A SIMPLE COMBINATORIAL ALGORITHM FOR DE BRUIJN SEQUENCES

ABBAS ALHAKIM
DEPARTMENT OF MATHEMATICS
& COMPUTER SCIENCE
CLARKSON UNIVERSITY
POTSDAM, NY 13699

Abstract. This note presents a combinatorial method to construct a De Bruijn cycle for any order n.

1. Introduction

A binary De Bruijn sequence of order n is a string of bits that contains every possible pattern of size n exactly once each. Although the existence of such sequences is not obvious, it is well known that they exist for all orders n and that the number of distinct sequences is $2^{2^{k-1}-k}$, see De Bruijn [3]. Observe that this number is $1, 1, 2, 16$ and 2048 for $n = 1, \ldots, 5$ respectively.

One aspect of these sequences makes them similar to Gray codes. A Gray code is a Hamiltonian cycle in a hypercube while a de Bruijn cycle is a Hamiltonian cycle in the so-called De Bruijn digraph. A De Bruijn digraph $G_n = (V_n, E_n)$ of order n has the same vertex set as a hypercube, i.e., the 2^n binary patterns, but the edges are different. For two vertices $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \cdots, y_n)$, $(\mathbf{x}, \mathbf{y})$ is an edge if and only if $y_i = x_{i+1};\ i = 1, \ldots, n-1$. The De Bruijn digraphs of orders 2 and 3 are shown in Figure 1. Interestingly, Eulerian circuits and Hamiltonian cycles are intimately related in the case of De Bruijn digraphs: the consecutive edges of a Eulerian circuit in G_{n-1} form a Hamiltonian cycle in G_n.

These combinatorial objects have been used to design a rotating drum and to break a hypothetical key-lock system [4]. Besides these recreational problems, they have been used in diverse fields such as the generation of pseudo-random sequences. In fact, linear feedback shift register sequences having the longest period possible are De Bruijn sequences without the all zero pattern, referred as punctured de Bruijn sequences and they have been popular in Engineering applications because they are implemented efficiently by hardware devices called linear feedback shift registers (LFSR), see Golomb [5].

The first conference on De Bruijn cycles was held in 2004 with a theme of advancing the interest and knowledge in the area of De Bruijn cycles and their many generalizations as they are seen to belong to a larger family of combinatorial objects called universal sets, see [1, 2, 6, 7].

There are various methods for generating De Bruijn cycles, but the most popular are by finding linear recurrences that can have an almost full period (LFSR), this is

Key words and phrases. De Bruijn sequence, prefer one algorithm.

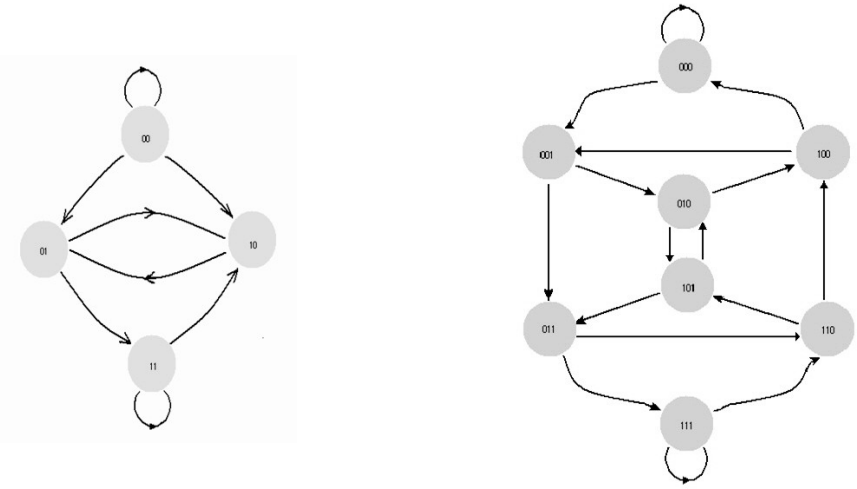

FIGURE 1. De Bruijn digraphs of orders 2 and 3 respectively

based on primitive polynomials in Galois field $GF(2)$. There are also graphical and combinatorial methods. Fredricksen [4] is an excellent exposition that surveys the known methods of generation. In this survey, Fredricksen [4, p. 207] writes "When the mathematician on the street is presented with the problem of generating a full cycle, one of three things happens: he gives up, or produces a sequence based on a primitive polynomial, or produces the [prefer-one sequence]. Only rarely is a new algorithm proposed." The prefer-one algorithm is a very simple method amazingly capable of generating a full cycle. For any positive integer $n \geq 1$, the algorithm puts n zeroes, and proceeds after this by proposing 1 for the next bit and accepting it when the word formed by the last n bits has not been encountered previously in the sequence, otherwise 0 is placed. The algorithm stops when both 0 and 1 do not bring a new word.

Fredricksen cites five authors who discovered the prefer-one algorithm. It is perhaps worth mentioning that the author of this note rediscovered this same algorithm as a Hamiltonian problem relating to the graph of some simple Markov chain.

A similar combinatorial algorithm, also given in Fredricksen [4, p. 212], is named prefer-same algorithm but is more elaborate than the prefer-one. In this note we present and prove yet another algorithm of the same family, which we refer as the prefer-opposite algorithm. To the best of the author's knowledge, it has not been published any where, or at least it is not popular enough, despite the fact that it enjoys better properties than its more famous cousins, as will be explained at the end of this note. For a bit b and a positive integer i we will use the terminology $\bar{b} = 1 - b$ and $b^i = \underbrace{b \ldots b}_{i}$.

Prefer-Opposite Algorithm

1. Let $x_1 = \cdots = x_n = 0$.
2. $i = n + 1$
3. If $x_{i-n+1} \dots x_{i-1}\bar{x}_{i-1}$ is a pattern that has not appeared before then let $x_i = \bar{x}_{i-1}$, increment i by one and repeat Step 3.
4. Otherwise, if $x_{i-n+1} \dots x_{i-1}x_{i-1}$ is a pattern that has not appeared earlier in the sequence then let $x_i = x_{i-1}$, increment i by one and go to Step 3.
5. Otherwise, stop.

Lemma 1.1. *The algorithm does not produce the all one pattern.*

Proof. Let $\mathbf{1}^n$ be the all one pattern of size n. Because of the self-loop at this vertex, it can only appear after the n-pattern $01 \dots 1$. By preferring the opposite, the latter pattern must be followed by $1 \dots 10$. □

The next result shows that the all one pattern is the only one that does not appear.

Theorem 1.2. *For any integer n, the Prefer Opposite Algorithm generates a cycle of size $2^n - 1$ that includes each pattern exactly once, except the all ones pattern.*

Proof. We will prove that the algorithm works by verifying two steps. (1) First, the algorithm terminates at 0^n. (2) All patterns except 1^n are included. To see (1), suppose the algorithm terminates at the point $x_1 \dots x_n$. This means that both $x_2 \dots x_n 0$ and $x_2 \dots x_n 1$ must have appeared earlier in the sequence. If $x_1 \dots x_n \neq 0^n$ then it follows that the $(n-1)$-pattern $\mathbf{x}_{n-1} := x_2 \dots x_n$ must have appeared three times, the third time being at the very end, as in the following diagram.

$$0^n \to \cdots \to b\mathbf{x}_{n-1} \to \mathbf{x}_{n-1}\bar{x}_n \to \cdots \to \bar{b}\mathbf{x}_{n-1} \to \mathbf{x}_{n-1}x_n \to \cdots \to \mathbf{x}_{n-1}x_n$$

Therefore, $x_1, \dots, x_n$ must have appeared earlier but this is a contradiction because the algorithm does not allow repetitions. Hence, the algorithm can only terminate with 0^n.

To prove (2) suppose a word $\mathbf{w} = x_1 \dots x_n$–that does not consist of ones only–does not appear in the sequence. Then we can assume without loss of generality that $x_{n-1} = x_n = b$ (if $x_1 \dots x_{n-2}x_{n-1}\bar{x}_{n-1}$ does not appear, neither does $x_1 \dots x_{n-2}bb$). Under this assumption we will show that the word $x_2 \dots x_{n-2}bbb$ does not appear in the sequence. If it did then we must have also seen $x_2 \dots x_{n-2}bb\bar{b}$. So for some bit c the following two words are present. $cx_2 \dots x_{n-2}bb$ and $\bar{c}x_2 \dots x_{n-2}bb$. But then the word $\mathbf{w}$ must be present. Let us consider two cases.
Case 1. If $b = 0$, repeating the previous argument inductively shows that the word $\mathbf{z} = x_{n-2}\underbrace{0 \dots 0}_{n-1}$ does not appear. This immediately leads to a contradiction, as it says that the algorithm does not terminate at zero.
Case2. $b = 1$. Let i be the smallest index $1 \leq i \leq n$ so that $x_i = 0$ and $x_j = 1; j > i$. This choice of i is possible due to the assumption that $\mathbf{w}$ is not all ones. Now repeating the same argument above $i - 1$ times reveals that $0\underbrace{1 \dots 1}_{n-1}$ is not in the sequence. By Lemma 1.1, this in turn implies that $\underbrace{1 \dots 1}_{n-1}0$ does not appear. Applying the argument in Case 1 shows again that the sequence does not end with 0^n, which establishes the proof. □

Proposition 1.3. *The longest run of ones, 1^{n-1}, is the last run of ones in the sequence.*

Proof. The proposition claims that 1^{n-1} is followed by 0^{n-1} (and then the algorithm halts). Suppose not. Then 1^{n-1} is followed by 0^j1 for some $j < n-1$. This implies that the pattern 1^i0^j1; $i+j = n-1$ occurs after 01^{n-1}. Since the algorithm cannot halt at this pattern, it follows that $1^{i-1}0^j10$ and therefore $1^{i-1}0^j11$ occur afterwards. Applying this argument $(n-2)$ times implies that 01^{n-1} occurs subsequently for a second time, a contradiction. □

To produce a full De Bruijn sequence, the algorithm can therefore be adjusted so as to count the number of ones in the running end of the sequence, when this count is $n-1$, we append 1 and exit.

The following table displays the prefer-opposite sequence as well as the prefer-one sequence without the lst $n-1$ zeroes for $n = 1, \ldots, 5$. It can be seen that for $n \leq 4$ one of these sequences is the reverse of the other. This is not the case though for $n = 5$.

By construction, the prefer-one is biased to 1's, while the prefer opposite tends to keep the balance between 1's and 0's. In fact, Fredricksen [4] states that the prefer-one sequence has most of its ones in the beginning. For $n = 59$ the first 10^6 bits are 90 percent one, and the first 1/4 of the sequence for $n = 22$ is 60 percent 1. These percentages of ones within the 1/4 of the prefer-opposite sequences are very close to 50% for $n = 10, \ldots 20$.

n	prefer-one sequence	prefer-opposite sequence
1	01	01
2	0011	0011
3	00011101	00010111
4	0000111101100101	0000101001101111
5	00000111110111001101011000101001	00000101011010010001100111011111

References

[1] F. Chung, P. Diaconis, R. Graham, Universal Cycles for Combinatorial Structures, Discrete Mathematics, 110, 43-59, (1992).

[2] J. Cooper, R. Graham, "Generalized de Bruijn Cycles". Annals of Combinatorics, 8(1): 13-25, (2004).

[3] N. G. de Bruijn, "A Combinatorial Problem". Koninklijke Nederlandse Akademie v. Wetenschappen 49: 758764, (1946).

[4] H. Fredricksen, "A Survey of Full Length Nonlinear Shift Register Cycle Algorithms". SIAM Review, 24(2): 195-221, (1982).

[5] S.Golomb, Shift Register Sequences. Holden-Day, Inc., (1967).

[6] D. Knuth, The Art of Computer Programming Vol. 4A. To appear.

[7] C. Savage, "A Survey of Combinatorial Gray Codes". SIAM Review 39(4): 605-629, (1997).

Clarkson University

E-mail address: aalhakim@clarkson.edu