The lexicographically smallest universal cycle for binary strings with minimum specified weight

Joe Sawada\textsuperscript{a,1}, Aaron Williams\textsuperscript{a,2}, Dennis Wong\textsuperscript{a,*}

\textsuperscript{a}School of Computer Science, University of Guelph, Canada.

Abstract

Fredricksen, Kessler and Maiorana discovered a simple but elegant construction of a universal cycle for binary strings of length $n$: Concatenate the aperiodic prefixes of length $n$ binary necklaces in lexicographic order. We generalize their construction to binary strings of length $n$ whose weights are in the range $c, c+1, \ldots, n$ by simply omitting the necklaces with weight less than $c$. We also provide an efficient algorithm that generates the universal cycles in constant amortized time per bit using $O(n)$ space. Our universal cycles have the property of being the lexicographically smallest universal cycle for the set of binary strings of length $n$.

1. Introduction

Let $B(n)$ denote the set of all binary strings of length $n$. A universal cycle for a set $S$ is a cyclic sequence $u_1u_2\ldots u_{|S|}$ where each substring of length $n$ corresponds to a unique object in $S$. When $S = B(n)$, these sequences are commonly known as de Bruijn sequences since they were proven exist and counted by de Bruijn [5] (also see [6]). For example, the cyclic sequence 0000100110101111 is a universal cycle (de Bruijn sequence) for $B(4)$; the 16 unique substrings of length 4 when considered cyclicly are:

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

When considering universal cycles for a specific set $S$, there are several important questions: Does a universal cycle exist for $S$? What is the number of universal cycles for $S$? How can a specific universal cycle for $S$ be constructed? Is there an efficient algorithm that constructs a universal cycle for $S$? The last two questions can also be asked for the lexicographically smallest universal cycle for $S$. By \textit{lexicographically smallest}, we mean that the linear representation is the smallest possible in lexicographic order. For instance, the universal cycle from our example is the lexicographically smallest for $B(4)$. (The term \textit{minimal} is also used in the literature [18, 19] for the same concept.)

*Corresponding author

Email addresses: jsawada@uoguelph.ca (Joe Sawada), haron@uvic.ca (Aaron Williams), cwong@uoguelph.ca (Dennis Wong)

\textsuperscript{1}Research supported by NSERC.

\textsuperscript{2}Research supported in part by the NSERC Accelerator and Discovery Programmes, and a basic research grant from ONR.

Preprint submitted to Elsevier

December 1, 2013
The lexicographically smallest universal cycle for $B(n)$ was first constructed by Martin in the 1930s [17]. They showed that the lexicographically smallest universal cycle for $B(n)$ can be constructed by a greedy algorithm that uses exponential space. Later, Fredricksen, Kessler and Maiorana provided a more direct method in [8] for constructing this universal cycle, and this method is now referred to as the FKM construction. Ruskey, Savage, and Wang [20] provided an algorithm for generating the FKM construction and analyzed its efficiency. Due to its importance and interesting history, Knuth refers to the lexicographically smallest universal cycle for $B(n)$ as the grand-daddy of de Bruijn sequences [15].

Universal cycles have been studied for a variety of combinatorial objects including permutations, partitions, subsets, multisets, labeled graphs, various functions, and passwords [1, 2, 4, 11, 12, 13, 14, 15, 16, 23, 26]. Fredricksen, Kessler and Maiorana generalize their results to construct the lexicographically smallest universal cycle for $k$-ary strings of length $n$ [9]. Many papers have focused on finding constructions and efficient algorithms to generate universal cycles for interesting subsets of $k$-ary strings of length $n$ [7, 10, 16, 22, 24, 25, 27].

Let $B^d_c(n)$ denote the set of length $n$ binary strings whose weights (number of 1s) are in the range $c, c + 1, \ldots, d$. A universal cycle for binary strings with a minimum specified weight is a cyclic sequence of length $\binom{n}{c} + \binom{n}{c+1} + \cdots + \binom{n}{d}$ that contains each string in $B_c^n(n)$ exactly once as a substring. We refer to these universal cycles as minimum-weight universal cycles for simplicity. For example, the circular sequence 00110101111 is a minimum-weight universal cycle for $B^4_2(4)$ since its 11 substrings of length 4 include each element in $B^4_2(4) = \{0011, 0101, 0110, 1001, 1010, 1100, 0111, 1011, 1101, 1110, 1111\}$ exactly once. Similarly, a universal cycle for binary strings with a maximum specified weight, or simply a maximum-weight universal cycle, is a cyclic sequence of length $\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{d}$ that contains each string in $B^d_0(n)$ exactly once as a substring. A maximum-weight universal cycle for $B^d_0(n)$ can be obtained by complementing each bit of a minimum-weight universal cycle for $B^d_{n-d}(n)$ [24].

In this paper, a universal cycle has an efficient algorithm if each successive symbol of the sequence can be generated in constant amortized time (CAT) while using a polynomial amount of space with respect to $n$. A universal cycle for $B^d_{d-1}(n)$ is known as a dual-weight universal cycle, and more generally a universal cycle for $B^d_c(n)$ is known as a weight-range universal cycle. Algorithms to generate universal cycles with various weight-ranges have previously been studied in the sequence of the following articles:

- an efficient algorithm for dual-weight universal cycles is given in [22],
- an efficient algorithm for minimum-weight and maximum-weight universal cycles is given in [24],
- an efficient algorithm for weight-range universal cycles is given in [25].

Although efficient algorithms for generating minimum-weight and maximum-weight universal cycles are given in [24] (and generalized in [25]), there are several advantages to our new results. Firstly, our new universal cycles are the lexicographically smallest, whereas the constructions in [22,
24, 25] are not. Secondly, the constructions in [24, 25] are based on cutting and pasting dual-weight
universal cycles from [22], whereas our new construction is much simpler. Thirdly, our new con-
structions are based on lexicographic order, whereas the constructions in [24, 25] are complicated
by their use of ‘cool-lex’ order. (The construction in [24] was simplified by a generalized version of
cool-lex order found in [27], although that article did not include an efficient algorithm.)

The de Bruijn graph \(G(S)\) for a set of length \(n\) strings \(S\) is a directed edge-labeled graph whose
vertex set consists of the length \(n-1\) strings that are a prefix or a suffix of the strings in \(S\). For each
string \(b_1b_2\ldots b_n \in S\) there is an edge labeled \(b_n\) that is directed from the vertex \(b_1b_2\ldots b_{n-1}\) to the
vertex \(b_2b_3\ldots b_n\). Thus, the graph has \(|S|\) edges. As an example, the de Bruijn graph \(G(B^4_2(4))\) is
illustrated in Figure 1. It is well known that \(S\) admits a universal cycle if and only if \(G(S)\) is directed
Eulerian. The de Bruijn graph \(G(B^d_c(n))\) is directed Eulerian for all \(0 \leq c < d \leq n\) [24, 25].

The problem of finding a directed Euler cycle of lexicographically minimal labels of an edge-
labeled directed graph has been applied to find the optimal encoding in a DRAM address bus [18].
The problem is proven to be NP-complete with respect to the number of edges for general directed
graphs [18]. For the de Bruijn graph \(G(B(n))\), the Euler cycle of lexicographically minimal labels
can be constructed in \(O(E)\) time where \(E\) denotes the number of edges in \(G(B(n))\) [20]. Before this
paper, it was not known if the lexicographically minimal Euler cycle can be constructed similarly in
\(O(E)\) time for \(G(B^d_c(n))\).

The main results of this paper are as follows:

1. a surprisingly simple generalization of the FKM construction that generates a minimum-
weight universal cycle,

2. a proof that demonstrates our construction generates the lexicographically smallest universal
cycle for \(B^d_c(n)\), and

3. an efficient algorithm that generates a minimum-weight universal cycle in constant amortized
time per bit using \(O(n)\) space.

The rest of this paper is presented as follows. In Section 2 we introduce the FKM construction and
some definitions and notations. In Section 3 we present a generalization of the FKM construction.
to generate a minimum-weight universal cycle. We prove that our new universal cycles are the lexicographically smallest in Section 4. In Section 5 we prove that each successive bit in our new universal cycles can be generated in constant amortized time using $O(n)$ space. This results in an $O(E)$ algorithm to find the Euler cycle in $G(B^u_c(n))$ with lexicographically minimal labels.

2. The FKM construction

Fredricksen, Kessler and Maiorana [8, 9] developed a construction for the lexicographically smallest universal cycle for $k$-ary strings of length $n$. Before we describe the construction for $k = 2$ in detail, we require some definitions and notations.

A necklace is the lexicographically smallest string in an equivalence class of strings under rotation. The aperiodic prefix of a string $\alpha$, denoted as $ap(\alpha)$, is its shortest prefix whose repeated concatenation yields $\alpha$. That is, the aperiodic prefix of $\alpha = a_1a_2\ldots a_n$ is the shortest prefix $ap(\alpha) = a_1a_2\ldots a_p$ such that $(ap(\alpha))^p = \alpha$, where exponentiation denotes repeated concatenation and $n$ is an integer. For example, when $\alpha = 001001001$, $ap(\alpha) = 001$.

A string is a prenecklace if it is the prefix of some necklace. Let the set of length $n$ binary prenecklaces, necklaces and Lyndon words with weight $w$ be denoted by $P(n,w)$, $N(n,w)$ and $L(n,w)$ respectively. For example:

- $P(6,4) = \{001111, 010111, 011011, 011101, 011110\}$,
- $N(6,4) = \{001111, 010111, 011011\}$,
- $L(6,4) = \{001111, 010111\}$.

Observe that the strings 011101 and 011110 are prefixes of the necklaces 01110111 and 011110111 respectively so they are in $P(6,4)$.

Let $\alpha = a_1a_2\ldots a_m$ and $\beta = b_1b_2\ldots b_n$ be $k$-ary strings of length $m$ and $n$ respectively, $\alpha$ is said to be lexicographically smaller than $\beta$, denoted by $\alpha < \beta$, if one of the following holds:

1. $m < n$ and $a_1a_2\ldots a_m = b_1b_2\ldots b_m$, or
2. there exists $1 \leq i \leq m, n$ such that $a_1a_2\ldots a_i = b_1b_2\ldots b_i$ and $a_{i+1} < b_{i+1}$.

The operations $> \text{ and } \leq$ are defined similarly to be the relations lexicographically larger and lexicographically smaller or equal to respectively.

Let the set of length $n$ binary necklaces be denoted by $N(n)$. The FKM construction generates a universal cycle for $B(n)$ by concatenating the aperiodic prefixes of $N(n)$ in lexicographic order. Their results can be summarized by the following formula, where $LEX$ is a function to list the input set of strings in lexicographic order.

$$FKM(n) = ap(\alpha_1) \cdot ap(\alpha_2)\ldots ap(\alpha_m) \text{ where } LEX(N(n)) = \alpha_1, \alpha_2, \ldots, \alpha_m.$$  

Figure 2 illustrates this FKM construction of a universal cycle for $B(6)$. 
3. The weighted FKM construction

Let the set of length \( n \) necklaces with weight in the range \( c, c + 1, \ldots, d \) be denoted by \( \mathbb{N}_c^d(n) \). In this section we study the lexicographic ordering of necklaces in \( \mathbb{N}_c^d(n) \) and propose a surprisingly simple construction to generate minimum-weight universal cycles. The construction follows a similar approach to the FKM construction by ordering aperiodic prefixes of the necklaces in \( \mathbb{N}_c^d(n) \) in lexicographic order. As an example, to construct a universal cycle for \( \mathbb{B}_{6}^{3}(6) \), consider the lexicographic ordering of necklaces in \( \mathbb{N}_6^3(6) \) with those that do not satisfy the weight constraint crossed out:

\[
000000, 000001, 000011, 000101, 000111, 001001, 001011, 001101, 001111, 010101, 010111, 011011, 011111, 111111.
\]

The strings that remain are the necklaces in \( \mathbb{N}_6^3(6) \). Figure 3 illustrates this weighted FKM construction of a universal cycle for \( \mathbb{B}_{6}^{3}(6) \). The construction can be expressed by the following formula:

\[
\text{FKM}_c^d(n) = \text{ap}(\alpha_1) \cdot \text{ap}(\alpha_2) \cdots \text{ap}(\alpha_m) \text{ where } \text{LEX}(\mathbb{N}_c^d(n)) = \alpha_1, \alpha_2, \ldots, \alpha_m.
\]

To prove that the construction is correct for \( d = n \), we need to consider the necklaces immediately before and after each necklace \( \alpha \) in the ordering \( \text{LEX}(\mathbb{N}_c^d(n)) \). We denote these necklaces by \( \text{prev}(\alpha) \) and \( \text{next}(\alpha) \) respectively.

**Lemma 1.** If \( \alpha = a_1a_2 \ldots a_{n-j}10^j \in \mathbb{N}_c^n(n) \), then \( \text{next}(\alpha) \) has the prefix \( a_1a_2 \ldots a_{n-j-1}1 \).

**Proof.** We need to prove that \( \alpha \) is not the last necklace in \( \text{LEX}(\mathbb{N}_c^n(n)) \), and that \( \text{next}(\alpha) \) has the stated prefix. Notice that \( \beta = a_1a_2 \ldots a_{n-j-1}1^{j+1} \in \mathbb{N}_c^n(n) \) and \( \beta > \alpha \). Therefore, \( \alpha \) is not the last
Aperiodic necklace in \( \text{LEX}(N_c^n(n)) \). Furthermore, if there is another necklace \( \gamma \in N_c^n(n) \) with \( \alpha < \gamma \leq \beta \), then \( \gamma \) must also have prefix \( a_1a_2 \ldots a_{n-j-1} \). Therefore, \( \text{next}(\alpha) \) has the stated prefix.

The following corollary follows immediately from the previous lemma.

**Corollary 2.** Suppose \( \alpha \in N_c^n(n) \) is a periodic necklace with \( |ap(\alpha)| = a_1a_2 \ldots a_{p-j-1}01^j \) where \( p < n \) and \( \frac{n}{p} \) is an integer. Then \( \text{next}(\alpha) \) has the prefix \( ap(\alpha)^{\frac{n}{p}-1}a_1a_2 \ldots a_{p-j-1} \).

**Lemma 3.** Let \( \alpha \in N_c^n(n) \) where \( 0 < c < n \). If \( \alpha \) is periodic and \( ap(\alpha) = a_1a_2 \ldots a_{p-1}1 \), then \( \text{prev}(\alpha) \) has the suffix \( 1^{n-p} \).

**Proof.** Since \( \alpha \) is periodic, it is the lexicographically smallest necklace with the prefix \( ap(\alpha) = a_1a_2 \ldots a_{p-1}1 \). Thus, \( \text{prev}(\alpha) \) must have a prefix \( \beta \) of length \( p \) that is lexicographically smaller than \( ap(\alpha) \). Since \( \text{prev}(\alpha) \) must be the lexicographically largest necklace with prefix \( \beta \), it must have suffix \( 1^{n-p} \).

In fact, if \( \alpha \) is as described in the previous lemma then \( \text{prev}(\alpha) = a_1a_2 \ldots a_{p-1}01^{n-p} \); however a proof of that result is not as simple and it is not required for our main result.

**Corollary 4.** In \( \text{LEX}(N_c^n(n)) \), there are no consecutive periodic necklaces when \( n > 1 \).

**Proof.** Consider a periodic necklace \( \alpha \in N_c^n(n) \) where \( ap(\alpha) = a_1a_2 \ldots a_{p-1}1 \). By Lemma 3 \( \text{prev}(\alpha) \) has the suffix \( 1^{n-p} \). Clearly \( \text{prev}(\alpha) \) cannot be \( 1^n \). Thus, in order for \( \text{prev}(\alpha) \) to be periodic it must contain at least two disjoint substrings of the form \( 1^{n-p} \). However, this is not possible since \( p \leq \frac{n}{2} \) because \( \alpha \) is periodic. Thus, no two consecutive necklaces in \( \text{LEX}(N_c^n(n)) \) are periodic.

**Lemma 5.** Let \( \alpha \in N_c^n(n) \) where \( 0 < c < n \) and \( \alpha \neq 1^n \). Then \( \alpha \) is a prefix of \( ap(\alpha) \cdot \text{ap}(\text{next}(\alpha)) \).

**Proof.** If \( \alpha \) is aperiodic, then the result is obvious. Otherwise if \( \alpha \) is periodic, \( \text{next}(\alpha) \) contains the prefix \( ap(\alpha)^{\frac{n}{p}-1} \) by Corollary 2 and it is aperiodic by Corollary 4. Thus \( ap(\alpha) \cdot \text{ap}(\text{next}(\alpha)) \) has the prefix \( ap(\alpha) \cdot ap(\alpha)^{\frac{n}{p}-1} = \alpha \).
Let \( \text{Neck}(\alpha) \) denote the set of strings rotationally equivalent to the binary string \( \alpha \). Observe that the length of the aperiodic prefix \( ap(\alpha) \) is equal to the number of strings in \( \text{Neck}(\alpha) \). As an example, the aperiodic prefixes of the necklaces 000111 and 010101 have length 6 and 2 which are equal to the number of strings in \( \text{Neck}(000111) = \{000111, 001110, 011100, 110001, 100011\} \) and \( \text{Neck}(010101) = \{010101, 101010\} \) respectively. Since each string \( \alpha \in B^n_c(n) \) belongs to exactly one necklace class \( \text{Neck}(\alpha) \), the following remark is easily observed.

**Remark 1.** \(|FKM^n_c(n)| = |B^n_c(n)|\).

We now prove that \( FKM^n_c(n) \) is a universal cycle for \( B^n_c(n) \).

**Theorem 1.** \( FKM^n_c(n) \) is a universal cycle for \( B^n_c(n) \).

**Proof.** From Remark 1, it suffices to show that if each string \( s \in B^n_c(n) \) appears in \( FKM^n_c(n) \) as a substring, then \( FKM^n_c(n) \) is a universal cycle for \( B^n_c(n) \). Let \( \alpha = a_1a_2...a_n \in N^n_c(n) \) be the necklace representative of the equivalence class \( \text{Neck}(s) \).

- **Case 1:** \( s \) is periodic.

  The last two necklaces in \( \text{LEX} \left( N^n_c(n) \right) \) are \( 01^{n-1} \) and \( 1^n \). The concatenation of \( ap(01^{n-1}) \) and \( ap(1^n) \) is \( 0^n \). Thus, when \( s = 1^n \), it occurs as a substring in \( FKM^n_c(n) \). Otherwise, assume \( s \neq 1^n \). Thus, \( ap(\alpha) \) must be of the form \( a_1a_2...a_{p-j}01^j \) for some \( 1 \leq j < p \). Also, \( s \) will be some rotation of \( \alpha \) of the form \( s = a_ta_{t+1}...a_na_1a_2...a_{t-1} \) where \( 1 \leq t \leq p \).

  From Lemma 3 and Corollary 2, we know that \( \text{prev}(\alpha) \) has the suffix \( 1^{n-p} \) and \( \text{next}(\alpha) \) has prefix \( (ap(\alpha))^{\frac{p-1}{2}} \cdot a_1a_2...a_{p-j}1 \). The necklaces \( \text{prev}(\alpha) \) and \( \text{next}(\alpha) \) are aperiodic by Corollary 4. Thus, the concatenation of \( \text{prev}(\alpha) \), \( ap(\alpha) \), \( \text{next}(\alpha) \), which is a substring of \( FKM^n_c(n) \), contains the substring \( 1^{n-p} \cdot ap(\alpha) \cdot (ap(\alpha))^{\frac{p-1}{2}} \cdot a_1a_2...a_{p-j}1 \) which can be expressed more simply as \( 1^{n-p}a_1a_2...a_{p-j}1 \). If \( t \leq p-j \) then \( s \) appears in the substring \( a_1a_2...a_{p-j}1 \); otherwise \( s \) appears in the substring \( 1^{n-p} \alpha \) since \( j < p \leq n-p \).

- **Case 2:** \( s \) is aperiodic.

  Since \( s \) is aperiodic it must contain at least one 0 and one 1. Thus, we can assume that \( \alpha \) has the suffix \( 01^j \) for some \( 1 \leq j < n \). If \( s = \alpha \), then clearly it is in \( FKM^n_c(n) \) since \( \alpha = ap(\alpha) \). Otherwise, since \( s \) is a rotation of \( \alpha \), let \( s = a_ta_{t+1}...a_na_1a_2...a_{t-1} \) where \( 2 \leq t \leq n \). We consider two cases depending on \( t \).

  First, suppose \( t \leq n-j \). Since \( s \neq \alpha, \alpha \) is not one of the last two necklaces in \( \text{LEX} \left( N^n_c(n) \right) \) as they are \( 01^{n-1} \) and \( 1^n \). From Lemma 1, \( \beta = \text{next}(\alpha) \) has the prefix \( a_1a_2...a_{n-j-1} \). Observe that \( s \) appears as a substring in \( \alpha \beta \). From Lemma 5, \( \beta \) occurs as a prefix of \( ap(\beta) \cdot ap(\text{next}(\beta)) \). Thus, since \( \alpha \) is aperiodic, \( ap(\alpha) \cdot ap(\beta) \cdot ap(\text{next}(\beta)) \), which is a substring of \( FKM^n_c(n) \), has the prefix \( \alpha \beta \), which contains \( s \).

  If \( t > n-j \), then \( s = 1^i a_1a_2...a_{n-j-1}01^{j-i} \) where \( i = n-t+1 \). First, we consider two special cases where \( s \) appears in the “wrap-around” of the universal cycle: those where \( s \) is of the form: \( 1^i0^{n-c}1^{c-i} \) or \( 1^i0^{n-i} \). The last two necklaces in \( \text{LEX} \left( N^n_c(n) \right) \) are \( 01^{n-1} \) and \( 1^n \), and that the first necklace is \( 0^n1^c \). Thus, when \( FKM^n_c(n) \) is considered cyclicly, it contains the substring \( 01^{n-1} \cdot 1 \cdot 0^{n-c}1^c \) which in turn has \( s \) as a substring in these cases.
For all other possible strings $s$, let $\gamma \in \mathbb{N}^n_c(n)$ be the lexicographically smallest necklace that starts with the prenecklace $a_1a_2 \ldots a_{n-j-1}10^{j-i}$. Note that $\gamma$ will not be $0^c1^{n-c}$ because we handled this special case already; hence $\text{prev}(\gamma)$ is well-defined. Observe that $\text{prev}(\gamma)$ will be the lexicographically largest necklace satisfying the weight constraint with its length $n - i$ prefix lexicographically smaller than $a_1a_2 \ldots a_{n-j-1}10^{j-i}$. This necklace will have the suffix $1^i$ because it is the lexicographically maximal with respect to this prefix. The concatenation of $\text{ap}(\text{prev}(\gamma)), \text{ap}(\gamma)$ and $\text{ap}(\text{next}(\gamma))$, which is a substring of $\text{FKM}^n_c(n)$, contains $1^i\gamma$ as a substring by Lemma 5. Thus, $s$, which is prefix of $1^i \cdot \gamma$, is a substring of $\text{FKM}^n_c(n)$.

\[ \]

One might hope that the same strategy works for the construction of universal cycles for $\text{B}^d_c(n)$ for all values of $c$ and $d$ where $0 \leq c < d \leq n$. Unfortunately, it only works when $d \in \{0, 1, n-1, n\}$.

To illustrate this fact, consider the attempted construction of a maximum-weight universal cycle for $\text{B}^1_c(6)$. The necklaces in $\mathbb{N}(6)$ are given in lexicographic order below, with those that do not satisfy the weight constraint crossed out.

\[
000000, 000001, 000011, 000101, 000111, 001001, 001011, 010101, 010111, 011011, 011111, 111111.
\]

Observe that concatenating the aperiodic prefixes of these remaining necklaces in lexicographic order:

\[
0 \cdot 000001 \cdot 000011 \cdot 000101 \cdot 000111 \cdot 001101 \cdot 001111 \cdot 011101 \cdot 011111 \cdot 01 \cdot 010111 \cdot 011,
\]

does not create a universal cycle for $\text{B}^1_c(6)$ because 111101 is a substring of the sequence but 111101 \notin $\text{B}^1_c(6)$.

**Corollary 6.** $\text{FKM}^d_c(n)$ is a universal cycle for $\text{B}^d_c(n)$ if and only if $d \in \{0, 1, n-1, n\}$.

**Proof.** First we prove the positive result for $d \in \{0, 1, n-1, n\}$. If $d = 0$, then $c = 0$ and $\text{FKM}^0_c(n) = \text{ap}(0^n) = 0$ is trivially a universal cycle for this case. If $d = 1$, then $c = 0$ or $c = 1$. In the first case $\text{FKM}^1_c(n) = \text{ap}(0^n) \cdot \text{ap}(0^{n-1}) = 0^n1$ is a universal cycle for $\text{B}^1_c(n)$. In the second case $\text{FKM}^1_c(n) = \text{ap}(1^n) = 1$ is a universal cycle for $\text{B}^1_c(n)$. If $d = n$, then the result follows from Theorem 1. If $d = n - 1$, then $\text{FKM}^{n-1}_c(n)$ is precisely $\text{FKM}^n_c(n)$ with the final bit $\text{ap}(1^n) = 1$ removed. The inclusion of this extra 1 accounts for the one extra string $1^n$ in $\text{FKM}^n_c(n)$ so the result immediately follows.

Now we prove the negative result for $d \in \{2, 3, \cdots , n-2\}$. Consider the aperiodic necklace $0^{n-d-1}d \in \mathbb{N}^d_c(n)$. The next necklace in $\text{LEX}(\mathbb{N}^d_c(n))$ has prefix $0^{n-d-1}$ by Lemma 1. Also, since $d \leq n - 2$ we have $n - d - 1 \geq 1$. Thus, $0^{n-d-1}d \cdot 0^{n-d-1}$ appears as a substring in $\text{FKM}^n_c(n)$. However this string contains the length $n$ substring $1^d0^{n-d-1}$ \notin $\text{B}^d_c(n)$. Therefore, $\text{FKM}^d_c(n)$ is not a universal cycle for $\text{B}^d_c(n)$ for $d \in \{2, 3, \cdots , n-2\}$.

\[ \]
4. The lexicographically smallest universal cycle for $B^n_c(n)$

In this section, we prove that the universal cycle $FKM^n_c(n)$ has the property of being the lexicographically smallest universal cycle for $B^n_c(n)$. Thus, $FKM^n_c(n)$ corresponds to the Euler cycle in $G(B^n_c(n))$ with lexicographically minimal labels.

Theorem 2. $FKM^n_c(n)$ is the lexicographically smallest universal cycle among all universal cycles for $B^n_c(n)$.

Proof. Suppose there is a universal cycle $U = u_1u_2\ldots u_m$ for $B^n_c(n)$ that is lexicographically smaller than $FKM^n_c(n) = a_1a_2\ldots a_m$. Let $q$ be the smallest index such that $u_q = 0$ and $a_q = 1$. Notice that $FKM^n_c(n)$ begins with $0^{n-c}1^c$ and no other universal cycle for $B^n_c(n)$ can have a lexicographically smaller prefix. Thus $q > n$. If $q = m$ then $U$ clearly misses the string $1^n$, a contradiction. Thus, we can also assume that $q < m$. Now, consider the length $n$ strings $s = a_{q-n+1}\ldots a_{q-1}$1 and $s' = u_{q-n+1}\ldots u_{q-1}0$. Since we just showed that $q < m$ we know that $s \neq 1^n$.

To complete the proof, we demonstrate that $s'$ appears before $s$ in $FKM^n_c(n)$, which implies that $s'$ appears more than once as a substring in $U$—a contradiction to $U$ being a universal cycle. Let $\alpha$ denote the necklace representative of $s$ and let $\beta$ denote the necklace representative of $s'$. Clearly $\beta < \alpha$. Stepping through the cases in the proof of Theorem 1, observe $s$ will be found starting within one of the following two substrings:

$$1^iap(\alpha) \text{ or } 1^iap(\gamma),$$

where $\gamma$ is the lexicographically smallest necklace that starts with some prefix of $\alpha$ and suffix of $s$. Thus $\gamma \leq \alpha$. Similarly $s'$ will be found starting within one of the substrings $1^iap(\beta)$ or $1^iap(\gamma')$, where $\gamma'$ is the lexicographically smallest necklace that starts with some prefix of $\beta$ and suffix of $s'$. Hence, $\gamma' \leq \beta$. Thus, since $\beta < \alpha$ and $u_q < a_q$, we have $\gamma' \leq \beta < \gamma \leq \alpha$. Therefore the only way that $s'$ does not appear before $s$ as a substring in $FKM^n_c(n)$ is if:

1. $\beta$ appears immediately before $\gamma$ in $\text{LEX}(N^d_c(n))$,
2. both $s$ and $s'$ start within the prefix $1^i$ of $1^iap(\gamma)$ and
3. $s$ starts before $s'$.

However, since $s$ and $s'$ have the same length $n-1$ prefix, the only possible string $s$ can be is $1^n$. But we have already ruled this case out, and hence $s'$ must appear before $s$ in $FKM^n_c(n)$. \qed

5. An efficient algorithm to construct minimum-weight universal cycles

In [3], Cattell et al. present a recursive necklace generation framework to generate prenecklaces, Lyndon words, or necklaces of length $n$. The basic idea is to recursively extend a prenecklace $\alpha = a_1a_2\ldots a_{t-1}$ to a length $t$ prenecklace in all possible ways. This is done efficiently by maintaining a variable $p$ which is the length of the longest prefix of $\alpha$ that is a Lyndon word. This algorithm can easily be adapted to satisfy a minimum weight constraint $c$ by maintaining an additional variable $w$ to store the current weight of $\alpha$. If $c - w = n - t + 1$, then the only way $\alpha$ can be extended to satisfy the weight constraint is by appending a 1. Pseudocode for this algorithm Gen$(t, p, w)$ is given in Algorithm 1. The necklaces are precisely the prenecklaces where $n \mod p = 0$. To generate $FKM^n_c(n)$, the aperiodic prefix $a_1a_2\ldots a_p$ is outputted for each necklace generated. The initial call is Gen$(1, 1, 0)$ with $a_0$ initialized to 0.
Algorithm 1 Algorithm to generate $\text{FKM}_c^n(n)$.

1: procedure GEN($t, p, w$)
2:     if $t > n$ then
3:         if $n \mod p = 0$ then PRINT($a_1a_2\ldots a_p$)
4:     else
5:         $a_t \leftarrow 0$
6:         if $(a_{t-p} = 0 \text{ and } c - w < n - t + 1)$ then GEN($t + 1, p, w$)
7:     else
8:         if $a_{t-p} = 1$ then GEN($t+1, p, w+1$)
9:         else GEN($t+1, t, w+1$)

To illustrate the algorithm, Figure 4 shows the recursive computation tree to generate the prenecklaces in $\mathcal{B}_{2}^{5}(5)$; the necklaces are highlighted in bold. A complete C implementation is given in the Appendix.

![Figure 4: Computation tree of Gen($t, p, w$) to generate the prenecklaces in $\mathcal{B}_{2}^{5}(5)$.

5.1. Analysis:

In the analysis we assume that $n > 0$ and $0 \leq w \leq n$. Ignoring the time required to output the bits of the universal cycle $\text{FKM}_c^n(n)$, each recursive call of Gen($t, p, w$) requires a constant amount of work. Thus, the overall running time to generate and output $\text{FKM}_c^n(n)$ is proportional to the number of nodes in the recursive computation tree, denoted by $\text{CompTree}(n)$. We show that $\text{CompTree}(n)$ is bounded by some constant times $|\text{FKM}_c^n(n)|$.

Let $N(n, w)$, $L(n, w)$ and $P(n, w)$ denote the cardinality of $N(n, w)$, $L(n, w)$ and $P(n, w)$ re-
respectively. Let \( P_0(n, w) \) and \( P_1(n, w) \) denote the cardinality of the set of length \( n \) binary prenecklaces with weight \( w \) that ends with 0 and 1 respectively. By partitioning the prenecklaces in \( P(n, w) \) that end with 1 into necklaces and non-necklaces, the following upper bound was given in [21]:

**Lemma 7.** \( P_1(n, w) \leq N(n, w) + L(n, w) \).

**Lemma 8.** \( P_0(n, w) \leq N(n, w + 1) \).

**Proof.** Consider a prenecklace in \( P(n, w) \) that ends with 0. It is easy to verify that replacing the last 0 with a 1 yields a string in \( N(n, w + 1) \). Such a mapping is clearly 1-1.

Upper bounds for \( N(n, w) \) and \( L(n, w) \) in terms of \( \binom{n}{w} \) have also been given in [21]:

\[
L(n, w) \leq \frac{1}{n} \binom{n}{w},
\]

\[
N(n, w) \leq 2L(n, w) \leq \frac{2}{n} \binom{n}{w}.
\]

**Lemma 9.** \( \text{CompTree}(n) \leq 5 \cdot |\text{FKM}^n_c(n)| \).

**Proof.** Since there is no dead end in the computation tree (each branch ends with a length \( n \) prenecklace), \( \text{CompTree}(n) \) is bounded by \( n \) times the number of leaves (prenecklaces generated). Thus:

\[
\text{CompTree}(n) \leq n \cdot \sum_{i=c}^{n} P(n, i)
\]

\[
= n \cdot \left( \sum_{i=c}^{n} P_0(n, i) + \sum_{i=c}^{n} P_1(n, i) \right)
\]

\[
= n \cdot \left( \sum_{i=c}^{n-1} P_0(n, i) + P_0(n, n) + \sum_{i=c}^{n} P_1(n, i) \right)
\]

\[
= n \cdot \left( \sum_{i=c}^{n-1} P_0(n, i) + 0 + \sum_{i=c}^{n} P_1(n, i) \right)
\]

\[
\leq n \cdot \left( \sum_{i=c}^{n-1} N(n, i + 1) + \sum_{i=c}^{n} (N(n, i) + L(n, i)) \right)
\]

\[
\leq n \cdot \left( \sum_{i=c}^{n-1} \frac{2}{n} \binom{n}{i+1} + \sum_{i=c}^{n} \left( \frac{2}{n} \binom{n}{i} + \frac{1}{n} \binom{n}{i} \right) \right)
\]

\[
= n \cdot \left( \sum_{i=c}^{n} \frac{2}{n} \binom{n}{i} + \sum_{i=c}^{n} \frac{3}{n} \binom{n}{i} \right)
\]

\[
\leq 5 \cdot \sum_{i=c}^{n} \binom{n}{i}
\]

\[
= 5 \cdot |\text{B}^n_c(n)| = 5 \cdot |\text{FKM}^n_c(n)|.
\]

\[\square\]
This immediately gives us the following result.

**Theorem 3.** $\text{FKM}_c^n(n)$ can be constructed in constant amortized time per bit using $O(n)$ space.

From Theorem 2, the universal cycle $\text{FKM}_c^n(n)$ corresponds to the Euler cycle with lexicographically minimal labels for $G(\text{B}_c^n(n))$. The following corollary follows immediately.

**Corollary 10.** An Euler cycle of lexicographically minimal labels for $G(\text{B}_c^n(n))$ can be constructed in $O(m)$ time using $O(n)$ space, where $m$ is the number of edges in $G(\text{B}_c^n(n))$.

**References**


6. Appendix – C code

```c
#include <stdio.h>
int n, c, a[100];

// Generate the lexicographically smallest universal cycle (de Bruijn sequence) for binary strings of length "n" with minimum weight "c"
void Gen(int t, int p, int w) {
    int i;
    if (t > n) {
        if (n%p == 0) {
            for (i=1; i <= p; i++) printf("%d", a[i]);
            printf(" ");
        }
    } else {
        // Append 0
        a[t] = 0;
        if (a[t-p] == 0 && c-w < n-t+1) Gen(t+1, p, w);

        // Append 1
        a[t] = 1;
        if (a[t-p] == 1) Gen(t+1, p, w+1);
        else Gen(t+1, t, w+1);
    }
}

int main() {
    printf("Enter n c: ");
    scanf("%d %d", &n, &c);
    a[0] = 0;
    if (n >= c) Gen(1, 1, 0);
    printf("\n");
}
```