Revisiting the Prefer-same and Prefer-opposite de Bruijn sequence constructions

Abbas Alhakim Evan Sala Joe Sawada

Abstract

We present a simple greedy algorithm to construct the prefer-same de Bruijn sequence and prove that it is equivalent to the more complex algorithm first stated by Eldert et al. without proof [*AIEE Transactions* 77 (1958)], and later by Fredricksen [*SIAM Review* 24 (1982)]. Then we prove that the resulting sequence has the lexicographically largest run-length representation among all de Bruijn sequences. Furthermore, we prove that the sequence resulting from a prefer-opposite greedy construction has the lexicographically smallest run-length representation among all de Bruijn sequences.

1 Introduction

A *de Bruijn sequence* of order n is a sequence of bits that when considered cyclically contains every length n binary string as a substring exactly once. A consequence of this definition is that every de Bruijn sequence has length 2^n. There are $2^{2^{n-1}-n}$ distinct de Bruijn sequences [10] and many known constructions (see surveys in [6] and [8]). Perhaps the most surprising constructions, and the ones most accessible to the broadest audience, are the greedy constructions. A greedy construction starts with a seed string, then repeatedly applies some greedy rule to determine the next bit of the sequence. The algorithm stops when it is impossible to add another bit without creating a duplicate substring of length n, or some related termination condition is reached. The most well-known greedy de Bruijn sequence construction is the *Prefer-one* (or equivalently Prefer-zero) described as follows.

Prefer-one algorithm [9]

1. Seed with 0^{n-1}
2. **Repeat** until no new bit is added: Append a 1 if it does not create a duplicate length n substring; otherwise append a 0 if it does not create a duplicate length n substring
3. Remove the seed

For example, applying this construction for $n = 4$ we obtain the string:

$$\text{~~000~~}\ 1111011001010000.$$

Note the importance of the string 0^{n-1} to seed the algorithm[1]. Starting with any other seed will not produce a de Bruijn sequence using this greedy approach. For instance, starting with 101 we obtain:

$$\text{~~101~~}\ 1110101.$$

Adding a 1 to the above string yields the repeated substring 1011 and adding a 0 yields the repeated substring 1010. Clearly 0000, among others, is missing from this construction.

There are two approaches whose greedy decision is based on the last bit generated (for an extension of these approaches see [2]): the *Prefer-opposite* and the *Prefer-same*. The Prefer-opposite algorithm (where *opposite* effectively means complement), presented below, differs from the original presentation [1] in two ways: (1) a termination condition is applied so that the string 1^n is not missed and (2) the seed prefix 0^{n-1} is rotated to the end of the resulting sequence.

Prefer-opposite algorithm

1. Seed with 0^{n-1}
2. Append 0
3. **Repeat** until the suffix is 1^{n-1}: Append the **opposite** bit as the last if it does not create a duplicate length n substring; otherwise append the same bit as the last
4. Append 10^{n-1}
5. Remove the seed

Let $\mathcal{O}_n$ denote the sequence resulting from the Prefer-opposite algorithm for a given n. Applying this construction for $n = 4$ we obtain:

$$\mathcal{O}_4 = \text{~~000~~}\ 0101001101111000.$$

In Section 3 we demonstrate that the run-length encoding of $\mathcal{O}_n$ is the lexicographically smallest over all de Bruijn sequences of order n.

The Prefer-same construction was first described by Eldert et al. [4] in Appendix II, with no formal proof of correctness. The algorithm is restated by Fredricksen in [6, P. 212], as shown in Figure 1, along with a proof that the algorithm produces a de Bruijn sequence.

[1]This algorithm has also been described to be seeded with 0^n to generate a linear version of the sequence (the seed is not removed at the end)[6].

ALGORITHM 4 (*"prefer same"*).

(1) Write n ones followed by n zeros to form

$$x_1 x_2 \cdots x_n x_{n+1} \cdots x_{2n} = 11 \cdots 10 \cdots 0.$$

(2) Assume at the kth step, $k \geqq 2n$ i, $i = 0$ or 1, has been placed. Then at step $k+1$ place another i provided the following two conditions are not violated.
   (a) The n-tuple $x_{k-n+2} \cdots (x_k = i)(x_{k+1} = i)$ has not previously appeared in the sequence, and
   (b) If placing $x_{k+1} = i$ makes a string of i's of length t, then we have not already written 2^{n-2-t} strings of i's of length t in the sequence. If (a) and (b) are not violated for i, then increase k and continue with (2); otherwise

(3) If (a) and (b) is violated for i, then place $x_{k+1} = 1 - i$ as long as (2a) and (2b) are not violated for $1 - i$ then increase k and continue with step 2. If we also cannot place $1 - i$, the algorithm terminates. Considering the beginning of the sequence to follow the end creates a full cycle.

Figure 1: The prefer-same algorithm as presented by Fredricksen [6, P. 212].

Observe that this algorithm does not have the same simplistic descriptions as the previous two; it has an additional condition that requires counting the number of occurrences of specific substrings as used in Step 2(b). Let $\mathcal{D}_n$ denote the sequence resulting from this original Prefer-same algorithm for a given n. As an example, this algorithm generates the following de Bruijn sequence for $n = 4$:

$$\mathcal{D}_4 = 1111000011\underline{0}1\underline{0}010.$$

The two underlined bits are the ones that are not the "same" as the previous bit even though applying the same bit would not create a duplicate length n substring; the same bit is not chosen since Step 2(b) is violated. Here we make an important clarification of this step. Prior to stating the pseudocode shown in Figure 1, Fredricksen [6] defines a *run of t 0's* to mean a string of t zeros surrounded by a 1 (or the empty string) on each side. Similarly for *runs of 1's*. These, in fact are the strings being counted at Step 2(b) from the expression "strings of i's of length t". Going back to the example above, for $n = 4$ and $t = 2$, the second underlined bit is not a 1 since we have already seen $2^{4-2-2} = 1$ substrings 0110 prior to it.

The primary result of this paper is to provide a simpler Prefer-same construction and prove that it is equivalent to the original construction. This is done in Section 2. Then in Section 3 we prove interesting properties of the Prefer-opposite and Prefer-same sequences with respect to their run-length encoding. Implementations of these greedy algorithms are available at `http://debruijnsequence.org`.

2 Revisiting the Prefer-Same Algorithm

In this section we present a simplified Prefer-same construction and prove it is equivalent to the construction given by Fredricksen [6] which is based on the algorithm from Eldert et al. [4]. The algorithm is as follows:

Simplified Prefer-same algorithm

1. Seed with length $n-1$ string $\mathbf{s} = \cdots 01010$
2. Append 1
3. **Repeat** until no new bit is added: Append the **same** bit as the last if it does not create a duplicate length n substring; otherwise append the opposite bit as the last if it does not create a duplicate length n substring
4. Remove the seed $\mathbf{s}$

Let $\mathcal{S}_n$ denote the sequence resulting from the Simplified Prefer-same algorithm for a given n. For $n = 4$, the sequence generated by this construction is the same as the original algorithm:

$$\mathcal{S}_4 = \text{~~010~~}\, 1111000011010010.$$

From the proof of the original Prefer-same algorithm [6] we obtain the following remark where $\mathbf{s}$ denotes the length $n-1$ seed string $\cdots 01010$.

Remark 2.1 *$\mathcal{D}_n$ ends with the length $n-1$ seed $\mathbf{s} = \cdots 01010$, for $n > 1$.*

Theorem 2.2 *The sequence $\mathcal{S}_n$ is a de Bruijn sequence for all $n > 1$. Moreover, $\mathcal{S}_n = \mathcal{D}_n$.*

Proof. Let $\mathcal{D}_n = d_1 d_2 \cdots d_{2^n}$, let $\mathcal{S}_n = s_1 s_2 \cdots s_m$, and let $\mathbf{s}$ denote the length $n-1$ seed string $\mathbf{s} = \cdots 01010$. Since $\mathbf{s}\mathcal{S}_n$ has no duplicate substrings of length n, $m \leq 2^n$. It is easy to observe from the two algorithms that $d_1 d_2 \cdots d_{2n} = s_1 s_2 \cdots s_{2n} = 1^n 0^n$. Suppose there exists a smallest integer $2n < j \leq m$ such that $d_j \neq s_j$.

- **Case 1:** Suppose $d_j = d_{j-1}$. Then $s_j \neq s_{j-1}$. Consider $\mathbf{d} = d_{j-n+1} \cdots d_{j-1} d_j$. Since $\mathcal{D}_n$ is a de Bruijn sequence, $\mathbf{d}$ is not a substring of $d_1 d_2 \cdots d_{j-1} = s_1 s_2 \cdots s_{j-1}$. Thus, since $s_j \neq s_{j-1}$, by the definition of $\mathcal{S}_n$, it must be that $\mathbf{d}$ is a substring of $\mathbf{s}\mathcal{S}_n$ and thus it must be a substring of $\mathbf{s} s_1 s_2 \cdots s_{n-1} = \mathbf{s}1^{n-1}$. However since $\mathcal{D}_n$ ends with $\mathbf{s}$ (Remark 2.1), $\mathbf{d}$ also appears in the wraparound of $\mathcal{D}_n$, contradicting $\mathcal{D}_n$ being a de Bruijn sequence.
- **Case 2:** Suppose $d_j \neq d_{j-1}$. Then $s_j = s_{j-1}$. Consider $\mathbf{b} = s_{j-n+1} \cdots s_{j-1} s_j$. By the definition of $\mathcal{S}_n$, $\mathbf{b}$ is not a substring of $\mathbf{s} s_1 s_2 \cdots s_{j-1}$ and hence also not a substring of

$d_1 d_2 \cdots d_{j-1}$. From the definition of $\mathcal{D}_n$, because $d_{j-1} \neq d_j$, this means that $\mathbf{b}$ will never become a substring of the linear $\mathcal{D}_n$. Since $\mathcal{D}_n$ ends with $\mathbf{s}$ (Remark 2.1), $\mathbf{b}$ will also not be found in the wraparound, since it is not a substring of $\mathbf{s}s_1 s_2 \cdots s_{j-1}$ and $j > 2n$. This contradicts $\mathcal{D}_n$ being a de Bruijn sequence.

Since both cases both end in contradictions, $d_j = s_j$ for all $1 \leq j \leq m$. Suppose $m < 2^n$. Consider the length $n-1$ suffix $\mathbf{z} = s_{m-n+2} \cdots s_{m-1} s_m$ of $\mathcal{S}_n$. Since $\mathcal{S}_n$ terminates after generating m symbols, $\mathbf{z}0$ and $\mathbf{z}1$ both appear in $\mathbf{s}\mathcal{S}_n$ as substrings. But since $\mathbf{z} = d_{m-n+2} \cdots d_{m-1} d_m$, the string $\mathbf{z}d_{m+1}$ does not appear in $d_1 d_2 \cdots d_m$ since $\mathcal{D}_n$ is a de Bruijn sequence, which means it must appear as a substring of $\mathbf{s}d_1 d_2 \cdots d_{n-1}$. But since $\mathcal{D}_n$ ends with $\mathbf{s}$ (Remark 2.1), this implies that $\mathbf{z}d_{m+1}$ appears twice as a substring in $\mathcal{D}_n$ when considered cyclically. This contradicts $\mathcal{D}_n$ being a de Bruijn sequence. Thus $m = 2^n$, $\mathcal{D}_n = \mathcal{S}_n$, and $\mathcal{S}_n$ is a de Bruijn sequence. □

In the proof above, $\mathcal{S}_n$ is shown to be a de Bruijn sequence by simply establishing that it coincides with the de Bruijn sequence $\mathcal{D}_n$. A more direct proof similar to the classical proof that the Prefer-one sequence is a de Bruijn sequence could also have been applied, but then we still would have to confirm that $\mathcal{S}_n = \mathcal{D}_n$.

3 Run-length Properties of $\mathcal{O}_n$ and $\mathcal{S}_n$

The sequences $\mathcal{O}_n$ and $\mathcal{S}_n$ both have an interesting property with respect to their run-length encoding. The *run-length encoding* of a string $w_1 w_2 \cdots w_m$ is a compressed representation that stores consecutively the lengths of the maximal runs of each symbol. For example the string 11000110 has run-length encoding 2321. Since we are dealing with binary strings we need the extra information regarding the starting symbol to obtain the original string from its run-length encoding. As further examples:

$\mathcal{O}_5 = 0101011010010001100111010111110000$ has run-length encoding 11111211213223154, and

$\mathcal{S}_5 = 1111100000111011001101000100101 0$ has run-length encoding 5531222113121111.

The main results of this section establish that $\mathcal{O}_n$ and $\mathcal{S}_n$ admit extremes with respect to lexicographical run-length encoding. These results are analogous to the well-known fact that the Prefer-one sequence is the lexicographically largest de Bruijn sequence for a given order n.

Proposition 3.1 *The de Bruijn sequence $\mathcal{O}_n$ has the lexicographically smallest run-length encoding of all de Bruijn sequences of order n beginning with 0.*

Proposition 3.2 *The de Bruijn sequence $\mathcal{S}_n$ has the lexicographically largest run-length encoding of all de Bruijn sequences of order n beginning with 1.*

To prove the property for $\mathcal{O}_n$ we first prove another property of the sequence. We define the *alternating string* of length n, denoted γ_n, to be the string $b_1 b_2 \cdots b_n$ where $b_i = 0$ if i is odd and $b_i = 1$ if i is even. Thus $\gamma_5 = 01010$ and $\gamma_4 = 0101$. Observe below that the five underlined strings $1\gamma_5, 10\gamma_4, 100\gamma_3, 1000\gamma_2, 10000\gamma_1$ appear in order from left to right within $\mathcal{O}_6$:

$$\mathcal{O}_6 = 0\underline{1010}\underline{10010}1010111\underline{10001}00110110\underline{10000}1100011100111101111\underline{100000}.$$

This observation is generalized in the upcoming Lemma 3.4. First we remark on the first and last strings of $\mathcal{O}_n$. Observe that the first $n-1$ applications of step 3 in the Prefer-opposite algorithm do not result in a duplicate substring by appending the "opposite bit". Thus $\mathcal{O}_n$ begins with γ_n. Furthermore, from step 4 in the algorithm it is clear that $\mathcal{O}_n$ ends with 0^{n-1}.

Remark 3.3 *The sequence $\mathcal{O}_n$ begins with γ_n and ends with 0^{n-1}.*

Lemma 3.4 *The strings $1\gamma_{n-1}, 10\gamma_{n-2}, 100\gamma_{n-3}, \ldots, 10^{n-2}\gamma_1$ appear in order from left to right within $\mathcal{O}_n$. Moreover, when considering $\mathcal{O}_n$ as a circular string, the bit following each string is the same as the given string's last bit.*

Proof. Let $\alpha = 10^{j-1}\gamma_{n-j}$ and $\beta = 10^{j}\gamma_{n-j-1}$ for some $1 \leq j \leq n-2$. Since $\mathcal{O}_n = o_1 o_2 \cdots o_{2^n}$ begins with γ_n and ends with 0^{n-1} (see Remark 3.3), $\beta = o_t o_{t+1} \cdots o_{t+n-1}$ for some $n \leq t \leq 2^n - n + 1$. Let σ be the substring of length $n-j-1$ preceding β in $\mathcal{O}_n$. Since $o_{t+j} = 0$ and the Prefer-opposite greedy algorithm sets $o_{t+j+1} = 0$ (the start of γ_{n-j-1}), it must be that $\sigma 10^j 1 = \sigma 10^{j-1}01$ already exists as a substring in $0^{n-1}o_1 o_2 \cdots o_{t+j}$. This means, by the definition of the Prefer-opposite algorithm, that either this substring continues into $\sigma 10^{j-1}\gamma_{n-j} = \sigma\alpha$ or α has previously appeared in $0^{n-1}o_1 o_2 \cdots o_{t+j}$ by applying similar arguments. For the latter case, since α begins with 1, it must be a substring of $o_1 o_2 \cdots o_{t+j}$. In both cases above, α appears before β in $\mathcal{O}_n$ and hence the strings $1\gamma_{n-1}, 10\gamma_{n-2}, \ldots, 10^{n-2}\gamma_1$ appear in order from left to right in $\mathcal{O}_n$. Moreover, $0^i\gamma_{n-i}$ is a substring of $0^{n-1}o_1 o_2 \cdots o_n$ for $0 \leq i \leq n-1$. Thus by the definition of $\mathcal{O}_n$, the bit following $10^i\gamma_{n-i-1}$ must be the *same* as the last bit of γ_{n-i-1}. For the special case when $i = n-2$, note that $\mathcal{O}_n$ has suffix $10^{n-2}\gamma_1 = 10^{n-1}$ which ends with 0. Considering the wraparound, the next bit is the first bit of $\mathcal{O}_n$ which is 0. □

We now use this result to prove Proposition 3.1. We do not require such an auxiliary result to prove Proposition 3.2. Instead, we apply the fact that $\mathcal{S}_n = \mathcal{D}_n$.

PROOF OF PROPOSITION 3.1. The proof is by contradiction. Suppose there exists a de Bruijn sequence $\mathcal{B} = b_1 b_2 \cdots b_{2^n}$ beginning with 0 that has a smaller run-length encoding than $\mathcal{O}_n = o_1 o_2 \cdots o_{2^n}$. Then there exists a smallest index j such that $o_1 o_2 \cdots o_{j-1} = b_1 b_2 \cdots b_{j-1}$, $b_j \neq o_j$, $b_{j-1} \neq b_j$, and $o_{j-1} = o_j$. From Remark 3.3, $\mathcal{O}_n$ begins with γ_n. Thus $j > n$. Since $\mathcal{B}$ is a de Bruijn sequence, the length n string $\mathbf{b} = b_{j-n+1} b_{j-n+2} \cdots b_j$ cannot exist as a substring in $b_1 b_2 \cdots b_{j-1}$ (otherwise it appears twice). This means $\mathbf{b}$ must be a substring of

$0^{n-1}o_1o_2\cdots o_{n-1}$ because otherwise from step 3 of the Prefer-opposite algorithm we would have $o_j \neq o_{j-1}$. Thus $\mathbf{b} = 0^i\gamma_{n-i}$ for some $0 < i < n$. Noting that $\mathcal{O}_n$ begins with 0 and ends with 0^{n-1} from Remark 3.3, the string 0^n is found in the wraparound of $\mathcal{O}_n$ and thus we further have $i < n-1$. Also $o_{j-n} \neq 0$ because otherwise $o_{j-n}o_{j-n+1}\cdots o_{j-1} = 0^{i+1}\gamma_{n-i-1}$ which is also found in the wraparound of $\mathcal{O}_n$ (by Remark 3.3). Thus $b_{j-n} = o_{j-n} = 1$ and $o_{j-n}o_{j-n+1}\cdots o_{j-1} = 10^i\gamma_{n-i-1}$. Now, because $\mathcal{B}_n$ already contains γ_n as a prefix, the bits following $\mathbf{b}$ cannot continue to alternate creating a substring γ_n. Furthermore, $\mathcal{B}_n$ cannot end without repeating consecutive bits after $\mathbf{b}$ since otherwise it will not contain the string 10^{n-1}, which is the length n suffix of $\mathcal{O}_n$, and thus is not already a substring of $b_1b_2\cdots b_j$. Thus, there exist some unique $1 \leq t < n$ such that the next $t-1$ bits following $\mathbf{b}$ continue to alternate, but the t-th bit is the same as the previous bit. This means the length n string $\mathbf{b}' = b_{j-n+1+t}\cdots b_{j+t-1}b_{j+t} = 0^{i-t}\gamma_{n-i+t-1}b_{j+t}$ where the last two bits have the same value. Since earlier we observed that $10^i\gamma_{n-i-1}$ appears in $o_1o_2\cdots o_j$, by Lemma 3.4 this means that both $10^{i-t}\gamma_{n-i+t-1}$ and $\mathbf{b}'$ appear in $o_1o_2\cdots o_{j-1} = b_1b_2\cdots b_{j-1}$. But this means $\mathbf{b}'$ appears twice in $\mathcal{B}$, which contradicts the supposition that $\mathcal{B}$ is de Bruijn sequence. □

PROOF OF PROPOSITION 3.2. The proof is by contradiction. Suppose there exists a de Bruijn sequence $\mathcal{B} = b_1b_2\cdots b_{2^n}$ beginning with 1 that has a larger run-length encoding than $\mathcal{D}_n = \mathcal{S}_n = s_1s_2\cdots s_{2^n}$. Then there exists a smallest index j such that $s_1s_2\cdots s_{j-1} = b_1b_2\cdots b_{j-1}$, $b_j \neq s_j$, $b_{j-1} = b_j$, and $s_{j-1} \neq s_j$. By construction of $\mathcal{D}_n$, $s_1s_2\cdots s_n = 1^n$, so $j > n$. Since $\mathcal{B}$ is a de Bruijn sequence, the string $\mathbf{b} = b_{j-n+1}b_{j-n+2}\cdots b_j$ cannot exist as a substring in $b_1b_2\cdots b_{j-1}$ (otherwise it appears twice). This means the algorithm that constructs $\mathcal{D}_n$ sets $s_j \neq s_{j-1}$ because otherwise setting $s_j = s_{j-1}$ violates some run constraint. Thus, if $b_{j-1} = b_j$, then $\mathcal{B}$ also violates a run constraint which contradicts $\mathcal{B}$ being a de Bruijn sequence. □

4 Conclusion

The greedy algorithms outlined in this paper can all be implemented to construct their respective de Bruijn sequences in $O(n)$ time per bit. However they have one major downside; they require $O(2^n)$ memory to store each length n substring. The sequence constructed by the Prefer-one greedy algorithm can also be constructed using only $O(n)$ space by either a successor-rule approach [5, 8] or a concatenation scheme [7]. For $\mathcal{O}_n$ and $\mathcal{S}_n$, efficient algorithms have been recently discovered that are experimentally validated up to $n = 30$ [3]. It is anticipated that the results provided in this paper will be helpful in discovering a formal proof of correctness.

5 Acknowledgements

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), RGPIN-2018-04211.

References

[1] A. Alhakim. A simple combinatorial algorithm for de Bruijn sequences. *The American Mathematical Monthly*, 117(8):728–732, 2010.

[2] A. Alhakim. Spans of preference functions for de Bruijn sequences. *Discrete Applied Mathematics*, 160(7-8):992 – 998, 2012.

[3] A. Alhakim, D. Gabric, J. Sawada, and E. Sala. Efficient constructions of the prefer-same and prefer-opposite de Bruijn sequences. *manuscript*, 2019.

[4] C. Eldert, H. Gray, H. Gurk, and M. Rubinoff. Shifting counters. *AIEE Trans.*, 77:70–74, 1958.

[5] H. Fredricksen. Generation of the Ford sequence of length 2^n, n large. *J. Combin. Theory Ser. A*, 12(1):153–154, 1972.

[6] H. Fredricksen. A survey of full length nonlinear shift register cycle algorithms. *Siam Review*, 24(2):195–221, 1982.

[7] H. Fredricksen and J. Maiorana. Necklaces of beads in k colors and k-ary de Bruijn sequences. *Discrete Math.*, 23:207–210, 1978.

[8] D. Gabric, J. Sawada, A. Williams, and D. Wong. A framework for constructing de Bruijn sequences via simple successor rules. *Discrete Mathematics*, 341(11):2977 – 2987, 2018.

[9] M. H. Martin. A problem in arrangements. *Bull. Amer. Math. Soc.*, 40(12):859–864, 1934.

[10] T. van Aardenne-Ehrenfest and N. de Bruijn. *Circuits and trees in oriented linear graphs*, pages 149–163. Birkhäuser Boston, Boston, MA, 1987.