Construction of orientable sequences in $O(1)$-amortized time per bit

Daniel Gabrić
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 bit and $O(n)$ space. This answers a longstanding open question from Dai, Martin, Robshaw, Wild [Cryptography and Coding III (1993)][1]. Applying a recent concatenation-tree framework, the same sequences can be generated in $O(1)$-amortized time per bit using $O(n^2)$ space. Our sequences are applied to find new longest-known (aperiodic) orientable sequences for $n \leq 20$.

[1] This paper was presented in part at Combinatorial Pattern Matching 2024 (CPM 2024) [13].

LIPICS 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 [7] 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 $n = 1$, and somewhat surprisingly, the maximum length M_n of an $\mathcal{OS}(n)$ is known only for $1 < n \leq 7$. 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 [7] provide a lower bound L_n on M_n by demonstrating the *existence* of $\mathcal{OS}(n)$s via cycle-joining with length L_n 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 2.1 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 [26]. 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 bit; 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. Then, by applying the recent theory of *concatenation trees* [28], the same orientable sequences can be constructed in $O(1)$-amortized time per bit using $O(n^2)$ space.

Outline. In Section 2, we present necessary background definitions and notation, a review of the lower bound L_n and upper bound U_n from [7], and 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 corresponding to asymmetric bracelets. This leads to our $O(n)$ time per bit construction of orientable sequences of length L_n. In Section 4, we present properties of the periodic nodes in our cycle-joining tree and in Section 5, we provide an algorithm for determining the children of a given node. In Section 6, we convert our cycle-joining trees to concatenation trees, which leads to a construction requiring $O(1)$-amortized time per bit. In Section 7 we discuss the algorithmic techniques used to extend our constructed orientable sequences to find longer ones for $n \leq 20$. Then in Section 8, we apply similar techniques to find some longest known acyclic orientable sequences for $n \leq 20$. We conclude in Section 9 with a summary of our results and directions for future research. Implementations of our algorithms are available for download at `http://debruijnsequence.org/db/orientable`.

1.1 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 *acyclic orientable sequence*. One does not consider the substrings in the wraparound for this variation of an orientable sequence. Note that one can always construct an acyclic $\mathcal{OS}(n)$ 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 [5] for more on acyclic orientable sequences, which they call *aperiodic* 2*-orientable window sequences*. Alhakim et al. [2] generalize the recursive results of Mitchell and Wild [26] to construct orientable sequences over alphabets of size two or greater; they also generalize the upper bound, by Dai et al. [7], on the length of an orientable sequence. Rampersad and Shallit [27] showed that for every alphabet of size two or greater, 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 [6, 24] 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 $\beta = \mathtt{b}_1\mathtt{b}_2\cdots\mathtt{b}_m \in \mathbf{B}(m)$ 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 $\alpha < \beta$ either if α is a prefix of β or if $\mathtt{a}_i < \mathtt{b}_i$ for the smallest i such that $\mathtt{a}_i \neq \mathtt{b}_i$. We say that α is a *rotation* of β if $m = n$ and there exist strings x and y such that $\alpha = xy$ and $\beta = yx$. 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 $\alpha = \gamma^j$ for some non-empty string γ and some $j > 1$, then α is said to be *periodic*[2]; otherwise, α is said to be *aperiodic* (or *primitive*). Let $\mathrm{ap}(\alpha)$ denote the shortest string γ such that $\alpha = \gamma^t$ for some positive integer t; we say γ is the *aperiodic prefix* of α. Observe that α is aperiodic if and only if $\mathrm{ap}(\alpha) = \alpha$.

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 $\tilde{\alpha}$ denote the necklace 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 $\langle\alpha\rangle$. Note that in general, a bracelet is always a necklace, but a necklace need not be a bracelet. For example, the string 001011 is both a bracelet and a necklace, but the string 001101 is a necklace not a bracelet.

A necklace α is *symmetric* 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 symmetric necklace is necessarily a bracelet. If a necklace or bracelet is not symmetric, it is said to be *asymmetric*. Let $\mathbf{A}(n)$ denote the set of all asymmetric bracelets of length n. Table 1 lists all 60 necklaces of length $n = 9$ partitioned into asymmetric necklace pairs and symmetric necklaces. The asymmetric necklace pairs belong to the same bracelet class, and the first string in each pair is an asymmetric 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 [32]. Asymmetric bracelets have been studied previously in the context of efficiently ranking/unranking bracelets [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]. ◀

---

[2] Periodic strings are are also known as *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, $i \geq 1$, and $\frac{|(\alpha\beta)^i\alpha|}{|\alpha\beta|} \geq 2$. Under this definition, the word `alfalfa` is periodic, but `bonobo` is not.

Asymmetric necklace pairs	Symmetric necklaces		
000001011 , 000001101	000000000	000100011	001110111
000010011 , 000011001	000000001	000101101	001111111
000010111 , 000011101	000000011	000110011	010101011
000100101 , 000101001	000000101	000111111	010101111
000100111 , 000111001	000000111	001001001	010111111
000101011 , 000110101	000001001	001001111	011011011
000101111 , 000111101	000001111	001010011	011011111
000110111 , 000111011	000010001	001010101	011101111
001001011 , 001001101	000010101	001011101	011111111
001010111 , 001110101	000011011	001100111	111111111
001011011 , 001101101	000011111	001101011	
001011111 , 001111101			
001101111 , 001111011			
010110111 , 010111011			

Table 1 A listing of all 60 necklaces in $\mathbf{N}(9)$ partitioned into asymmetric necklace pairs and symmetric necklaces. The first column of the asymmetric necklaces corresponds to the 14 asymmetric bracelets $\mathbf{A}(9)$.

Lemma 2 is considered a folklore result in combinatorics on words; see Theorem 4 in [4] for a variant of the lemma. We provide a short proof for the interested reader.

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

Proof. Suppose α is a symmetric 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 symmetric, 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 symmetric. ◀

▶ **Corollary 3.** *If $\alpha = 0^s\beta$ is a symmetric 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.*

2.1 Bounds on M_n

Dai, Martin, Robshaw, and Wild [7] gave a lower bound L_n and an upper bound U_n on the maximum length M_n of an $\mathcal{OS}(n)$.[3] The lower bound L_n corresponds to the length of a universal cycle that is the result of joining all asymmetric necklaces in a specific way. Their lower bound L_n is the following, where μ is the Möbius function:

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

Their upper bound U_n is the following:[3]

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

[3] These bounds correspond to $\tilde{L}_n$ and $\tilde{U}_n$, respectively, as they appear in [7].

These bounds are calculated in Table 2 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 [26], 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 7) 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 2 Lower bounds R_n, L_n, L_n^* and upper bound U_n for M_n.

2.2 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 *successor rule* for U is a function $f : \mathbf{S} \to \{0, 1\}$ such that $f(\alpha)$ is the bit 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, 15, 17, 18, 30, 31]. 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 [16].

Example 2 Consider disjoint subsets $\mathbf{S}_1 = [011111] \cup [001111]$ and $\mathbf{S}_2 = [010111] \cup [010101]$, where $n = 6$. 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 $\mathbf{S}_1 \cup \mathbf{S}_2$.

A *cycle-joining tree* is a tree with nodes representing disjoint universal cycles; an edge between two nodes implies they each contain one string of a conjugate pair. If $\mathbf{S}$ is the set of all length-n strings belonging to the disjoint cycles of a cycle-joining tree, then the tree defines a universal U for $\mathbf{S}$ along with a corresponding successor rule; 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 necklace cycles represented by $\mathbf{N}(n)$ [14, 28].

- **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 $\mathrm{T}_1, \mathrm{T}_2, \mathrm{T}_3, \mathrm{T}_4$ 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 $\alpha = 0\mathbf{1}1011$ and $\beta = 001011$ in T_2 from Figure 1 are joined by the conjugate pair $(\mathbf{1}10110, 010110)$.

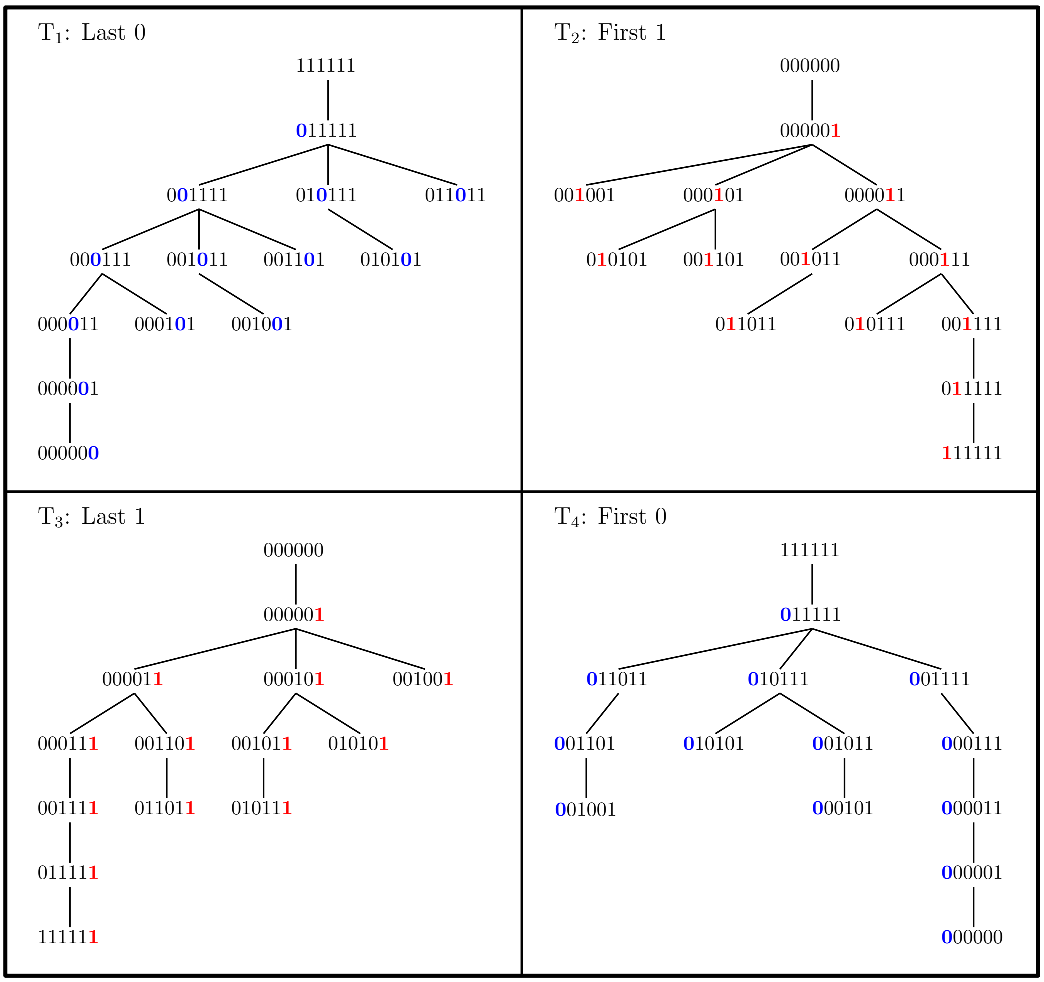


Figure 1 Cycle-joining trees for $\mathbf{B}(6)$ from simple parent rules.

3 An efficient cycle-joining construction of orientable sequences

Consider the set of asymmetric bracelets $\mathbf{A}(n) = \{\alpha_1, \alpha_2, \ldots, \alpha_t\}$. Recall, that each symmetric bracelet is a necklace. Let $\mathbf{S}(n) = [\alpha_1] \cup [\alpha_2] \cup \cdots \cup [\alpha_t]$. From [7], we have $|\mathbf{S}(n)| = L_n$. By 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 $\mathcal{OS}(n)$.

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. Assume $n \geq 8$. Observe that none of the necklaces in $\mathbf{A}(n)$ have 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 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., if the rule applies a combination of the four rules from Section 2.2, it 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

- $\mathrm{first1}(\alpha)$ 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;
- $\mathrm{last1}(\alpha)$ be the necklace of $[\mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_{n-1}\mathbf{0}]$;
- $\mathrm{first0}(\alpha)$ be the necklace of $[\mathbf{1}\mathtt{a}_2\cdots\mathtt{a}_n]$;
- $\mathrm{last0}(\alpha)$ 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 $\mathrm{first1}(\alpha)$ and $\mathrm{last0}(\alpha)$ are necklaces (easily observed by definition) obtained by flipping the i-th and j-th bit in α, respectively; $\mathrm{last1}(\alpha)$ and $\mathrm{first0}(\alpha)$ are the result of flipping a bit and rotating the resulting string to obtain a necklace. The following remark follows from the definition of necklace.

▶ Remark 4. Let $\alpha = \beta 10^t 1$ be a necklace where β is some string, and $t \geq 0$. Then $\mathrm{last1}(\alpha) = 0^{t+1}\beta 1$.

Proposition 5 illustrates that for n sufficiently large, no two of the above four parent rules can be applied in combination to obtain a cycle-joining tree with nodes $\mathbf{A}(n)$.

▶ **Proposition 5.** *Let* p *be a parent rule that applies some combination of* first1*,* last1*,* first0*, and* last0 *to construct a cycle-joining tree with nodes* $\mathbf{A}(n)$*. Then* p *must apply at least three of these rules for all* $n \geq 10$*.*

Proof. Suppose $n \geq 10$. By our earlier observation, any parent rule for a cycle-joining tree with nodes $\mathbf{A}(n)$ must be able to flip a 0 to a 1, and a 1 to a 0. Therefore, p must include one of first0 or last0, and one of first1 and last1.

Suppose p does not apply first0. Then it must apply last0. Consider three asymmetric bracelets in $\mathbf{A}(n)$: $\alpha_1 = 0^{n-4}1011$, $\alpha_2 = 0^{n-5}10111$, and $\alpha_3 = 0^{n-6}110111$. Clearly, $\mathrm{first1}(\alpha_1) = 0^{n-2}11$, $\mathrm{last1}(\alpha_1) = 0^{n-3}101$, and $\mathrm{last0}(\alpha_1) = 0^{n-4}1111$ are symmetric. Thus, α_1 must be the root. Both $\mathrm{first1}(\alpha_2) = 0^{n-3}111$ and $\mathrm{last0}(\alpha_2) = 0^{n-5}11111$ are symmetric, so p must apply last1. Both $\mathrm{last0}(\alpha_3) = 0^{n-6}111111$ and $\mathrm{last1}(\alpha_3) = 0^{n-5}11011$ are symmetric, so p must apply first1.

Suppose p does not apply last0. Then it must apply first0. Let $m = 0$ if n is even, and $m = 1$ otherwise. Let $\ell = (n-6-m)/2$. Note that $\ell \geq 2$ for $n \geq 10$. Consider three asymmetric bracelets in $\mathbf{A}(n)$: $\beta_1 = 00101^{n-7}011$, $\beta_2 = 00101^{n-4}$, and $\beta_3 = 0^{\ell+1}10^{m+1}10^{\ell}11$. Clearly, $\mathrm{last1}(\beta_1) = 000101^{n-7}01$ is symmetric and $\mathrm{first1}(\beta_1) = 00001^{n-7}011$ is not a bracelet. Additionally, $\mathrm{first0}(\beta_1) = 0101^{n-7}0111$ is symmetric when $n = 10$ and is not a bracelet for all $n > 10$. Thus, β_1 must be the root. Both $\mathrm{first1}(\beta_2) = 00001^{n-4}$ and $\mathrm{first0}(\beta_2) = 0101^{n-3}$ are symmetric, so p must apply last1. Now for β_3, we have that $\mathrm{first0}(\beta_3) = 0^{\ell}10^{m+1}10^{\ell}111$ is symmetric. We also have that $\mathrm{last1}(\beta_3) = 0^{\ell+2}10^{m+1}10^{\ell}1$ is symmetric when $n = 11$ and is not a bracelet when $n = 10$ or $n > 11$. Thus, p must apply first1. ◀

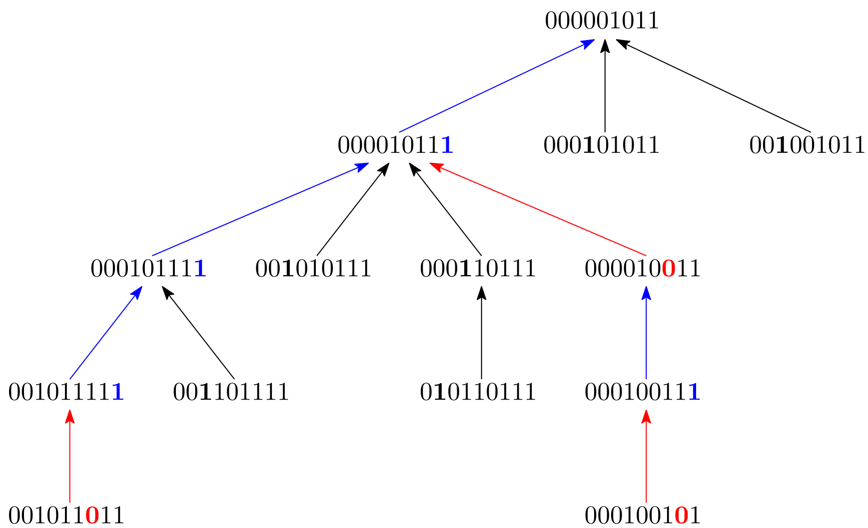
For $n \geq 6$, we choose the lexicographically smallest length-n asymmetric bracelet $r_n = 0^{n-4}1011$ to be the root of our cycle-joining tree.

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

$$\mathrm{par}(\alpha) = \begin{cases} \mathrm{first1}(\alpha) & \text{if } \mathrm{first1}(\alpha) \in \mathbf{A}(n); \\ \mathrm{last1}(\alpha) & \text{if } \mathrm{first1}(\alpha) \notin \mathbf{A}(n) \text{ and } \mathrm{last1}(\alpha) \in \mathbf{A}(n); \\ \mathrm{last0}(\alpha) & \text{otherwise.} \end{cases} \tag{2}$$

▶ **Theorem 6.** *For $n \geq 6$, the parent rule* $\mathrm{par}(\alpha)$ *in* (2) *induces a cycle-joining tree with nodes* $\mathbf{A}(n)$ *rooted at* r_n*. The tree has height less than* $2(n-4)$*.*

Let $\mathbb{T}_n$ denote the cycle-joining tree with nodes $\mathbf{A}(n)$ induced by the parent rule in (2); Figure 2 illustrates $\mathbb{T}_9$. The proof of Theorem 6 relies on the following lemma.



■ **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 $\mathrm{par}(\alpha) = \mathrm{last0}(\alpha)$.

▶ **Lemma 7.** *Let $\alpha \neq r_n$ be an asymmetric bracelet in $\mathbf{A}(n)$. If neither* $\mathrm{first1}(\alpha)$ *nor* $\mathrm{last1}(\alpha)$ *are in* $\mathbf{A}(n)$*, then the last* 0 *in α is at index $n-2$ or $n-1$, and both* $\mathrm{last0}(\alpha)$ *and* $\mathrm{last1}(\mathrm{last0}(\alpha))$ *are in* $\mathbf{A}(n)$*.*

Proof. Since α is an asymmetric bracelet, it must have the form $\alpha = 0^i 1\beta 01^j$ where $i, j \geq 1$ and $\beta 0$ does not contain 0^{i+1} as a substring. Furthermore, $1\beta 01^j < (1\beta 01^j)^R$, which implies $\beta 01^{j-1} < (\beta 01^{j-1})^R$.
Suppose $j > 2$. Since $\mathrm{last1}(\alpha) = 0^{i+1}1\beta 01^{j-1}$ is not an asymmetric bracelet, we have $1\beta 01^{j-1} \geq (1\beta 01^{j-1})^R$. Thus, β begins with 1. Since $\mathrm{first1}(\alpha) = 0^{i+1}\beta 01^j$ is not an asymmetric bracelet, Lemma 2 implies $\beta 01^j \geq (\beta 01^j)^R$, contradicting the earlier observation that $\beta 01^{j-1} < (\beta 01^{j-1})^R$. Thus, the last 0 in α is at index $n-2$ or $n-1$.
Suppose $j = 1$ or $j = 2$. Then the last 0 in α must be at position $n-2$ or $n-1$. Write $\alpha = x0y$ where $y = 1$ or $y = 11$. Since α is a bracelet, it is straightforward to see that $\mathrm{last0}(\alpha) = x1y$ is also a bracelet. If it is symmetric, Lemma 2 implies there exist palindromes β_1 and β_2 such that $\mathrm{last0}(\alpha) = x1y = \beta_1\beta_2$. However, flipping the 1 in $x1y$ that allows us to obtain α implies that α is greater than or equal to the necklace in $[\alpha^R]$, contradicting the assumption that α is an asymmetric bracelet. Thus, $\mathrm{last0}(\alpha)$ is an asymmetric bracelet.

Consider $\mathrm{last1}(\mathrm{last0}(\alpha)) = 0^{i+1}1\beta 1^j$. Let $\beta = \mathtt{b}_1\mathtt{b}_2\cdots\mathtt{b}_m$. Suppose that $m = 0$. Then $\mathrm{last1}(\mathrm{last0}(\alpha)) = 0^{i+1}1^{j+1} \Rightarrow \mathrm{last0}(\alpha) = 0^i 1^{j+2}$. Since $j = 1$ or $j = 2$, we have that $\mathrm{last0}(\alpha) = 0^i 111$ or $\mathrm{last0}(\alpha) = 0^i 1111$. 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 symmetric necklace, a contradiction. Thus, assume $m \geq 1$. Suppose $\beta = 1^m$. Then, α is not an asymmetric bracelet, a contradiction. Suppose $\beta = 0^m$. If $j = 1$, then α is symmetric, a contradiction; if $j = 2$, then $\mathrm{last1}(\mathrm{last0}(\alpha)) = 0^{i+1}10^m 11$ 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^{i+1} as a substring, by Lemma 2, we must show that (i) $\beta 1^{j-1}$ is less than its reversal $1^{j-1}\beta^R$, recalling that (ii) $\beta 01^{j-1}$ is less than its reversal $1^{j-1}0\beta^R$. Let ℓ be the largest index of β such that $\mathtt{b}_\ell = 1$. Then $\mathtt{b}_{\ell+1}\cdots\mathtt{b}_m = 0^{m-\ell}$; note that $\mathtt{b}_{\ell+1}\cdots\mathtt{b}_m$ is the empty string when $\ell = m$. Suppose $j = 1$. From (ii), we have $\mathtt{b}_1 = 0$ and $\mathtt{b}_2\cdots\mathtt{b}_{\ell-1}10^{m-\ell} < 0^{m-\ell}1\mathtt{b}_{\ell-1}\cdots\mathtt{b}_2$. But this implies that $\mathtt{b}_2\cdots\mathtt{b}_{m-\ell+1} = 0^{m-\ell}$. Therefore, we have $\beta = 0^{m-\ell+1}\mathtt{b}_{m-\ell+2}\cdots\mathtt{b}_m < 0^{m-\ell}1\mathtt{b}_{\ell-1}\cdots\mathtt{b}_1 = \beta^R$, hence (i) is satisfied. Suppose $j = 2$. 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 $\mathtt{b}_3 \cdots \mathtt{b}_{m-\ell+2} = 0^{m-\ell}$. Therefore, we have $\beta 1 = 10^{m-\ell+1} \mathtt{b}_{m-\ell+3} \cdots \mathtt{b}_m 1 < 10^{m-\ell} 1 \mathtt{b}_{\ell-1} \cdots \mathtt{b}_1 = 1\beta^R$, hence (i) is satisfied. Thus, $\mathrm{last1}(\mathrm{last0}(\alpha))$ is an asymmetric bracelet. ◀

Proof of Theorem 6. Let α be an asymmetric bracelet in $\mathbf{A}(n) \setminus \{r_n\}$. We can write α as $0^i 1 \beta$ for some string β and $i \geq 1$. We demonstrate that the parent rule par from (2) 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 unique asymmetric bracelet with prefix 0^{n-4}. By Lemma 7, $\mathrm{par}(\alpha) \in \mathbf{A}(n)$. In the first two cases of the parent rule, $\mathrm{par}(\alpha)$ will have prefix 0^{i+1}. If the third case applies, Lemma 7 states that $\mathrm{last1}(\mathrm{last0}(\alpha))$ is an asymmetric 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 asymmetric bracelet has prefix 0^{i+1}. Since either $\mathrm{par}(\alpha)$ or $\mathrm{par}(\mathrm{par}(\alpha))$ has prefix 0^{i+1}, the parent rule induces a path from α to r_n and the height of the resulting tree is at most $2(n-4)-1$.

3.1 A successor rule

Each application of the parent rule $\mathrm{par}(\alpha)$ in (2) corresponds to a conjugate pair. For instance, consider the asymmetric bracelet $\alpha = 00010111\mathbf{1}$. The parent of α is obtained by flipping the last 1 to obtain $00010111\mathbf{0}$ (see Figure 2). The corresponding conjugate pair is $(\mathbf{1}00010111, \mathbf{0}00010111)$. 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 successor rule for an $\mathcal{OS}(n)$:

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

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:

$$0\underline{\mathbf{0}00010111}110010110110010111100110111\underline{\mathbf{1}00010111}00101011100011011$$
$$1010110111000010011100010010100010011000010110010010110001010 11,$$

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 successor rule 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

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

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 8.** *For $n \geq 6$, the function g is a successor rule that generates an $\mathcal{OS}(n)$ with length L_n for the set $\mathbf{S}(n)$ 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 2:

- 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 successor rule on $\mathbf{S}(n)$ that 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 g, the corresponding $\mathcal{OS}(n)$ can be computed in $O(n)$-time per bit using $O(n)$ space. ◀

4 Periodic nodes in $\mathbb{T}_n$

In this section, we present several results on periodic nodes in $\mathbb{T}_n$, assuming $n \geq 6$.

▶ **Lemma 9.** *If a node $\alpha \in \mathbf{A}(n)$ from the cycle-joining tree $\mathbb{T}_n$ is periodic, it has no children.*

Proof. Let α be a non-root node $\mathbf{A}(n)$. We demonstrate that $\mathrm{par}(\alpha)$ is aperiodic, which implies the periodic nodes in $\mathbb{T}_n$ have no children. Let j denote the index of the first 1 in α. Then α has prefix 0^{j-1} and no substring 0^j. Consider the three possibilities for $\mathrm{par}(\alpha)$. Suppose $\mathrm{first1}(\alpha)$ is in $\mathbf{A}(n)$. Then it has prefix 0^j and is aperiodic since there is no substring 0^j not in the initial prefix of 0s. Similarly, if $\mathrm{last1}(\alpha)$ is in $\mathbf{A}(n)$, then it has prefix 0^j and is aperiodic since it also has no substring 0^j not in the initial prefix of 0s. Suppose $\mathrm{last0}(\alpha)$ is in $\mathbf{A}(n)$ and is periodic. Then we can write $\mathrm{last0}(\alpha) = \beta^k$ where $k > 1$ and β is some string that contains a 1. Either β contains a 0, or it does not. If β does not contain a 0, then $\beta = 1^i$ for some $i \geq 1$. But this implies $\alpha = 01^{n-1}$, which is not an asymmetric bracelet, a contradiction. Suppose β contains at least one 0. Write $\alpha = uv = yx$ where u, v, x, y are nonempty strings such that $|u| = |x| = |\beta|$. Since β contains at least one 0, the last 0 in α must occur in x and we must have $u = \beta$. Thus, one can obtain x from β by flipping a single 1 to a 0, which implies $x < \beta$. So we have $xy < \beta v = uv = yx = \alpha$, which contradicts α being a bracelet. Therefore $\mathrm{par}(\alpha)$ is aperiodic. ◀

▶ **Lemma 10.** *The number of periodic nodes in $\mathbb{T}_n$ is less than or equal to the number of aperiodic nodes in $\mathbb{T}_n$.*

Proof. It suffices to show the existence of a 1-1 mapping f from the periodic strings in $\mathbf{A}(n)$ to the aperiodic strings in $\mathbf{A}(n)$. Let α be periodic and in $\mathbf{A}(n)$. Then $\alpha = \beta^i$ for some aperiodic asymmetric bracelet β where $i > 1$. Let $p = |\beta|$. Define $f(\alpha) = 0^{p-1}1\beta^{i-1}$. Clearly f is 1-1; if $f(\alpha) = f(\alpha')$ for some periodic $\alpha' \in \mathbf{A}(n)$, then $f(\alpha)$ and $f(\alpha')$ share the prefix $0^{p-1}1$, which implies $\alpha = \alpha'$. Now we prove that $f(\alpha)$ is aperiodic and is in $\mathbf{A}(n)$. We must have $\beta > 0^{p-1}1$, for otherwise α would be a symmetric bracelet. Thus, $f(\alpha)$ is an aperiodic necklace, but is not necessarily in $\mathbf{A}(n)$. Write $\beta = 0^k1\gamma$ where $k \geq 1$ and γ is a non-empty string. Since β is an aperiodic bracelet, it is an aperiodic necklace. Therefore, any nonempty proper prefix of β cannot also be a suffix of β [23, Proposition 5.1.2], and β has no substring 0^{k+1}. So β^R must begin with a string larger than 10^k1, and thus $1\beta = 10^k1\gamma < \beta^R1$. It follows that $f(\alpha) \in \mathbf{A}(n)$. ◀

From equation (1), we immediately have the following corollary.

▶ **Corollary 11.** $n|\mathbf{A}(n)| \leq 2L_n$.

5 Computing the children of a node in $\mathbb{T}_n$

In this section, we present an optimized way to determine the children of a node $\beta = \mathtt{b}_1\mathtt{b}_2\cdots\mathtt{b}_n$ in $\mathbb{T}_n$. We use this optimization in Section 6 to generate orientable sequences in $O(1)$-amortized time per bit.

Let s and t be integers such that β has prefix 0^s1 and suffix 10^t1. Let s' denote the largest integer such that $0^{s'}$ is a substring of $\mathtt{b}_{s+1}\cdots\mathtt{b}_n$. Let β_k denote $\mathtt{b}_1\cdots\mathtt{b}_{k-1}\overline{\mathtt{b}}_k\mathtt{b}_{k+1}\cdots\mathtt{b}_n$; it differs from β only at index k. Recall that $\tilde{\beta_k}$ is the necklace in $[\beta_k]$. Let $\mathrm{MAX}(x,y)$ denote the maximum of the integers x and y. Our goal is to determine the indices k such that $\tilde{\beta_k}$ is in $\mathbf{A}(n)$ and $\mathrm{par}(\tilde{\beta_k}) = \beta$. Consider the three cases of the parent rule par:

- Suppose $\mathrm{par}(\tilde{\beta_k}) = \mathrm{first1}(\tilde{\beta_k}) = \beta$. Since β and β_k differ only at index k, it must be that k is the index of the first 1 in β_k. Thus $\tilde{\beta_k} = \beta_k$ has prefix $0^{k-1}1$ and $k \leq s$. Since β_k is a necklace, $k > \mathrm{MAX}(\lfloor s/2 \rfloor, s')$. Suppose $\mathrm{MAX}(\lfloor s/2 \rfloor, s')$ $+1 < k < s$ and β_k is not in $\mathbf{A}(n)$. Note that β_k is a necklace since it has a unique substring 0^{k-1} as a prefix. Thus, it must be that $1\mathtt{b}_{k+1}\cdots\mathtt{b}_n \geq (1\mathtt{b}_{k+1}\cdots\mathtt{b}_n)^R$. Since $k+1 \leq s$, this implies that $1\mathtt{b}_{k+2}\cdots\mathtt{b}_n \geq (1\mathtt{b}_{k+2}\cdots\mathtt{b}_n)^R$ and hence β_{k+1} is also not in $\mathbf{A}(n)$. Thus, starting from index $k = \mathrm{MAX}(\lfloor s/2 \rfloor, s')$ $+1$ (which may or may not lead to a child), and incrementing up to s, we can stop testing once an index $k > \mathrm{MAX}(\lfloor s/2 \rfloor, s')$ $+1$ does not lead to a child.
- Suppose $\mathrm{par}(\tilde{\beta_k}) = \mathrm{last1}(\tilde{\beta_k}) = \beta$. It follows from Remark 4 that $\tilde{\beta_k} = \mathtt{b}_{k+1}\cdots\mathtt{b}_n0^{k-1}1$. Since $\tilde{\beta_k}$ is a necklace, it must be that $k \leq \lceil s/2 \rceil$. If k is the smallest index in $1, 2, \ldots, \lceil s/2 \rceil - 1$ such that $\mathtt{b}_{k+1}\cdots\mathtt{b}_n0^{k-1}1$ is not in $\mathbf{A}(n)$, then by applying the definition of an asymmetric bracelet, it is straightforward to verify that $\mathtt{b}_{k+2}\cdots\mathtt{b}_n0^k1$ is also not in $\mathbf{A}(n)$. Thus, starting from index $k = 1$ and incrementing, we can stop testing indices k for this case once $\mathtt{b}_{k+1}\cdots\mathtt{b}_n0^{k-1}1$ is not in $\mathbf{A}(n)$.
- Suppose $\mathrm{par}(\tilde{\beta_k}) = \mathrm{last0}(\tilde{\beta_k}) = \beta$. Then it must be that $k = n-1$ or $k = n-2$ from Lemma 7.

Based on this analysis, the function FINDCHILDREN(β) defined in Algorithm 1 will return $\mathtt{c}_1\mathtt{c}_2\cdots\mathtt{c}_n$ such that $\mathtt{c}_k = 1$ if and only if $\tilde{\beta_k}$ is a child of β.

Algorithm 1 Determine the children of a node $\beta = \mathtt{b}_1\mathtt{b}_2\cdots\mathtt{b}_n$ in $\mathbb{T}_n$, returning $\mathtt{c}_1\mathtt{c}_2\cdots\mathtt{c}_n$ such that $\mathtt{c}_k = 1$ if and only if $\tilde{\beta_k}$ is a child of β.

function FINDCHILDREN(β)
  $\mathtt{c}_1\mathtt{c}_2\cdots\mathtt{c}_n \leftarrow 0^n$
  $s \leftarrow$ integer such that 0^s1 is a prefix of β
  $s' \leftarrow$ largest integer such that $0^{s'}$ is a substring of $\mathtt{b}_{s+1}\cdots\mathtt{b}_n$

  ▷ FIRST 1
  for k **from** $\mathrm{MAX}(\lfloor s/2 \rfloor, s') + 1$ **to** s **do**
    if $0^{k-1}1\mathtt{b}_{k+1}\cdots\mathtt{b}_n \in \mathbf{A}(n)$ **then** $\mathtt{c}_k \leftarrow 1$
    else if $k > \mathrm{MAX}(\lfloor s/2 \rfloor, s') + 1$ **then break**

  ▷ LAST 1
  for k **from** 1 **to** $\lceil s/2 \rceil$ **do**
    if $\mathtt{b}_{k+1}\cdots\mathtt{b}_n0^{k-1}1 \in \mathbf{A}(n)$ **then**
      if $\beta = \mathrm{par}(\mathtt{b}_{k+1}\cdots\mathtt{b}_n0^{k-1}1)$ **then** $\mathtt{c}_k \leftarrow 1$
    else break

  ▷ LAST 0
  if $\mathtt{b}_{n-1} = 1$ **and** $\mathtt{b}_1\cdots\mathtt{b}_{n-2}01 \in \mathbf{A}(n)$ **and** $\beta = \mathrm{par}(\mathtt{b}_1\cdots\mathtt{b}_{n-2}01)$ **then** $\mathtt{c}_{n-1} \leftarrow 1$
  if $\mathtt{b}_{n-1} = \mathtt{b}_{n-2} = 1$ **and** $\mathtt{b}_1\cdots\mathtt{b}_{n-3}011 \in \mathbf{A}(n)$ **and** $\beta = \mathrm{par}(\mathtt{b}_1\cdots\mathtt{b}_{n-3}011)$ **then** $\mathtt{c}_{n-2} \leftarrow 1$

  return $\mathtt{c}_1\cdots\mathtt{c}_n$

▶ **Lemma 12.** *The time required by calls to* FINDCHILDREN(β) *summed over all* $\beta \in \mathbf{A}(n)$ *is* $O(L_n)$.

Proof. Each operation in FINDCHILDREN requires at most $O(n)$ work, including membership testing to $\mathbf{A}(n)$, and the parent function. Consider each of the two **for** loops. In the first **for** loop on line 6, there are at most two membership tests to $\mathbf{A}(n)$ that do not detect children; for all other tests the $O(n)$ work can be assigned to the corresponding child node in $\mathbf{A}(n)$. For the second **for** loop starting at line 10, only one membership test to $\mathbf{A}(n)$ will fail; however, there may be multiple parent tests on line 12 that do not lead to a child. In these cases, $\mathrm{last1}(\mathtt{b}_{j+1}\cdots\mathtt{b}_n0^{k-1}1) = \beta$, but $\mathrm{par}(\mathtt{b}_{j+1}\cdots\mathtt{b}_n0^{k-1}1) = \mathrm{first1}(\mathtt{b}_{j+1}\cdots\mathtt{b}_n0^{k-1}1)$. The $O(n)$ work from each of these parent tests can be assigned uniquely to the node corresponding to the asymmetric bracelet being tested $\mathtt{b}_{j+1}\cdots\mathtt{b}_n0^{k-1}1$; each node in $\mathbf{A}(n)$ can receive at most one such assignment because of the fact that $\mathrm{last1}(\mathtt{b}_{j+1}\cdots\mathtt{b}_n0^{k-1}1) = \beta$. Since a linear amount of work can be assigned to each $\beta \in \mathbf{A}(n)$, the time required by calls to FINDCHILDREN(β) summed over all $\beta \in \mathbf{A}(n)$ is $O(n|\mathbf{A}(n)|)$. Thus, by Corollary 11 we have our result. ◀

6 Concatenation trees and RCL order

In this section, we apply the recent theory of concatenation trees [28] to produce the orientable sequences constructed in the previous section in $O(1)$-amortized time per bit using $O(n^2)$ space.

Since Lemma 9 demonstrates that every periodic node in $\mathbb{T}_n$ is a leaf, we can simplify the upcoming definition of a concatenation tree relative to the original definition in [28]. A *bifurcated ordered tree* (BOT) is a rooted tree where each node contains two *ordered* lists of children, the left-children and right-children, respectively. The *concatenation tree* $\mathcal{T}_n$ is derived from $\mathbb{T}_n$ by converting it into a BOT; the label, which is also the representative, of each node may change, and the children are partitioned into ordered left-children and right-children. The definitions of the node labels are defined recursively along with a corresponding *change index*, which is the unique index where a node's label differs from its parent. If a node has change index c, its left-children are the children with change index less than c, and the right-children are the children with change index greater than c; in both cases the children are ordered from smallest to largest based on their change index. The root is labeled r_n and it is assigned change index n. As an example, the concatenation tree $\mathcal{T}_9$ in Figure 3 is obtained from the cycle-joining tree $\mathbb{T}_9$ illustrated in Figure 2.

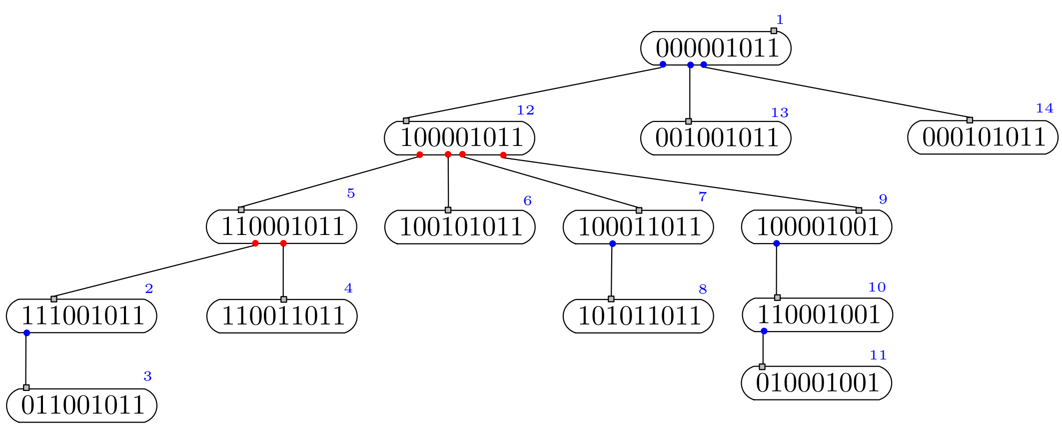


Figure 3 The concatenation tree $\mathcal{T}_9$ derived from the cycle-joining tree $\mathbb{T}_9$ shown in Figure 2. The small grey box on the top edge of each node indicates the change index; the left-children descend from blue dots • and the right-children descend from red dots • . The small numbers above each node indicate the order the nodes are visited in an RCL traversal.

A *right-current-left (RCL) traversal* of $\mathcal{T}_n$ starts at the root and recursively visits the right-children from first to last, followed by the current node, followed by recursively visiting the left-children from first to last. Let $\mathrm{RCL}(\mathcal{T}_n)$ denote the sequence generated by traversing $\mathcal{T}_n$ in RCL order, outputting the aperiodic prefix $\mathrm{ap}(\alpha)$ as each node α is visited. The order the nodes of $\mathcal{T}_9$ are visited by an RCL traversal is illustrated in Figure 3; the corresponding sequence $\mathrm{RCL}(\mathcal{T}_9)$ is an $\mathcal{OS}(9)$ of length $L_9 = 126$:

000001011 111001011 011001011 110011011 110001011 100101011 100011011
101011011 100001001 110001001 010001001 100001011 001001011 000101011.

In this example, each node α is aperiodic and hence $\text{ap}(\alpha) = \alpha$, but this is not always the case.

The following theorem follows directly from the main result in [28], recalling the successor-rule g defined in Section 3.1.

▶ **Theorem 13.** *For $n \geq 6$, the sequence $\text{RCL}(\mathcal{T}_n)$ is an $\mathcal{OS}(n)$ of length L_n that has successor-rule g.*

To avoid the exponential space required to store a concatenation tree, we demonstrate how to efficiently determine the children of a given node $\alpha = \mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n$ in $\mathcal{T}_n$. In particular, given an index k, we want to determine whether or not $\alpha_k = \mathtt{a}_1\cdots\mathtt{a}_{k-1}\overline{\mathtt{a}}_k\mathtt{a}_{k+1}\cdots\mathtt{a}_n$ is a child of α. From Lemma 9, if α is periodic, it has no children. Otherwise, $\tilde{\alpha} = \mathtt{a}_s\cdots\mathtt{a}_n\mathtt{a}_1\cdots\mathtt{a}_{s-1}$ is a node in $\mathbb{T}_n$ for some $1 \leq s \leq n$; it is the necklace of $[\alpha]$. Thus, if $\mathtt{c}_1\mathtt{c}_2\cdots\mathtt{c}_n =$ FINDCHILDREN($\tilde{\alpha}$) (see Section 5), $\mathtt{d}_1\mathtt{d}_2\cdots\mathtt{d}_n = \mathtt{c}_s\cdots\mathtt{c}_n\mathtt{c}_1\cdots\mathtt{c}_{s-1}$ is a sequence such that $\mathtt{d}_k = 1$ if and only if α_k is a child of α in $\mathcal{T}_n$. The procedure FASTRCL(α, c), shown in Algorithm 2, applies this observation to generate $\text{RCL}(\mathcal{T}_n)$ when initialized with $\alpha = r_n$ and $c = n$.

Algorithm 2 RCL traversal of $\mathcal{T}_n$ with the initial call of FASTRCL(r_n, n). The current node $\alpha = \mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n$ has change index c.

procedure FASTRCL(α, c)
  $p \leftarrow$ period of α
  if $p < n$ **then** PRINT($\mathtt{a}_1\cdots\mathtt{a}_p$) ▷ Visit periodic node (it has no children)
  else
    $s \leftarrow$ unique index such that $\mathtt{a}_s\cdots\mathtt{a}_n\mathtt{a}_1\cdots\mathtt{a}_{s-1}$ is a necklace
    $\mathtt{c}_1\mathtt{c}_2\cdots\mathtt{c}_n \leftarrow$FINDCHILDREN($\mathtt{a}_s\cdots\mathtt{a}_n\mathtt{a}_1\cdots\mathtt{a}_{s-1}$) ▷ Determine the children indices relative to $\mathbb{T}_n$
    $\mathtt{d}_1\mathtt{d}_2\cdots\mathtt{d}_n \leftarrow \mathtt{c}_s\cdots\mathtt{c}_n\mathtt{c}_1\cdots\mathtt{c}_{s-1}$ ▷ Make child indices relative to α in $\mathcal{T}_n$

    ▷ RCL traversal
    for $i \leftarrow c+1$ **to** n **do**
      if $\mathtt{d}_i = 1$ **then** FASTRCL($\mathtt{a}_1\cdots\mathtt{a}_{i-1}\overline{\mathtt{a}}_i\mathtt{a}_{i+1}\cdots\mathtt{a}_n$, i) ▷ Visit Right-children
    PRINT($\mathtt{a}_1\cdots\mathtt{a}_n$) ▷ Visit Current node
    for $i \leftarrow 1$ **to** $c-1$ **do**
      if $\mathtt{d}_i = 1$ **then** FASTRCL($\mathtt{a}_1\cdots\mathtt{a}_{i-1}\overline{\mathtt{a}}_i\mathtt{a}_{i+1}\cdots\mathtt{a}_n$, i) ▷ Visit Left-children

▶ **Theorem 14.** *For $n \geq 6$,* FASTRCL*(r_n, n) generates $\text{RCL}(\mathcal{T}_n)$ in $O(1)$-amortized time per bit using $O(n^2)$ space.*

Proof. Each recursive call requires $O(n)$ space and from Theorem 6, the tree has height less than $2(n-4)$. Thus, the space required by the algorithm is $O(n^2)$. By Lemma 12, the work required by all calls to FINDCHILDREN is $O(L_n)$. Ignoring these calls, there is a $O(n)$ work done at each recursive call to RCL(α, c); determining the necklace and period of a string can be computed in $O(n)$ time [3]. Since there are $|\mathbf{A}(n)|$ nodes in $\mathcal{T}_n$, the total work is $O(L_n)$ by applying Corollary 11. Thus, the algorithm FASTRCL(r_n, n), which outputs L_n bits, runs in $O(1)$-amortized time per bit. ◀

7 Extending orientable sequences

The values from the column labeled L^*_n in Table 2 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 [26] 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.

8 Acyclic orientable sequences

Let $\mathcal{AOS}(n)$ denote an acyclic orientable sequence of order n. If $\mathtt{o}_1\mathtt{o}_2\cdots\mathtt{o}_m$ is an $\mathcal{OS}(n)$, then it follows from our definitions that $\mathtt{o}_1\cdots\mathtt{o}_m\mathtt{o}_1\cdots\mathtt{o}_{n-1}$ is an $\mathcal{AOS}(n)$. As noted in [5], none of the $2^{\lfloor(n+1)/2\rfloor}$ binary palindromes of length n can appear as a substring in any $\mathcal{AOS}(n)$. Thus, a straightforward upper bound on the length of any $\mathcal{AOS}(n)$ is

$$\hat{U}_n = \frac{1}{2}(2^n - 2^{\lfloor(n+1)/2\rfloor}) + (n-1) \quad [5].$$

By applying our cycle-joining based construction, we can efficiently construct an $\mathcal{AOS}(n)$ of length $L_n + (n-1)$. Previously, the only construction of $\mathcal{AOS}(n)$s recursively applied Lempel's lift [26], requiring exponential space. Starting with an $\mathcal{OS}(n)$ found by extending a constructed sequence of length L_n (see Section 7), we apply a computer search to extend the $\mathcal{OS}(n)$ to an $\mathcal{AOS}(n)$ by considering each $\mathtt{o}_i\cdots\mathtt{o}_m\mathtt{o}_1\cdots\mathtt{o}_{i-1}$ and attempting to extend in each direction. This approach produced the longest known $\mathcal{AOS}(n)$s for $n = 12, 13, 14$, improving on the lengths discovered by Burns and Mitchell [5] from applying a computer search. The original data from [5] was for $n \leq 16$; we extend the list of longest known $\mathcal{AOS}(n)$s up to $n = 20$.[4] These results are summarized in Table 3.

	Constructions		Computer Search		
n	Recursion [MW21]	$L_n + (n-1)$	Extended from $\mathcal{OS}(n)$	[BM93]	$\hat{U}_n$
6	**26**	11	**26**	**26**	33
7	**48**	20	**48**	**48**	62
8	92	55	**108**	**108**	127
9	178	134	193	**210**	248
10	350	309	435	**440**	505
11	692	692	868	**872**	1002
12	1376	1541	**1874**	1860	2027
13	2742	3288	**3732**	3710	4044
14	5474	6929	**7724**	7400	8141
15	10936	14534	15432	15467	16270
16	21860	29823	31560	31766	32655
17	43706	61216	**63219**	–	65296
18	87398	124461	**128680**	–	130833
19	174780	252842	**257340**	–	261650
20	349544	509239	**519212**	–	523795

Table 3 The lengths of the longest known $\mathcal{AOS}(n)$s found via construction and computer search for $n = 6, 7, \ldots, 20$.

[4] The resulting $\mathcal{AOS}(n)$s generated up to $n = 20$ are available for download at http://debruijnsequence.org/db/orientable.

9 Conclusion

In this paper we presented two algorithms to construct orientable sequences with asymptotically optimal length. The first algorithm is a successor rule that outputs each bit in $O(n)$ time using $O(n)$ space; the second algorithm generates the same sequences in $O(1)$-amortized time per bit using $O(n^2)$ space by applying a recent concatenation-tree framework [28]. This answers a long-standing open question by Dai, Martin, Robshaw, and Wild [7]. We conclude with the following directions for future research:

1. Can the lower bound of L_n for $\mathcal{OS}(n)$s be improved?
2. Can small strings be inserted systematically into our constructed $\mathcal{OS}(n)$s to obtain longer orientable sequences?
3. 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 [25, 33]. 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 [29] 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 Brlek, S., Hamel, S., Nivat, M., and Reutenauer, C. On the palindromic complexity of infinite words. *Internat. J. Found. Comp. Sci. 15*, 02 (2004), 293–306.

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

6 Currie, J., and Lafrance, P. Avoidability index for binary patterns with reversal. *Electronic J. Combinatorics 23*, (1) P1.36 (2016), 1–14.

7 Dai, Z. D., Martin, K. M., Robshaw, M. J. B., and Wild, P. R. Orientable sequences. In *Cryptography and Coding III (M.J.Ganley, ed.)* (1993), Oxford University Press, pp. 97–115.

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 Gabrić, D., and Sawada, J. Efficient construction of long orientable sequences. In *35th Annual Symposium on Combinatorial Pattern Matching (CPM 2024)* (2024), pp. 1–12. to appear.

14 Gabrić, 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.

15 Gabrić, 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.

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

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

18 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.

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 Lothaire, M. *Combinatorics on Words*, 2 ed. Cambridge Mathematical Library. Cambridge University Press, 1997.

24 Mercaş, R. On the aperiodic avoidability of binary patterns with variables and reversals. *Theoret. Comput. Sci. 682* (2017), 180–189.

25 Mitchell, C., Etzion, T., and Paterson, K. A method for constructing decodable de Bruijn sequences. *IEEE Trans. Inform. Theory 42*, 5 (1996), 1472–1478.

26 Mitchell, C. J., and Wild, P. R. Constructing orientable sequences. *IEEE Trans. Inform. Theory 68*, 7 (2022), 4782–4789.

27 Rampersad, N., and Shallit, J. O. Words that avoid reversed subwords. *J. Combin. Math. Combin. Comput. 54* (2005), 157–164.

28 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.

29 Sawada, J., and Williams, A. Practical algorithms to rank necklaces, Lyndon words, and de Bruijn sequences. *J. Disc. Alg. 43* (2017), 95–110.

30 Sawada, J., and Williams, A. Constructing the first (and coolest) fixed-content universal cycle. *Algorithmica 85*, 6 (2023), 1754–1785.

31 Sawada, J., and Wong, D. Efficient universal cycle constructions for weak orders. *Discrete Math. 343*, 10 (2020), 112022.

32 Sloane et al., N. J. A. OEIS Foundation Inc. (2024), The On-Line Encyclopedia of Integer Sequences, `https://oeis.org`.

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