Efficient Construction of Long Orientable Sequences

Daniel Gabric
University of Guelph, Canada

Joe Sawada
University of Guelph, Canada

Abstract

An orientable sequence of order n is a cyclic binary sequence such that each length-n substring appears at most once *in either direction*. Maximal length orientable sequences are known only for $n \leq 7$, and a trivial upper bound on their length is $2^{n-1} - 2^{\lfloor (n-1)/2 \rfloor}$. This paper presents the first efficient algorithm to construct orientable sequences with asymptotically optimal length; more specifically, our algorithm constructs orientable sequences via cycle-joining and a successor-rule approach requiring $O(n)$ time per symbol and $O(n)$ space. This answers a longstanding open question from Dai, Martin, Robshaw, Wild [Cryptography and Coding III (1993)]. Our sequences are applied to find new longest-known orientable sequences for $n \leq 20$.

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Orientable sequences were introduced by Dai, Martin, Robshaw, and Wild [6] with applications related to robotic position sensing. In particular, consider an autonomous robot with limited sensors. To determine its location on a cyclic track labeled with black and white squares, the robot scans a window of n squares directly beneath it. For the position *and* orientation to be uniquely determined, the track must designed with the property that each length n window can appear at most once in *either direction*. A cyclic binary sequence (track) with such a property is called an *orientable sequence* of order n (an $\mathcal{OS}(n)$). By this definition, an orientable sequence does not contain a length-n substring that is a palindrome.

Example 1 Consider $\mathcal{S} = 001011$. In the forward direction, including the wraparound, $\mathcal{S}$ contains the six 5-tuples 00101, 01011, 10110, 01100, 11001, and 10010; in the reverse direction $\mathcal{S}$ contains 11010, 10100, 01001, 10011, 00110, and 01101. Since each substring is unique, $\mathcal{S}$ is an $\mathcal{OS}(5)$ with length (period) six.

Orientable sequences do not exist for $n < 5$, and somewhat surprisingly, the maximum length M_n of an $\mathcal{OS}(n)$ is known only for $n = 5, 6, 7$. Since the number of palindromes of length n is $2^{\lfloor (n+1)/2 \rfloor}$, a trivial upper bound on M_n is $(2^n - 2^{\lfloor (n+1)/2 \rfloor})/2 = 2^{n-1} - 2^{\lfloor (n-1)/2 \rfloor}$.

In addition to providing a tighter upper bound, Dai, Martin, Robshaw, and Wild [6] provide a lower bound on M_n by demonstrating the *existence* of $\mathcal{OS}(n)$s via cycle-joining with length L_n (defined in Section 1.1) asymptotic to their upper bound. They conclude by stating the following open problem relating to orientable sequences whose lengths (periods) attain the lower bound. See Section 1.1 for the explicit upper and lower bounds.

> *We note that the lower bound on the maximum period was obtained using an existence construction . . . It is an open problem whether a more practical procedure exists for the construction of orientable sequences that have this asymptotically optimal period.*

Recently, some progress was made in this direction by Mitchell and Wild [25]. They apply Lempel's lift [22] to obtain an $\mathcal{OS}(n)$ recursively from an $\mathcal{OS}(n-1)$. This construction can generate orientable sequences in $O(1)$-amortized time per symbol; however, it requires exponential space, and there is an exponential time delay before the first bit can be output. Furthermore, they state that their work "only *partially* answer the question, since the periods/lengths of the sequences produced are not asymptotically optimal."

Main result: By developing a parent rule to define a cycle-joining tree, we construct an $\mathcal{OS}(n)$ of length L_n in $O(n)$ time per bit using $O(n)$ space.

Outline. In Section 1.1, we review the lower bound L_n and upper bound U_n from [6]. In Section 2, we present necessary background definitions and notation, including a review of the cycle-joining technique. In Section 3, we provide a parent rule for constructing a cycle-joining tree composed of "reverse-disjoint" cycles. This leads to our $O(n)$ time per bit construction of orientable sequences of length L_n. In Section 4 we discuss the algorithmic techniques used to extend our constructed orientable sequences to find longer ones for $n \leq 20$. We conclude in Section 5 with a summary of our results and directions for future research. An implementation of our construction is available for download at `http://debruijnsequence.org/db/orientable`.

1.1 Bounds on M_n

Dai, Martin, Robshaw, and Wild [6] gave a lower bound L_n and an upper bound U_n on the maximum length M_n of an $\mathcal{OS}(n)$.[1] Their lower bound L_n is the following, where μ is the Möbius function:

$$L_n = \left(2^{n-1} - \frac{1}{2}\sum_{d|n}\mu(n/d)\frac{n}{d}H(d)\right), \quad \text{where} \quad H(d) = \frac{1}{2}\sum_{i|d} i\left(2^{\lfloor\frac{i+1}{2}\rfloor} + 2^{\lfloor\frac{i}{2}\rfloor+1}\right).$$

Their upper bound U_n is the following:[1]

$$U_n = \begin{cases} 2^{n-1} - \frac{41}{9}2^{\frac{n}{2}-1} + \frac{n}{3} + \frac{16}{9} & \text{if } n \bmod 4 = 0, \\ 2^{n-1} - \frac{31}{9}2^{\frac{n-1}{2}} + \frac{n}{3} + \frac{19}{9} & \text{if } n \bmod 4 = 1, \\ 2^{n-1} - \frac{41}{9}2^{\frac{n}{2}-1} + \frac{n}{6} + \frac{20}{9} & \text{if } n \bmod 4 = 2, \\ 2^{n-1} - \frac{31}{9}2^{\frac{n-1}{2}} + \frac{n}{6} + \frac{43}{18} & \text{if } n \bmod 4 = 3. \end{cases}$$

These bounds are calculated in Table 1 for n up to 20. This table also illustrates the length R_n of the $\mathcal{OS}(n)$ produced by the recursive construction by Mitchell and Wild [25], starting from an initial orientable sequence of length 80 for $n = 8$. The column labeled L_n^* indicates the longest known orientable sequences we discovered by applying a combination of techniques (discussed in Section 4) to our orientable sequences of length L_n.

n	R_n	L_n	L_n^*	U_n
5	-	0	**6**	6
6	-	6	**16**	17
7	-	14	**36**	40
8	80	48	92	96
9	161	126	174	206
10	322	300	416	443
11	645	682	844	918
12	1290	1530	1844	1908
13	2581	3276	3700	3882
14	5162	6916	7694	7905
15	10325	14520	15394	15948
16	20650	29808	31483	32192
17	41301	61200	63135	64662
18	82602	124368	128639	129911
19	165205	252434	257272	260386
20	330410	509220	519160	521964

Table 1 Lower bounds R_n, L_n, L_n^* and upper bound U_n for M_n.

1.2 Related work

Recall the problem of determining a robot's position and orientation on a track. Suppose now that we allow the track to be non-cyclic. That is, the beginning of the track and the end of the track are not connected. Then the corresponding sequence that allows one to determine orientation and position is called an *aperiodic orientable sequence*. One does not

[1] These bounds correspond to $\tilde{L}_n$ and $\tilde{U}_n$, respectively, as they appear in [6].

consider the substrings in the wraparound for this variation of an orientable sequence. Note that one can always construct an aperiodic $\mathcal{OS}(n)$ from a cyclic $\mathcal{OS}(n)$ by taking the cyclic $\mathcal{OS}(n)$ and appending its prefix of length $n-1$ to the end. See the paper by Burns and Mitchell [4] for more on aperiodic orientable sequences, which they call *aperiodic 2-orientable window sequences*. Alhakim et al. [2] generalize the recursive results of Mitchell and Wild [25] to construct orientable sequences over an alphabet of arbitrary size $k \geq 2$; they also generalize the upper bound, by Dai et al. [6], on the length of an orientable sequence. Rampersad and Shallit [26] showed that for every alphabet size $k \geq 2$ there is an infinite sequence such that for every sufficiently long substring, the reversal of the substring does not appear in the sequence. Fleischer and Shallit [11] later reproved the results of the previous paper using theorem-proving software. See [5, 23] for more work on sequences avoiding reversals of substrings.

2 Preliminaries

Let $\mathbf{B}(n)$ denote the set of all length-n binary strings. Let $\alpha = \mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n \in \mathbf{B}(n)$ and $\gamma = \mathtt{g}_1\mathtt{g}_2\cdots\mathtt{g}_m \in \mathbf{B}(m)$ for some $m, n \geq 0$. Throughout this paper, we assume $0 < 1$ and use lexicographic order when comparing two binary strings. More specifically, we say that $\alpha < \gamma$ either if α is a prefix of γ or if $\mathtt{a}_\mathtt{i} < \mathtt{g}_\mathtt{i}$ for the smallest i such that $\mathtt{a}_\mathtt{i} \neq \mathtt{g}_\mathtt{i}$. The *weight* (density) of a binary string is the number of 1s in the string. Let $\overline{\mathtt{a}}_i$ denote the complement of bit $\mathtt{a}_i$. Let α^R denote the reversal $\mathtt{a}_n\cdots\mathtt{a}_2\mathtt{a}_1$ of α; α is a *palindrome* if $\alpha = \alpha^R$. For $j \geq 1$, let α^j denote j copies of α concatenated together. If $\alpha = \beta^j$ for some non-empty string β and some $j > 1$, then α is said to be *periodic*[2]; otherwise, α is said to be *aperiodic*[3]. A *necklace class* is an equivalence class of strings under rotation; let $[\alpha]$ denote the set of strings in α's necklace class. We say α is a *necklace* if it is the lexicographically smallest string in $[\alpha]$. Let $\mathbf{N}(n)$ denote the set of length-n necklaces. A *bracelet class* is an equivalence class of strings under rotation and reversal; let $\langle\alpha\rangle$ denote the set of strings in α's bracelet class. Thus, $\langle\alpha\rangle = [\alpha] \cup [\alpha^R]$. We say α is a *bracelet* if it is the lexicographically smallest string in $\langle\alpha\rangle$.

A necklace α is *palindromic* if it belongs to the same necklace class as α^R, i.e., both α and α^R belong to $[\alpha]$. By this definition, a palindromic necklace is necessarily a bracelet. If a necklace or bracelet is not palindromic, it is said to be *apalindromic*. Let $\mathbf{A}(n)$ denote the set of all apalindromic bracelets of order n. Table 2 lists all 60 necklaces of length $n = 9$ partitioned into apalindromic necklace pairs and palindromic necklaces. The apalindromic necklace pairs belong to the same bracelet class, and the first string in each pair is an apalindromic bracelet. Thus, $|\mathbf{A}(9)| = 14$. In general, $|\mathbf{A}(n)|$ is equal to the number of necklaces of length n minus the number of bracelets of length n; for $n = 6, 7, \ldots 15$, this sequence of values $|\mathbf{A}(n)|$ is given by 1, 2, 6, 14, 30, 62, 128, 252, 495, 968 and it corresponds to sequence A059076 in The On-Line Encyclopedia of Integer Sequences [31]. Apalindromic bracelets have been studied previously in the context of efficiently ranking/unranking bracelets [1]. One can test whether a string is an apalindromic bracelet in linear time using linear space; see Theorem 1.

▶ **Theorem 1.** *One can determine whether a string α is in $\mathbf{A}(n)$ in $O(n)$ time using $O(n)$ space.*

Proof. A string α will belong to $\mathbf{A}(n)$ if α is a necklace and the necklace of $[\alpha^R]$ is lexicographically larger than α. These tests can be computed in $O(n)$ time using $O(n)$ space [3]. ◀

▶ **Lemma 2.** *A necklace α is palindromic if and only if there exists palindromes β_1 and β_2 such that $\alpha = \beta_1\beta_2$.*

Proof. Suppose α is a palindromic necklace. By definition, it is equal to the necklace of $[\alpha^R]$. Thus, there exist strings β_1 and β_2 such that $\alpha = \beta_1\beta_2 = (\beta_2\beta_1)^R = \beta_1^R\beta_2^R$. Therefore, $\beta_1 = \beta_1^R$ and $\beta_2 = \beta_2^R$, which means β_1 and β_2 are palindromes. Suppose there exists two palindromes β_1 and β_2 such that $\alpha = \beta_1\beta_2$. Since β_1 and β_2 are palindromic, we have that $\alpha^R = (\beta_1\beta_2)^R = \beta_2^R\beta_1^R = \beta_2\beta_1$. So α belongs to the same necklace class as α^R and hence is palindromic. ◀

▶ **Corollary 3.** *If $\alpha = 0^s\beta$ is a palindromic bracelet such that the string β begins and ends with 1 and does not contain 0^s as a substring, then β is a palindrome.*

---

[2] Periodic strings are sometimes called *powers* in the literature. The term *periodic* is sometimes used to denote a string of the form $(\alpha\beta)^i\alpha$ where α is non-empty, β is possibly empty, and $i \geq 1$.

[3] Aperiodic strings are sometimes called *primitive* in the literature.

Apalindromic necklace pairs	Palindromic necklaces		
000001011 , 000001101	000000000	00010011	00111011
000010011 , 000011001	000000001	00010111	00111111
000010111 , 000011101	00000011	00011011	01010111
000100101 , 000101001	00000101	00011111	01010111
000100111 , 000111001	00000111	00100101	01011111
000101011 , 000110101	00001001	00100111	01101111
000101111 , 000111101	00001111	00101011	01101111
000110111 , 000111011	00010001	00101011	01110111
001001011 , 001001101	00010101	00101101	01111111
001010111 , 001110101	00011011	00110011	11111111
001011011 , 001101101	00011111	00110101	
001011111 , 001111101			
001101111 , 001111011			
010110111 , 010111011			

Table 2 A listing of all 60 necklaces in $\mathbf{N}(9)$ partitioned into apalindromic necklace pairs and palindromic necklaces. The first column of the apalindromic necklaces corresponds to the 14 apalindromic bracelets $\mathbf{A}(9)$.

2.1 Cycle-joining

Given $\mathbf{S} \subseteq \mathbf{B}(n)$, a *universal cycle* U for $\mathbf{S}$ is a cyclic sequence of length $|\mathbf{S}|$ that contains each string in $\mathbf{S}$ as a substring (exactly once). Thus, an orientable sequence is a universal cycle. If $\mathbf{S} = \mathbf{B}(n)$ then U is known as a *de Bruijn sequence*. Given a universal cycle U for $\mathbf{S}$, a *UC-successor* for U is a function $f : \mathbf{S} \to \{0, 1\}$ such that $f(\alpha)$ is the symbol following α in U.

Cycle-joining is perhaps the most fundamental technique applied to construct universal cycles; for some applications, see [8, 9, 10, 12, 14, 16, 17, 29, 30]. If $\mathbf{S}$ is closed under rotation, then it can be partitioned into necklace classes (cycles); each cycle is disjoint. Let $\alpha = \mathtt{a}_1\mathtt{a}_2 \cdots \mathtt{a}_n$ and $\hat{\alpha} = \bar{\mathtt{a}}_1\mathtt{a}_2 \cdots \mathtt{a}_n$; we say $(\alpha, \hat{\alpha})$ is a *conjugate pair*. Two disjoint cycles can be joined if they each contain one string of a *conjugate pair* as a substring. This approach resembles Hierholzer's algorithm to construct an Euler cycle in an Eulerian graph [15].

Example 2 Consider disjoint subsets $\mathbf{S}_1 = [011111] \cup [001111]$ and $\mathbf{S}_2 = [010111] \cup [010101]$. Then $U_1 = 110011\underline{110111}$ is a universal cycle for $\mathbf{S}_1$ and $U_2 = 01\underline{010111}$ is a universal cycle for $\mathbf{S}_2$. Since $(110111, 010111)$ is a conjugate pair, $U = 110011\underline{110111} \cdot 01\underline{010111}$ is a universal cycle for $\mathbf{S}_1 \cup \mathbf{S}_2$.

If all necklace cycles can be joined via conjugate pairs to form a cycle-joining tree, then the tree defines a universal U for $\mathbf{S}$ with a corresponding UC-successor (see Section 3 for an example).

For most universal cycle constructions, a corresponding cycle-joining tree can be defined by a rather simple *parent rule*. For example, when $\mathbf{S} = \mathbf{B}(n)$, the following are perhaps the *simplest* parent rules that define how to construct cycle-joining trees with nodes corresponding to $\mathbf{N}(n)$ [13, 27].

- **Last-0**: rooted at 1^n and the parent of every other node $\alpha \in \mathbf{N}(n)$ is obtained by flipping the last 0.
- **First-1**: rooted at 0^n and the parent of every other node $\alpha \in \mathbf{N}(n)$ is obtained by flipping the first 1.
- **Last-1**: rooted at 0^n and the parent of every other node $\alpha \in \mathbf{N}(n)$ is obtained by flipping the last 1.
- **First-0**: rooted at 1^n and the parent of every other node $\alpha \in \mathbf{N}(n)$ is obtained by flipping the first 0.

These rules induce the cycle-joining trees T_1, T_2, T_3, T_4 illustrated in Figure 1 for $n = 6$. Note that for T_3 and T_4, the parent of a node α is obtained by first flipping the highlighted bit and then rotating the string to its lexicographically least

rotation to obtain a necklace. Each node α and its parent β are joined by a conjugate pair, where the highlighted bit in α is the first bit in one of the conjugates. For example, the nodes $\alpha = 011011$ and $\beta = 001011$ in T_2 from Figure 1 are joined by the conjugate pair $(110110, 010110)$.

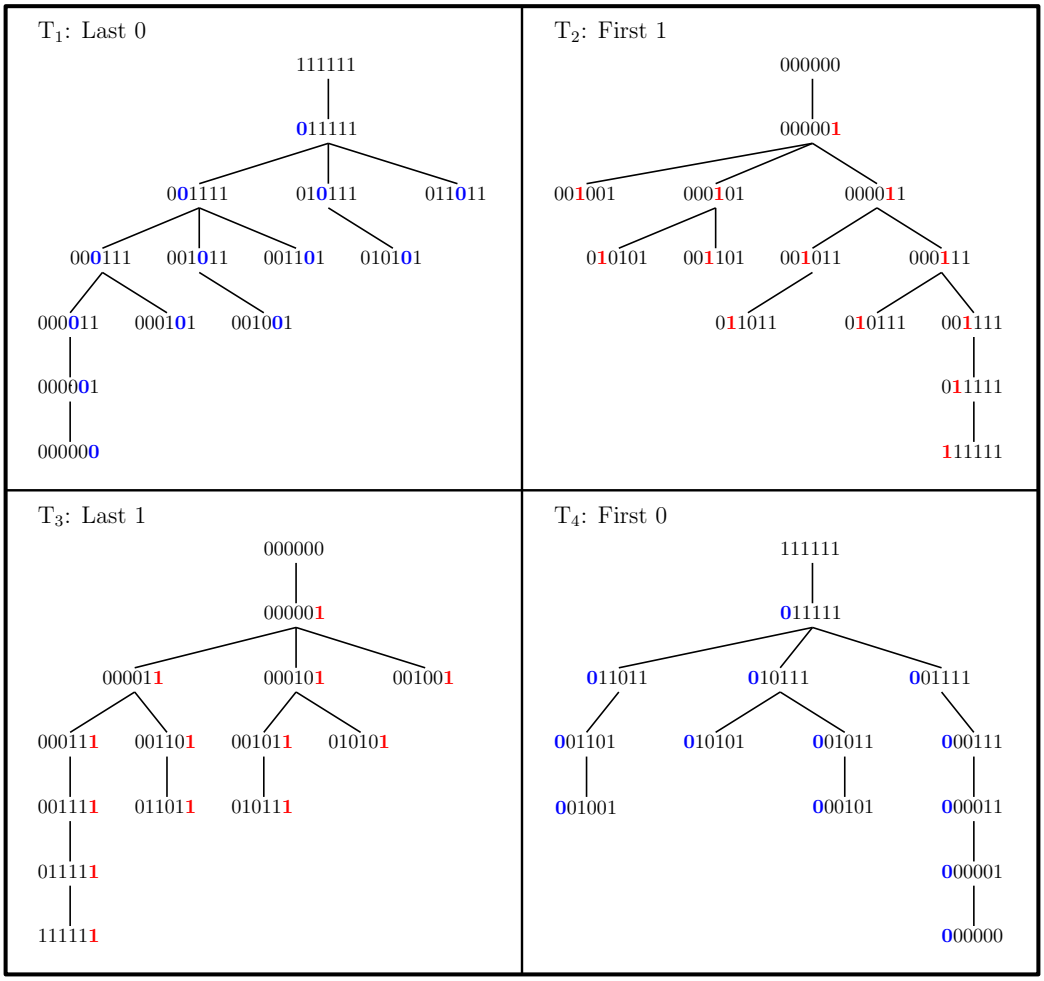

Figure 1 Cycle-joining trees for $\mathbf{B}(6)$ from simple parent rules.

3 An efficient cycle-joining construction of orientable sequences

Consider the set of apalindromic bracelets $\mathbf{A}(n) = \{\alpha_1, \alpha_2, \ldots, \alpha_t\}$. Recall, that each palindromic bracelet is a necklace. Let $\mathbf{S}(n) = [\alpha_1] \cup [\alpha_2] \cup \cdots \cup [\alpha_t]$. From [6], we have $|\mathbf{S}(n)| = L_n$. By its definition, there is no string $\alpha \in \mathbf{S}(n)$ such that $\alpha^R \in \mathbf{S}(n)$. Thus, a universal cycle for $\mathbf{S}(n)$ is an $\mathcal{OS}(n)$. For the rest of this section, assume $n \geq 8$.

To construct a cycle-joining tree with nodes $\mathbf{A}(n)$, we apply a combination of three of the four simple parent rules described in the previous section. First, we demonstrate that there is no such parent rule, using at most two rules in combination. Observe, there are no necklaces in $\mathbf{A}(n)$ with weight 0, 1, 2, $n-2$, $n-1$, or, n. Thus, $0^{n-4}1011$ and $0^{n-5}10011$ are both necklaces in $\mathbf{A}(n)$ with minimal weight three. Similarly, 00101^{n-4} and 001101^{n-5} are necklaces in $\mathbf{A}(n)$ with maximal weight $n-3$. Therefore, when considering a *simple* parent rule for a cycle-joining tree with nodes $\mathbf{A}(n)$, the rule must be able to flip a 0 to a 1, or a 1 to a 0, i.e., the rule must include one of First-0 or Last-0, and one of First-1 and Last-1.

Let $\alpha = \mathtt{a}_1\mathtt{a}_2 \cdots \mathtt{a}_n$ denote a necklace in $\mathbf{A}(n)$; it must begin with 0 and end with 1. Then let

- first1(α) be the necklace $\mathtt{a}_1 \cdots \mathtt{a}_{i-1}\mathbf{0}\mathtt{a}_{i+1} \cdots \mathtt{a}_n$, where i is the index of the first 1 in α;
- last1(α) be the necklace of $[\mathtt{a}_1\mathtt{a}_2 \cdots \mathtt{a}_{n-1}\mathbf{0}]$;
- first0(α) be the necklace of $[\mathbf{1}\mathtt{a}_2 \cdots \mathtt{a}_n]$;
- last0(α) be the necklace $\mathtt{a}_1 \cdots \mathtt{a}_{j-1}\mathbf{1}\mathtt{a}_{j+1} \cdots \mathtt{a}_n$, where j is the index of the last 0 in α.

Note that first1(α) and last0(α) are necklaces (easily observed by definition) obtained by flipping the i-th and j-th bit in α, respectively; last1(α) and first0(α) are the result of flipping a bit and rotating the resulting string to obtain a necklace. The next example illustrates that no two of the *simple* parent rules can be applied in combination to obtain a spanning tree with nodes in $\mathbf{A}(n)$.

Example 3 Suppose p(α) is a parent rule that applies a combination of the four simple parent rules to construct a cycle-joining tree with nodes $\mathbf{A}(n)$. The following examples are for $n = 10$ but generalize to larger n. In both cases, we see that at least three of the simple parent rules must be applied in p.

Suppose p does not use First-0; it must apply Last-0. Consider three apalindromic bracelets in $\mathbf{A}(10)$: $\alpha_1 = 0000001011$, $\alpha_2 = 0000010111$, and $\alpha_3 = 0011001011$. Clearly, first1(α_1), last1(α_1), and last0(α_1) are palindromic. Thus, α_1 must be the root. Both first1(α_2) and last0(α_2) are palindromic; thus, p must apply Last-1. Note last0(α_3) is palindromic and last1(α_3) = 0001100101 is not a bracelet; thus, p must apply First-1.

Suppose p does not use Last-0; it must apply First-0. Consider three apalindromic bracelets in $\mathbf{A}(10)$: $\beta_1 = 0000100011$, $\beta_2 = 0001001111$, and $\beta_3 = 0001100111$. Clearly, first1(β_1), last1(β_1), and first0(β) are palindromic. Thus, β_1 must be the root. Both first1(β_2) and first0(β_2) are palindromic; thus, p must apply Last-1. Both last1(β_3) and first0(β_3) are palindromic; thus, p must apply First-1.

Let r_n denote the apalindromic bracelet $0^{n-4}1011$.

Parent rule for cycle-joining $\mathbf{A}(n)$: Let r_n be the root. Let α denote a non-root node in $\mathbf{A}(n)$. Then

$$\text{par}(\alpha) = \begin{cases} \text{first1}(\alpha) & \text{if first1}(\alpha) \in \mathbf{A}(n); \\ \text{last1}(\alpha) & \text{if first1}(\alpha) \notin \mathbf{A}(n) \text{ and last1}(\alpha) \in \mathbf{A}(n); \\ \text{last0}(\alpha) & \text{otherwise.} \end{cases} \tag{1}$$

▶ **Theorem 4.** *The parent rule* par(α) *in* (1) *induces a cycle-joining tree with nodes* $\mathbf{A}(n)$ *rooted at* r_n.

Let $\mathbb{T}_n$ denote the cycle-joining tree with nodes $\mathbf{A}(n)$ induced by the parent rule in (1); Figure 2 illustrates $\mathbb{T}_9$. The proof of Theorem 4 relies on the following lemma.

▶ **Lemma 5.** *Let* $\alpha \neq r_n$ *be an apalindromic bracelet in* $\mathbf{A}(n)$. *If neither* first1(α) *nor* last1(α) *are in* $\mathbf{A}(n)$, *then the last* 0 *in* α *is at index* $n-2$ *or* $n-1$, *and both* last0(α) *and* last1(last0(α)) *are in* $\mathbf{A}(n)$.

Proof. Since α is an apalindromic bracelet, it must have the form $\alpha = 0^s1\beta01^v$ where $s, v \geq 1$ and $\beta0$ does not contain 0^{s+1} as a substring. Furthermore, $1\beta01^v < (1\beta01^v)^R$, which implies $\beta01^{v-1} < (\beta01^{v-1})^R$.

Suppose $v > 2$. Since $\text{last1}(\alpha) = 0^{s+1}1\beta01^{v-1}$ is not an apalindromic bracelet, we have $1\beta01^{v-1} \geq (1\beta01^{v-1})^R$. Thus, β begins with 1. Since $\text{first1}(\alpha) = 0^{s+1}\beta01^v$ is not an apalindromic bracelet, Lemma 2 implies $\beta01^v \geq (\beta01^v)^R$, contradicting the earlier observation that $\beta01^{v-1} < (\beta01^{v-1})^R$. Thus, the last 0 in α is at index $n-2$ or $n-1$.

Suppose $v = 1$ or $v = 2$. Let j be the index of the last 0 in α. Since α is a bracelet, it is straightforward to see that $\text{last0}(\alpha) = \mathtt{a}_1 \cdots \mathtt{a}_n$ is also a bracelet. If it is palindromic, Lemma 2 implies there exists an index i such that $\beta_1 = \mathtt{a}_1 \cdots \mathtt{a}_i$ and $\beta_2 = \mathtt{a}_{i+1} \cdots \mathtt{a}_n$ are both palindromes. However, flipping $\mathtt{a}_j$ to 0 to obtain α implies that α is greater than or equal to the necklace in $[\alpha^R]$, contradicting the assumption that α is an apalindromic bracelet. Thus, last0(α) is an apalindromic bracelet.

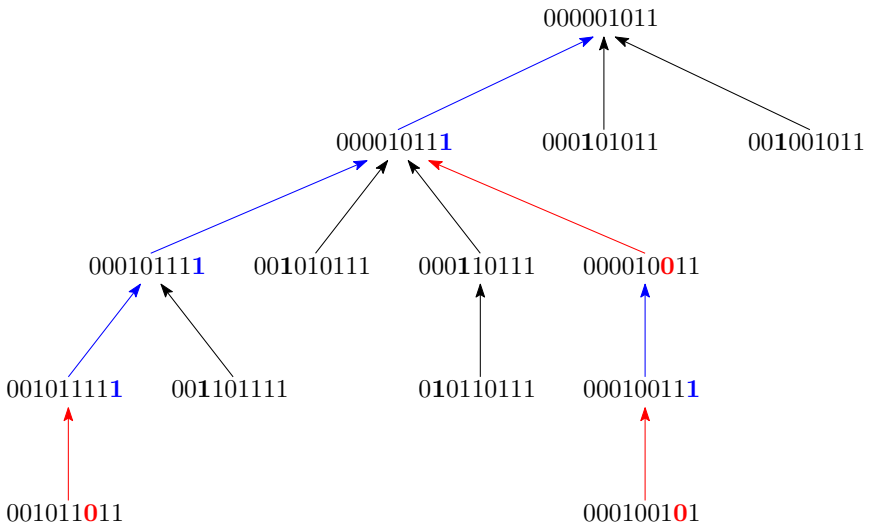

Figure 2 The cycle-joining tree $\mathbb{T}_9$. The black edges indicate that $\mathrm{par}(\alpha) = \mathrm{first1}(\alpha)$; the blue edges indicate that $\mathrm{par}(\alpha) = \mathrm{last1}(\alpha)$; the red edges indicate that $\mathrm{par}(\alpha) = \mathrm{last0}(\alpha)$.

Consider $\mathrm{last1}(\mathrm{last0}(\alpha)) = 0^{s+1}1\beta 1^v$. Let $\beta = \mathtt{b}_1\mathtt{b}_2\cdots\mathtt{b}_m$. Suppose that $m = 0$. Then $\mathrm{last1}(\mathrm{last0}(\alpha)) = 0^{s+1}1^{v+1} \implies \mathrm{last0}(\alpha) = 0^s1^{v+1}$. Since $v = 1$ or $v = 2$, we have that $\mathrm{last0}(\alpha) = 0^s11$ or $\mathrm{last0}(\alpha) = 0^s111$. Now α is the result of flipping one of the 1s in $\mathrm{last0}(\alpha)$ to a 0 and performing the appropriate rotation. But in every case, we end up with α being a palindromic necklace, a contradiction. Thus, assume $m \geq 1$. Suppose $\beta = 1^m$. Then, α is not an apalindromic bracelet, a contradiction. Suppose $\beta = 0^m$. If $v = 1$, then α is palindromic, a contradiction; if $v = 2$ then $\mathrm{last1}(\mathrm{last0}(\alpha)) = 0^{s+1}10^m11$ which is in $\mathbf{A}(n)$. For all other cases, β contains at least one 1 and at least one 0; $m \geq 2$. Since β does not contain 0^{s+1} as a substring, by Lemma 2, we must show that (i) $\mathtt{b}_1\cdots\mathtt{b}_m1^{v-1}$ is less than its reversal $1^{v-1}\mathtt{b}_m\cdots\mathtt{b}_1$, recalling that (ii) $\mathtt{b}_1\cdots\mathtt{b}_m01^{v-1}$ is less than its reversal $1^{v-1}0\mathtt{b}_m\cdots\mathtt{b}_1$. Let ℓ be the largest index of β such that $\mathtt{b}_\ell = 1$. Then $\mathtt{b}_{\ell+1}\cdots\mathtt{b}_\mathtt{m} = 0^{m-\ell}$; note that $\mathtt{b}_{\ell+1}\cdots\mathtt{b}_\mathtt{m}$ is the empty string when $\ell = m$. Suppose $v = 1$. From (ii), $\mathtt{b}_1 = 0$. By (ii) we have $\mathtt{b}_2\cdots\mathtt{b}_{\ell-1}10^{m-\ell} < 0^{m-\ell}1\mathtt{b}_{\ell-1}\cdots\mathtt{b}_2$. But this implies that $\mathtt{b}_2\cdots\mathtt{b}_{\mathtt{m}-\ell+1} = 0^{m-\ell}$. Therefore, we have $\mathtt{b}_1\cdots\mathtt{b}_\mathtt{m} = 0^{m-\ell+1}\mathtt{b}_{\mathtt{m}-\ell+2}\cdots\mathtt{b}_\mathtt{m} < 0^{m-\ell}1\mathtt{b}_{\ell-1}\cdots\mathtt{b}_1 = \mathtt{b}_\mathtt{m}\cdots\mathtt{b}_1$, hence (i) is satisfied. Suppose $v = 2$. If $\mathtt{b}_1 = 0$, then (i) is satisfied. Otherwise $\mathtt{b}_1 = 1$ and from (ii) $\mathtt{b}_2 = 0$. From (ii), we get that $\mathtt{b}_3\cdots\mathtt{b}_{\ell-1}10^{m-\ell} < 0^{m-\ell}\mathtt{b}_{\ell-1}\cdots\mathtt{b}_3$. This inequality implies that $\mathtt{b}_3\cdots\mathtt{b}_{\mathtt{m}-\ell+2} = 0^{m-\ell}$. Therefore, we have $\mathtt{b}_1\cdots\mathtt{b}_\mathtt{m}1 = 10^{m-\ell+1}\mathtt{b}_{\mathtt{m}-\ell+3}\cdots\mathtt{b}_\mathtt{m}1 < 10^{m-\ell}1\mathtt{b}_{\ell-1}\cdots\mathtt{b}_1 = 1\mathtt{b}_\mathtt{m}\cdots\mathtt{b}_1$, hence (i) is satisfied. Thus, $\mathrm{last1}(\mathrm{last0}(\alpha))$ is an apalindromic bracelet. ◀

Proof of Theorem 4. Let α be an apalindromic bracelet in $\mathbf{A}(n) \setminus \{r_n\}$. We demonstrate that the parent rule par from (1) induces a path from α to r_n, i.e., there exists an integer j such that $\mathrm{par}^j(\alpha) = r_n$. Note that r_n is the lexicographically smallest apalindromic bracelet of order n. By Lemma 5, $\mathrm{par}(\alpha) \in \mathbf{A}(n)$. In the first two cases of the parent rule, $\mathrm{par}(\alpha)$ is lexicographically smaller than α. If the third case applies, let $\alpha = 0^s1\beta$. From Lemma 5, $\mathrm{last1}(\mathrm{last0}(\alpha))$ is an apalindromic bracelet. Thus, $\mathrm{par}(\mathrm{par}(\alpha))$ is either $\mathrm{first1}(\mathrm{last0}(\alpha))$ or $\mathrm{last1}(\mathrm{last0}(\alpha))$; in each case the resulting apalindromic bracelet has 0^{s+1} as a prefix and is therefore lexicographically smaller than α. Therefore, the parent rule induces a path from α to r_n.

3.1 A successor rule

Each application of the parent rule $\mathrm{par}(\alpha)$ in (1) corresponds to a conjugate pair. For instance, consider the apalindromic bracelet $\alpha = 000101111$. The parent of α is obtained by flipping the last 1 to obtain 000101110 (see Figure 2). The corresponding conjugate pair is $(100010111, 000010111)$. Let $\mathbf{C}(n)$ denote the set of all strings belonging to a conjugate

pair in the cycle-joining tree $\mathbb{T}_n$. Then the following is a UC-successor for an $\mathcal{OS}(n)$:

$$f(\alpha) = \begin{cases} \bar{\mathtt{a}}_1 & \text{if } \alpha \in \mathbf{C}(n); \\ \mathtt{a}_1 & \text{otherwise.} \end{cases}$$

For example, if $\mathbf{C}(9)$ corresponds to the conjugate pairs to create the cycle-joining tree $\mathbb{T}_9$ shown in Figure 2, then the corresponding universal cycle is:

$$\begin{aligned} &0\underline{\mathbf{0}0001011111}00101101100101111001101111\underline{\mathbf{1}0001011}100101011100011011 \\ &10101101110000100111000100101000100110000101100100101100010101 1, \end{aligned}$$

where the two underlined strings belong to the conjugate pair $(\mathbf{1}00010111, \mathbf{0}00010111)$. In general, this rule requires exponential space to store the set $\mathbf{C}(n)$. However, in some cases, it is possible to test whether a string is in $\mathbf{C}(n)$ without pre-computing and storing $\mathbf{C}(n)$. In our UC-successor for an $\mathcal{OS}(n)$, we use Theorem 1 to avoid pre-computing and storing $\mathbf{C}(n)$, thereby reducing the space requirement from exponential in n to linear in n.

Successor-rule g to construct an $\mathcal{OS}(n)$ of length L_n

Let $\alpha = \mathtt{a}_1\mathtt{a}_2 \cdots \mathtt{a}_n \in \mathbf{S}(n)$ and let

- $\beta_1 = 0^{n-i}\mathbf{1}\mathtt{a}_2 \cdots \mathtt{a}_i$ where i is the largest index of α such that $\mathtt{a}_i = 1$ (first 1);
- $\beta_2 = \mathtt{a}_2\mathtt{a}_3 \cdots \mathtt{a}_n\mathbf{1}$ (last 1);
- $\beta_3 = \mathtt{a}_j\mathtt{a}_{j+1} \cdots \mathtt{a}_n\mathbf{0}1^{j-2}$ where j is the smallest index of α such that $\mathtt{a}_j = 0$ and $j > 1$ (last 0).

Let

$$g(\alpha) = \begin{cases} \bar{\mathtt{a}}_1 & \text{if } \beta_1 \text{ and } \mathrm{first1}(\beta_1) \text{ are in } \mathbf{A}(n); \\ \bar{\mathtt{a}}_1 & \text{if } \beta_2 \text{ and } \mathrm{last1}(\beta_2) \text{ are in } \mathbf{A}(n), \text{ and } \mathrm{first1}(\beta_2) \text{ is not in } \mathbf{A}(n); \\ \bar{\mathtt{a}}_1 & \text{if } \beta_3 \text{ and } \mathrm{last0}(\beta_3) \text{ are in } \mathbf{A}(n), \text{ and neither } \mathrm{first1}(\beta_3) \text{ nor } \mathrm{last1}(\beta_3) \text{ are in } \mathbf{A}(n); \\ \mathtt{a}_1 & \text{otherwise.} \end{cases}$$

Starting with any string in $\alpha \in \mathbf{S}(n)$, we can repeatedly apply $g(\alpha)$ to obtain the next bit in a universal cycle for $\mathbf{S}(n)$.

▶ **Theorem 6.** *The function g is a UC-successor for $\mathbf{S}(n)$ and generates an $\mathcal{OS}(n)$ with length L_n in $O(n)$-time per bit using $O(n)$ space.*

Proof. Consider $\alpha = \mathtt{a}_1\mathtt{a}_2 \cdots \mathtt{a}_n \in \mathbf{S}(n)$. If α belongs to some conjugate pair in $\mathbb{T}_n$, then it must satisfy one of three possibilities stepping through the parent rule in 1:

- Both β_1 and $\mathrm{first1}(\beta_1)$ must be in $\mathbf{A}(n)$. Note, β_1 is a rotation of α when $\mathtt{a}_1 = 1$, where $\mathtt{a}_1$ corresponds to the first one in β_1.
- Both β_2 and $\mathrm{last1}(\beta_2)$ must both be in $\mathbf{A}(n)$, but additionally, $\mathrm{first1}(\beta_2)$ can not be in $\mathbf{A}(n)$. Note, β_2 is a rotation of α when $\mathtt{a}_1 = 1$, where $\mathtt{a}_1$ corresponds to the last one in β_2.
- Both β_3 and $\mathrm{last0}(\beta_3)$ must both be in $\mathbf{A}(n)$, but additionally, both $\mathrm{first1}(\beta_3)$ and $\mathrm{last1}(\beta_3)$ can not be in $\mathbf{A}(n)$. Note, β_3 is a rotation of α when $\mathtt{a}_1 = 0$, where $\mathtt{a}_1$ corresponds to the last zero in β_3.

Thus, g is a UC-successor for $\mathbf{S}(n)$ and generates a cycle of length $|\mathbf{S}(n)| = L_n$. By Theorem 1, one can determine whether a string is in $\mathbf{A}(n)$ in $O(n)$ time using $O(n)$ space. Since there are a constant number of tests required by each case of the UC-successor g, the corresponding $\mathcal{OS}(n)$ can be computed in $O(n)$-time per bit using $O(n)$ space.

◀

4 Extending orientable sequences

The values from the column labeled L_n^* in Table 1 were found by extending an $\mathcal{OS}(n)$ of length L_n constructed in the previous section. Given an $\mathcal{OS}(n)$, $\mathtt{o}_1 \cdots \mathtt{o}_m$, the following approaches were applied to find longer $\mathcal{OS}(n)$s for $n \leq 20$:

1. For each index i, apply a standard backtracking search to see whether $\mathtt{o}_i \cdots \mathtt{o}_m \mathtt{o}_1 \cdots \mathtt{o}_{i-1}$ can be extended to a longer $\mathcal{OS}(n)$. We followed several heuristics: (a) find a maximal length extension for a given i, and then attempt to extend starting from index $i+1$; (b) find a maximal length extension over all i, then repeat; (c) find the "first" possible extension for a given i, and then repeat for the next index $i+1$. In each case, we repeat until no extension can be found for any starting index. This approach was fairly successful for even n, but found shorter extensions for n odd. Steps (a) and (b) were only applied to n up to 14 before the depth of search became infeasible.
2. Refine the search in the previous step so the resulting $\mathcal{OS}(n)$ of length m' has an odd number of 1s and at most one substring 0^{n-4}. Then we can apply the recursive construction by Mitchell and Wild [25] to generate an $\mathcal{OS}(n+1)$ with length $2m'$ or $2m'+1$. Then, starting from the sequences generated by recursion, we again apply the exhaustive search to find minor extensions (the depth of recursion is significantly reduced). This approach found significantly longer extensions to obtain $\mathcal{OS}(n+1)$s when $n+1$ is odd.

5 Future research directions

We present the first polynomial time and space algorithm to construct orientable sequences with asymptotically optimal length; it is a successor-rule-based approach that requires $O(n)$ time per bit and uses $O(n)$ space. This answers a long-standing open question by Dai, Martin, Robshaw, and Wild [6]. The following questions are currently being addressed. (1) How can our parent rule be generalized to an arbitrary alphabet like $\{C, G, A, T\}$. The notion of orientation is especially applicable in areas of computational biology [7, 18]. (2) Can the recent concatenation tree framework [27] be applied to construct our $\mathcal{OS}(n)$s in $O(1)$-amortized time per symbol? Additional interesting questions and problems include:

1. Can the lower bound of L_n for orientable sequences be improved?
2. Can small strings be inserted systematically into our constructed $\mathcal{OS}(n)$s to obtain longer orientable sequences?
3. Can our $\mathcal{OS}(n)$s be used to find longer *aperiodic* orientable sequences than reported in [4]?
4. A problem closely related to efficiently generating long $\mathcal{OS}(n)$s is the problem of *decoding* or *unranking* orientable sequences. That is, given an arbitrary length-n substring of an $\mathcal{OS}(n)$, efficiently determine where in the sequence this substring is located. There has been little to no progress in this area. Even in the well-studied area of de Bruijn sequences, only a few efficient decoding algorithms have been discovered. Most decoding algorithms are for specially constructed de Bruijn sequences; for example, see [24, 32]. It seems hard to decode an arbitrary de Bruijn sequence. The only de Bruijn sequence whose explicit construction was discovered before its decoding algorithm is the lexicographically least de Bruijn sequence, sometimes called the *Ford sequence* in the binary case, or the *Granddaddy sequence* (see Knuth [19]). Algorithms to efficiently decode this sequence were independently discovered by Kopparty et al. [21] and Kociumaka et al. [20]. Later, Sawada and Williams [28] provided a practical implementation.

References

1 Adamson, D., Gusev, V. V., Potapov, I., and Deligkas, A. Ranking bracelets in polynomial time. In *32nd Annual Symposium on Combinatorial Pattern Matching (CPM 2021)* (Dagstuhl, Germany, 2021), P. Gawrychowski and T. Starikovskaya, Eds., vol. 191 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, pp. 4:1–4:17.

2 Alhakim, A., Mitchell, C. J., Szmidt, J., and Wild, P. R. Orientable sequences over non-binary alphabets. *manuscript, 2023*.

3 Booth, K. S. Lexicographically least circular substrings. *Inform. Process. Lett. 10*, 4/5 (1980), 240–242.

4 Burns, J., and Mitchell, C. Position sensing coding schemes. In *Cryptography and Coding III (M.J.Ganley, ed.)* (1993), Oxford University Press, pp. 31–66.

5 CURRIE, J., AND LAFRANCE, P. Avoidability index for binary patterns with reversal. *Electronic J. Combinatorics 23*, (1) P1.36 (2016), 1–14.

6 DAI, Z.-D., MARTIN, K., ROBSHAW, B., AND WILD, P. Orientable sequences. In *Cryptography and Coding III (M.J.Ganley, ed.)* (1993), Oxford University Press, pp. 97–115.

7 DOMARATZKI, M. Hairpin structures defined by DNA trajectories. In *DNA Computing* (2006), C. Mao and T. Yokomori, Eds., vol. 4287 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, pp. 182–194.

8 ETZION, T. An algorithm for generating shift-register cycles. *Theoret. Comput. Sci. 44*, 2 (1986), 209–224.

9 ETZION, T. Self-dual sequences. *J. Combin. Theory Ser. A 44*, 2 (1987), 288–298.

10 ETZION, T., AND LEMPEL, A. Algorithms for the generation of full-length shift-register sequences. *IEEE Trans. Inform. Theory 30*, 3 (1984), 480–484.

11 FLEISCHER, L., AND SHALLIT, J. O. Words that avoid reversed factors, revisited. Arxiv preprint arXiv:1911.11704 [cs.FL], available at http://arxiv.org/abs/1911.11704, 2019.

12 FREDRICKSEN, H. A survey of full length nonlinear shift register cycle algorithms. *SIAM Review 24*, 2 (1982), 195–221.

13 GABRIC, D., SAWADA, J., WILLIAMS, A., AND WONG, D. A framework for constructing de Bruijn sequences via simple successor rules. *Discrete Math. 241*, 11 (2018), 2977–2987.

14 GABRIC, D., SAWADA, J., WILLIAMS, A., AND WONG, D. A successor rule framework for constructing k-ary de Bruijn sequences and universal cycles. *IEEE Trans. Inform. Theory 66*, 1 (2020), 679–687.

15 HIERHOLZER, C. Ueber die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren. *Math. Annalen 6* (1873), 30–32.

16 HUANG, Y. A new algorithm for the generation of binary de Bruijn sequences. *J. Algorithms 11*, 1 (1990), 44–51.

17 JANSEN, C. J. A., FRANX, W. G., AND BOEKEE, D. E. An efficient algorithm for the generation of DeBruijn cycles. *IEEE Trans. Inform. Theory 37*, 5 (1991), 1475–1478.

18 KARI, L., KONSTANTINIDIS, S., SOSÍK, P., AND THIERRIN, G. On hairpin-free words and languages. In *Developments in Language Theory* (2005), C. De Felice and A. Restivo, Eds., vol. 3572 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 296–307.

19 KNUTH, D. E. *The Art of Computer Programming, Volume 4A, Combinatorial Algorithms*. Addison-Wesley Professional, 2011.

20 KOCIUMAKA, T., RADOSZEWSKI, J., AND RYTTER, W. Computing k-th Lyndon word and decoding lexicographically minimal de Bruijn sequence. In *Combinatorial Pattern Matching*, A. S. Kulikov, S. O. Kuznetsov, and P. Pevzner, Eds., vol. 8486 of *Lecture Notes in Computer Science*. Springer International Publishing, 2014, pp. 202–211.

21 KOPPARTY, S., KUMAR, M., AND SAKS, M. Efficient indexing of necklaces and irreducible polynomials over finite fields. In *Automata, Languages, and Programming* (2014), J. Esparza, P. Fraigniaud, T. Husfeldt, and E. Koutsoupias, Eds., vol. 8572 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 726–737.

22 LEMPEL, A. On a homomorphism of the de Bruijn graph and its applications to the design of feedback shift registers. *IEEE Trans. Comput. C-19*, 12 (1970), 1204–1209.

23 MERCAŞ, R. On the aperiodic avoidability of binary patterns with variables and reversals. *Theoret. Comput. Sci. 682* (2017), 180–189. Special Issue on Languages and Combinatorics in Theory and Nature.

24 MITCHELL, C., ETZION, T., AND PATERSON, K. A method for constructing decodable de Bruijn sequences. *IEEE Trans. Inform. Theory 42*, 5 (1996), 1472–1478.

25 MITCHELL, C. J., AND WILD, P. R. Constructing orientable sequences. *IEEE Trans. Inform. Theory 68*, 7 (2022), 4782–4789.

26 RAMPERSAD, N., AND SHALLIT, J. O. Words that avoid reversed subwords. *J. Combin. Math. Combin. Comput. 54* (2005), 157–164.

27 SAWADA, J., SEARS, J., TRAUTRIM, A., AND WILLIAMS, A. Concatenation trees: A framework for efficient universal cycle and de Bruijn sequence constructions. Arxiv preprint arXiv:2308.12405 [math.CO], available at https://arxiv.org/abs/2308.12405, 2023.

28 SAWADA, J., AND WILLIAMS, A. Practical algorithms to rank necklaces, Lyndon words, and de Bruijn sequences. *J. Disc. Alg. 43* (2017), 95–110.

29 SAWADA, J., AND WILLIAMS, A. Constructing the first (and coolest) fixed-content universal cycle. *Algorithmica 85*, 6 (2023), 1754–1785.

30 SAWADA, J., AND WONG, D. Efficient universal cycle constructions for weak orders. *Discrete Math. 343*, 10 (2020), 112022.

31 SLOANE ET AL., N. J. A. OEIS Foundation Inc. (2024), The On-Line Encyclopedia of Integer Sequences, https://oeis.org.

32 TULIANI, J. De Bruijn sequences with efficient decoding algorithms. *Discrete Math. 226*, 1 (2001), 313–336.