift-register sequences, IEEE Trans. Inform. Theory 30 (3) (1984) 480–484.
[5] H. Fredricksen, Generation of the Ford sequence of length 2^n, n large, J. Combin. Theory Ser. A 12 (1972) 153–154.
[6] H. Fredricksen, A class of nonlinear de Bruijn cycles, J. Combin. Theory Ser. A 19 (2) (1975) 192–199.
[7] H. Fredricksen, I. Kessler, Lexicographic compositions and de Bruijn sequences, J. Combin. Theory Ser. A 22 (1) (1977) 17–30.
[8] H. Fredricksen, J. Maiorana, Necklaces of beads in k colors and k-ary de Bruijn sequences, Discrete Math. 23 (1978) 207–210.
[9] S.W. Golomb, Shift Register Sequences, Aegean Park Press, 1982.
[10] Y. Huang, A new algorithm for the generation of binary de Bruijn sequences, J. Algorithms 11 (1) (1990) 44–51.
[11] M.H. Martin, A problem in arrangements, Bull. Amer. Math. Soc. 40 (1934) 859–864.
[12] A. Ralston, A new memoryless algorithm for de Bruijn sequences, J. Algorithms 2 (1) (1981) 50–62.
[13] J. Sawada, B. Stevens, A. Williams, De Bruijn sequences for the binary strings with maximum density, in: N. Katoh, A. Kumar (Eds.), WALCOM: Algorithms and Computation, in: LNCS, vol. 6552, Springer, 2011, pp. 182–190.
[14] J. Sawada, A. Williams, D. Wong, Universal cycles for weight-range binary strings, in: Proceedings of 24th International Workshop on Combinatorial Algorithms (IWOCA 2013), in: LNCS, vol. 8288, Springer, 2013, pp. 388–401.
[15] D. Wong, Novel universal cycle constructions for a variety of combinatorial objects (Ph.D. thesis), University of Guelph, Canada, 2015.