

Decoding universal cycles for t -subsets and t -multisets by decoding bounded-weight de Bruijn sequences

Daniel Gabrić

University of Guelph, Canada

Wazed Imam

University of Guelph, Canada

Lukas Janik-Jones

University of Guelph, Canada

Joe Sawada

University of Guelph, Canada

Abstract

A universal cycle for a set S of combinatorial objects is a cyclic sequence of length $|S|$ that contains a *representative* of each element in S exactly once as a substring. Despite the many universal cycle constructions known in the literature for various sets including k -ary strings of length n , permutations of order n , t -subsets of an n -set, and t -multisets of an n -set, remarkably few have efficient decoding (ranking/unranking) algorithms. In this paper we develop the first polynomial time/space decoding algorithms for bounded-weight de Bruijn sequences for strings of length n over an alphabet of size k . The results are then applied to efficiently decode universal cycles for t -subsets and t -multisets.

1 Introduction

A *universal cycle* for a set S of combinatorial objects is a cyclic sequence of length $|S|$ that contains a *representation* of each element in S exactly once as a substring. Unless otherwise stated, assume that the representative of each element $s \in S$ is simply s itself. In cases where s is not a string, then its representative will be clearly articulated. The following “fundamental questions” can be asked of any set S [CDG92]:

- (i) Do universal cycles *exist* for S ?
- (ii) How many universal cycles for S are there?
- (iii) Can a universal cycle for S be (efficiently) constructed?
- (iv) Does a universal cycle for S exist with properties that make it efficiently *decodable*? In other words, are there efficient ranking and unranking algorithms for a specific universal cycle of S ?

There are numerous results pertaining to the first three questions for a wide variety of interesting sets S ; see for instance [Fre82, CDG92, JSH09, BG11, Etz24, dbs25]. The decoding question, though, has proven to be more elusive.

Let $S_k(n)$ denote the set of all length- n strings over $\{1, 2, \dots, k\}$. A universal cycle for $S_k(n)$ is known as a *de Bruijn sequence*. The *weight* of a string is the sum of its symbols. Let $S_k(n, w^\uparrow)$ denote the subset of $S_k(n)$ containing the strings with weight *at least* w . Let $S_k(n, w_\downarrow)$ denote the subset of $S_k(n)$ containing the strings with weight *at most* w . Universal cycles for $S_k(n, w^\uparrow)$ and $S_k(n, w_\downarrow)$ are known as a *bounded-weight de Bruijn sequences*. In this paper we present efficient ranking and unranking algorithms for a bounded-weight de Bruijn sequence. The algorithms are applied to efficiently decode universal cycles for t -subsets and t -multisets. They are the first known efficient algorithms to decode specific universal cycles for these objects.

Decoding universal cycles. Let $\mathcal{U} = u_1u_2 \cdots u_m$ be a universal cycle for a set $\mathbf{S}$ of length- n strings. The *rank* of a string $s \in \mathbf{S}$ is the starting index where s is found in $\mathcal{U}$. If s is a prefix of $\mathcal{U}$ then it has rank 1; if s starts at the last symbol of $\mathcal{U}$ and wraps around, then it has rank $|\mathbf{S}|$. Given a rank r , the *unranking* process determines the string $s \in \mathbf{S}$ starting at index r in $\mathcal{U}$. Together, ranking and unranking algorithms allow for a universal cycle to be *decoded*. Any universal cycle can be decoded by traversing the sequence until a given string, or rank, is discovered; however, this requires $O(|\mathbf{S}|)$ time. Similarly, a look-up table can be applied to store the position of a given string, but this requires $\Theta(|\mathbf{S}|)$ space. We say a decoding algorithm is *efficient* if the ranking and unranking algorithms run in polynomial time and require polynomial space, with respect to the size of the alphabet and the length of each string s . The efficient decoding of a universal cycle is necessary when it is applied to robotic position sensing (vision) applications [BM93, DG11].

There are numerous de Bruijn sequence constructions that apply a variety of different algorithmic approaches; see, for instance [Fre82, Etz24, dbs25]. However, somewhat surprisingly, only the lexicographically smallest de Bruijn sequence for $\mathbf{S}_k(n)$ is known to be efficiently decodable for all n and k [KRR16, SW17]. We review the decoding approach for this de Bruijn sequence in Section 3. A recursively constructed de Bruijn sequence can be efficiently decoded in the binary case, but not for all n and k [MEP96, Tul01]. See also [Knu11, p.317, ex.97-99]. The only other known efficient decoding algorithms are for universal cycles of permutations using a shorthand representation [RW10, HRW12]. This highlights the difficulty in discovering such decoding algorithms.

t -subsets and t -multisets. Let $\text{Subset}(n, t)$ denote the set of all t -subsets of $\{1, 2, \dots, n\}$. Universal cycles for $\text{Subset}(n, t)$ have been considered using a standard string representation for each subset, where, for instance, either 12 or 21 can be used to represent the subset $\{1, 2\}$ [CDG92, Jac93, Hur94, JSH09, Rud13]. Using this representation, universal cycles for t -subsets exist only if n divides $\binom{n}{t}$. A t -subset can also be represented by a length- n binary string with t ones. A shorthand representation excludes the final redundant bit. Applying this shorthand representation, universal cycles for t -subsets can be efficiently constructed for all $1 \leq t \leq n$ [RSW12]. No efficient decoding algorithms are known for these universal cycles. In [CGGP21], a labelled graph is applied to represent a t -subset. With this representation they demonstrate the existence of a universal cycle for t -subsets, however, no efficient construction is provided. A difference representation was recently described in [CJJS25]. Given a subset $\{s_1, s_2, \dots, s_t\}$ with elements listed in increasing order, its *difference representative* is the length- t string where the first symbol is s_1 and the j -th symbol is $s_j - s_{j-1}$ for $1 < j \leq t$. For example, the difference representatives for $\text{Subset}(5, 3) =$

$$\{\{1, 2, 3\}, \{1, 2, 4\}, \{1, 2, 5\}, \{1, 3, 4\}, \{1, 3, 5\}, \{1, 4, 5\}, \{2, 3, 4\}, \{2, 3, 5\}, \{2, 4, 5\}, \{3, 4, 5\}\}$$

are given by 111, 112, 113, 121, 122, 131, 211, 212, 221, and 311, respectively. In general, the difference representatives for $\text{Subset}(n, t)$ correspond to the strings in $\mathbf{S}_{n-t+1}(t, n_{\downarrow})$.

Let $\text{Multiset}(n, t)$ denote the set of all t -multisets of $\{0, 1, \dots, n-1\}$. Universal cycles for $\text{Multiset}(n, t)$ have been considered using a standard string representation for each subset, where, for instance, either 112, 121, or 211 can be used to represent the multiset $\{1, 1, 2\}$ [Jac93, HJZ09]. Using this representation, universal cycles for t -multisets exist only if n divides $\binom{n+t-1}{t}$. Like with t -subsets, we can also apply a difference representation for t -multisets [CJJS25]. Given a multiset $\{m_1, m_2, \dots, m_t\}$ with elements listed in non-decreasing order, its *difference representative* is the length t string where the first symbol is m_1 and the j -th symbol is $m_j - m_{j-1}$ for $1 < j \leq t$. For example, the difference representatives for $\text{Subset}(3, 3) =$

$$\{\{0, 0, 0\}, \{0, 0, 1\}, \{0, 0, 2\}, \{0, 1, 1\}, \{0, 1, 2\}, \{0, 2, 2\}, \{1, 1, 1\}, \{1, 1, 2\}, \{1, 2, 2\}, \{2, 2, 2\}\}$$

are given by 000, 001, 002, 010, 011, 020, 100, 101, 110, and 200, respectively. By replacing each symbol x with $x + 1$ in the difference representatives for $\text{Multiset}(n, t)$, the resulting set of strings is $\mathbf{S}_n(t, (n+t-1)_{\downarrow})$.

Main results. The main results of this paper are as follows:

1. We demonstrate that the lexicographically smallest universal cycle for $S_k(n, w^\uparrow)$ (a bounded-weight de Bruijn sequence) is efficiently decodable.
2. We demonstrate a universal cycle for t -subsets that is efficiently decodable.
3. We demonstrate a universal cycle for t -multisets that is efficiently decodable.

As we mention later in the paper, a decodable universal cycle for $S_k(n, w^\uparrow)$ can be easily applied to decode a universal cycle for the strings in $S_k(n, w_\downarrow)$. The analysis of all algorithms described in this paper assume the unit-cost RAM model of computation. A C implementation of our decoding algorithms is available at https://debruijnsequence.org/db/subset_decode.

Outline. The remainder of the paper is organized as follows. In Section 2 we present background definitions and notation. In Section 3 we review the efficient decoding approach for the lexicographically smallest de Bruijn sequence. In Section 4 we generalize these results by considering a lower bound on the weight of the strings; in Section 5 we consider an upper bound on the weight. In Section 6, we apply these results to demonstrate efficiently decodable universal cycles for t -subsets and t -multisets. In Section 7, we prove the technical results.

2 Preliminaries

Recall that $S_k(n)$ denotes the set of all length- n strings over $\{1, 2, \dots, k\}$. Consider two strings s and t . Let $s \cdot t$ denote the concatenation of s and t , and let t^j denote j copies of t concatenated together. A *necklace* is the lexicographically smallest string in an equivalence class of strings under rotation. Let $\text{neck}(s)$ denote the lexicographically smallest rotation of s , i.e., its necklace. Let $\text{ap}(s)$ denote the *aperiodic prefix* of s , i.e., the shortest string t such that $s = t^j$ for some $j \geq 1$. For example, $\text{ap}(1111) = 1$. A string s is said to be *aperiodic* (primitive) if $\text{ap}(s) = s$; otherwise s is *periodic* (a power).

Let $\mathbf{N}_k(n) = (\sigma_1, \sigma_2, \dots, \sigma_t)$ denote the tuple consisting of all necklaces in $S_k(n)$ listed in lexicographic order. For example,

$$\mathbf{N}_2(4) = (1111, 1112, 1122, 1212, 1222, 2222).$$

Amazingly, by concatenating together the aperiodic prefixes of the necklaces in $\mathbf{N}_k(n)$ [FM78] we obtain the lexicographically smallest de Bruijn sequence for $S_k(n)$ [FM78, Fre70]. Let

$$\mathcal{G}_k(n) = \text{ap}(\sigma_1) \text{ap}(\sigma_2) \cdots \text{ap}(\sigma_t),$$

denote this lexicographically smallest de Bruijn sequence for $S_k(n)$. For example,

$$\mathcal{G}_2(4) = 1 \cdot 1112 \cdot 1122 \cdot 12 \cdot 1222 \cdot 2.$$

The sequence $\mathcal{G}_k(n)$ is also known as the *Granddaddy*¹ de Bruijn sequence and the *Ford* sequence².

A note on notation. In this paper we use bold lower case letters such as $\mathbf{r}, \mathbf{s}, \mathbf{t}$ to denote arbitrary strings, while we reserve Greek letters such as $\alpha, \beta, \sigma, \gamma$ to denote necklaces.

¹ This name was originally given to Martin's prefer-largest greedy construction [Mar34] by Knuth [Knu11, p.317]. It is equivalent to $\mathcal{G}_k(n)$ by a simple mapping of the alphabet symbols.

² Based on a 1957 technical report by Lester Ford who independently discovered the sequence via an equivalent greedy construction.

2.1 Bounding the weight

The *weight* of a string s , denoted by $\text{weight}(s)$, is the sum of its symbols. Recall that $\mathbf{S}_k(n, w^\uparrow)$ denotes the subset of $\mathbf{S}_k(n)$ containing strings with weight *at least* w . Let $\mathbf{N}_k(n, w^\uparrow) = (\alpha_1, \alpha_2, \dots, \alpha_m)$ denote the tuple consisting of all necklaces in $\mathbf{S}_k(n, w^\uparrow)$ listed in lexicographic order. For example,

$$\mathbf{N}_2(4, 6^\uparrow) = (1122, 1212, 1222, 2222).$$

Remarkably, the Granddaddy construction can be generalized to $\mathbf{S}_k(n, w^\uparrow)$. Let

$$\mathcal{G}_k(n, w^\uparrow) = \text{ap}(\alpha_1) \text{ap}(\alpha_2) \cdots \text{ap}(\alpha_m).$$

The sequence $\mathcal{G}_k(n, w^\uparrow)$ is the lexicographically smallest bounded-weight de Bruijn sequence for $\mathbf{S}_k(n, w^\uparrow)$ [SWW14, SWW16]. For example,

$$\mathcal{G}_2(4, 6^\uparrow) = 1122 \cdot 12 \cdot 1222 \cdot 2.$$

Unfortunately, adapting the Granddaddy for strings with an upper bound on the weight does not necessarily lead to a universal cycle. For example, the sequence $1 \cdot 1112 \cdot 1122 \cdot 12$ is not a universal cycle for $\mathbf{S}_2(4, 6_\downarrow)$: the string 2211 is missing.

There are a number of other known constructions for bounded-weight de Bruijn sequences (for instance, see [SWW14, SWW16, GSWW20, SSTW24]); however, no efficient decoding algorithms for them was previously known. Efficient universal cycles are also known for the subsets of $\mathbf{S}_k(n)$ with both a lower and upper bound on weights [SWW13].

2.2 Necklace properties

The following five properties of necklaces can be derived from the definition of a necklace.

► **Property 1.** *If $\alpha = a_1 \cdots a_{j-1} x k^{n-j}$ is in $\mathbf{N}_k(n)$ for some $j \geq 1$ and symbol $x \neq k$, then $a_1 \cdots a_j (x + 1) k^{n-j}$ is also a necklace.*

► **Property 2.** *Let $n, k > 1$ and $k < w < kn$. The first necklace in $\mathbf{N}_k(n, w^\uparrow)$ is $1^{n-j-1} x k^j$, where $j = \lfloor \frac{w-n}{k-1} \rfloor$ and $x = w - (n-j-1) - kj$; it has weight w and is aperiodic.*

► **Property 3.** *Let $n, k > 1$ and $w < kn$. The last two necklaces in $\mathbf{N}_k(n, w^\uparrow)$ are $(k-1)k^{n-1}$ and k^n .*

The next property follows from the definition of a periodic necklace.

► **Property 4.** *Let α_i and α_{i+1} be consecutive necklaces in $\mathbf{N}_k(n, w^\uparrow)$ where $n, k > 1$ and $w < kn$. If α_i is periodic then $\text{ap}(\alpha_i) \cdot \text{ap}(\alpha_{i+1})$ has prefix α_i . Moreover α_{i+1} is aperiodic.*

Proof. Since α_i is periodic, it can be written as $\alpha_i = (a_1 \cdots a_p)^j$ for some $j > 1$. Let ℓ be the largest index in $[1, p]$ such that $a_\ell \neq k$. Such an index exists since α_i is not the last necklace k^n in $\mathbf{N}_k(n, w^\uparrow)$. By Property 1, there exists a necklace with prefix $(a_1 \cdots a_p)^{j-1}$ that is lexicographically larger than α_i and also in $\mathbf{N}_k(n, w^\uparrow)$. Thus α_{i+1} must also have prefix $(a_1 \cdots a_p)^{j-1}$ since $\mathbf{N}_k(n, w^\uparrow)$ is lexicographically ordered. Thus $\text{ap}(\alpha_i) \cdot \text{ap}(\alpha_{i+1})$ has prefix α_i . The fact that α_{i+1} is aperiodic is proved in [SWW16, Lemma 5]. ◀

The following property is proved in [SW17].

► **Property 5.** *Let $\alpha = a_1 a_2 \cdots a_n$ be a necklace and let $1 \leq i \leq j \leq n$. Then $a_i a_{i+1} \cdots a_{j-1} x$ is lexicographically larger than α if $x > a_j$.*

3 Ranking the Granddaddy de Bruijn sequence $\mathcal{G}_k(n)$

In this section we review how the Granddaddy de Bruijn sequence $\mathcal{G}_k(n)$ can be efficiently decoded based on the groundbreaking results of Kociumaka, Radoszewski, and Rytter [KRR16]. Then in Section 4, we adapt the approach to decode the bounded-weight de Bruijn sequence $\mathcal{G}_k(n, w^\uparrow)$.

Let s be a string in $\mathbf{S}_k(n)$. It can be written as $s = \mathbf{p}\mathbf{q}$ where $\mathbf{q}$ is the longest suffix of s such that $\mathbf{qp}$ is a necklace. For example, if $s = 221221$ then $\mathbf{p} = 22$ and $\mathbf{q} = 1221$. Let $\text{rank}(s)$ denote the rank of s in $\mathcal{G}_k(n)$. The last n length- n substrings of $\mathcal{G}_k(n)$ are all strings $s = k^{n-j}1^j$ for $0 \leq j < n$. For these strings $\text{rank}(s) = k^n - (n - j) + 1$. For all other strings, there are two key steps involved in the efficient computation of $\text{rank}(s)$.

Step 1: Determine consecutive necklaces β_1, β_2 , and β_3 in $\mathbf{N}_k(n)$ such that $\text{ap}(\beta_1)\text{ap}(\beta_2)\text{ap}(\beta_3)$ contains s as a substring. The following remark summarizes results from [FM78].

► **Remark 6.** There exist unique consecutive necklaces β_1, β_2 , and β_3 in $\mathbf{N}_k(n)$ (considered cyclically) such that $\mathbf{p}$ is a suffix of β_1 and $\mathbf{q}$ is a prefix of β_2 , i.e., s is a substring of $\beta_1\beta_2$. Moreover, s is a substring of $\text{ap}(\beta_1)\text{ap}(\beta_2)\text{ap}(\beta_3)$.

The necklaces β_1, β_2 , and β_3 from the above remark can be determined in $\mathcal{O}(n^2)$ time using $\mathcal{O}(n)$ space [KRR16].

Step 2: Compute $T_k(n, \sigma_i) = |\text{ap}(\sigma_1)\text{ap}(\sigma_2)\cdots\text{ap}(\sigma_{i-1})|$ which counts the number of strings in $\mathbf{S}_k(n)$ whose necklaces are lexicographically smaller than σ_i . For example,

$$T_2(4, 1122) = |\{1111\}| + |\{1112, 1121, 1211, 2111\}| = 5.$$

The value $T_k(n, \sigma_i)$ can be computed in $\mathcal{O}(n^2)$ time using $\mathcal{O}(n^2)$ space [KRR16].

Putting these two steps together,

$$\text{rank}(s) = \begin{cases} k^n - (n-j) + 1, & \text{if } s = k^{n-j}1^j \text{ for some } 0 \leq j < n; \\ T_k(n, \beta_2), -|\mathbf{p}| + 1, & \text{otherwise.} \end{cases}$$

Example 1 Recall $\mathcal{G}_2(4) = 1111211221212222$. Consider $s = 2112$. It is a substring in the concatenation of $\beta_1 = 1112$ and $\beta_2 = 1122$, where $\mathbf{p} = 2$ and $\mathbf{q} = 112$. Since $T_2(4, 1122) = 5$, the rank of s is $5 - 1 + 1 = 5$. The rank of 2211 is $2^4 - 2 + 1 = 15$.

The string at rank r in $\mathcal{G}_k(n)$ can be determined in $\mathcal{O}(n^3 \log k)$ time by applying a binary search and repeatedly applying the ranking algorithm [KRR16]. See also a reinterpretation of these results in [SW17].

4 Decoding the bounded-weight de Bruijn sequence $\mathcal{G}_k(n, w^\uparrow)$

In this section we adapt the decoding algorithm for the Granddaddy to decode the bounded-weight de Bruijn sequence $\mathcal{G}_k(n, w^\uparrow)$. If $w = k$, then $\mathcal{G}_k(n, w^\uparrow) = \mathcal{G}_k(n)$. If $w = kn$, then $\mathcal{G}_k(n, w^\uparrow)$ is the single symbol k . Throughout the remainder of this section assume that $k < w < kn$ and let:

- $\alpha_1 = \mathbf{a}_1\mathbf{a}_2\cdots\mathbf{a}_n$ be the first necklace in $\mathbf{N}_k(n, w^\uparrow)$,
- $s = \mathbf{p}\mathbf{q}$ be a string in $\mathbf{S}_k(n, w^\uparrow)$, where $\mathbf{q}$ is the longest suffix of s such that $\mathbf{qp}$ is a necklace, and
- β_1 and β_2 be the necklaces defined in Remark 6 for s .

6 Decoding universal cycles

Recall that β_1 has suffix $\mathbf{p}$ and β_2 has prefix $\mathbf{q}$.

To adapt **Step 1** from the previous section, observe that necklaces β_1 and β_2 may not satisfy the required weight constraint. This is illustrated in the following example.

Example 2 Consider $n = 3$, $k = 4$, and $w = 9$. Since

$$\mathbf{N}_4(3, 9^\uparrow) = 144, 234, 243, 244, 333, 334, 344, 444,$$

we have

$$\mathcal{G}_4(3, 9^\uparrow) = 144 \cdot 234 \cdot 243 \cdot 244 \cdot 3 \cdot 334 \cdot 344 \cdot 4.$$

Consider $\mathbf{s} = 423$ where $\mathbf{p} = 4$ and $\mathbf{q} = 23$. In $\mathcal{G}_4(3)$, $\mathbf{s}$ is found as a substring of $\beta_1 \cdot \beta_2 = 224 \cdot 233$. However, in $\mathcal{G}_4(3, 9^\uparrow)$ it is found as a substring of $144 \cdot 234$.

To address this challenge, first consider the strings found in the wraparound of $\mathcal{G}_k(n, w^\uparrow)$. Recall from Property 2 that α_1 is aperiodic. Thus, α_1 is a prefix of $\mathcal{G}_k(n, w^\uparrow)$. From Property 3, $\alpha_{m-1} = (\mathbf{k}-1)\mathbf{k}^{n-1}$ and $\alpha_m = \mathbf{k}^n$. Thus, $\mathcal{G}_k(n, w^\uparrow)$ has suffix $\mathbf{k}^n$. Therefore, if $\mathbf{s}$ is of the form $\mathbf{k}^{n-j}\mathbf{a}_1 \cdots \mathbf{a}_j$ for some $0 \leq j < n$, then it occurs as one of the last n substrings of $\mathcal{G}_k(n, w^\uparrow)$. The rank of such a string $\mathbf{s}$ is $|\mathbf{S}_k(n, w^\uparrow)| - (n - j) + 1$. For all other possible strings $\mathbf{s}$ consider necklaces δ_1 , δ_2 , and δ_3 such that

- δ_1 is the *largest* necklace in $\mathbf{N}_k(n, w^\uparrow)$ that is lexicographically smaller than or equal to β_1 ,
- δ_2 is the *smallest* necklace in $\mathbf{N}_k(n, w^\uparrow)$ that is lexicographically larger than or equal to β_2 , and
- δ_3 is the necklace following δ_2 in $\mathbf{N}_k(n, w^\uparrow)$.

By their definitions δ_1 , δ_2 , and δ_3 appear consecutively in $\mathbf{N}_k(n, w^\uparrow)$. The necklaces δ_2 and δ_3 are well defined due to the constraints on $\mathbf{s}$ and w . The fact that δ_1 is well defined follows from the proof of the following lemma. Ultimately, we will show that $\mathbf{s} = \mathbf{p}\mathbf{q}$ is found in $\mathcal{G}_k(n, w^\uparrow)$ as a substring of $\text{ap}(\delta_1) \text{ap}(\delta_2) \text{ap}(\delta_3)$.

► **Lemma 7.** *The string $\text{ap}(\delta_1)$ has suffix $\mathbf{p}$.*

Proof. First, we show that δ_1 has suffix $\mathbf{p}$. Recall that necklace β_1 has suffix $\mathbf{p}$. Let $j = |\mathbf{p}|$ and let $\beta_1 = \mathbf{b}_1 \cdots \mathbf{b}_{n-j}\mathbf{p}$. If $\mathbf{p} \neq \mathbf{k}^j$, then by Property 1 it must be that β_2 also has prefix $\mathbf{b}_1 \cdots \mathbf{b}_{n-j}$ which is equal to $\mathbf{q}$ by the definition of β_2 . Thus $\beta_1 = \mathbf{q}\mathbf{p} \in \mathbf{N}_k(n, w^\uparrow)$. Then by the definition of δ_1 , we have $\delta_1 = \beta_1$, meaning δ_1 has suffix $\mathbf{p}$. Now suppose $\mathbf{p} = \mathbf{k}^j$. If $\beta_1 \in \mathbf{N}_k(n, w^\uparrow)$, then $\delta_1 = \beta_1$ and thus δ_1 has suffix $\mathbf{p}$. Otherwise $\delta_1 < \beta_1$ and $\beta_1 \notin \mathbf{N}_k(n, w^\uparrow)$. Suppose, for a contradiction, that δ_1 does not have $\mathbf{p}$ as a suffix. If the length- $(n-j)$ prefix of δ_1 is $\mathbf{b}_1 \cdots \mathbf{b}_{n-j}$, then the weight of β_1 is greater than the weight of δ_1 , a contradiction. If the length- $(n-j)$ prefix of δ_1 is not $\mathbf{b}_1 \cdots \mathbf{b}_{n-j}$, then it must be smaller than $\mathbf{b}_1 \cdots \mathbf{b}_{n-j}$. But then by Property 1, there is a necklace in $\mathbf{N}_k(n, w^\uparrow)$ that is larger than δ_1 but smaller than β_1 , a contradiction. Thus δ_1 has suffix $\mathbf{p}$.

Now we show that $\text{ap}(\delta_1)$ has suffix $\mathbf{p}$. If δ_1 is aperiodic, then we have already proved that $\mathbf{p}$ is a suffix of $\text{ap}(\delta_1) = \delta_1$. If δ_1 is periodic then it can be written as $\delta_1 = \gamma^j$ for some $j > 1$ where $\text{ap}(\delta_1) = \gamma$. If $|\mathbf{p}| \leq |\gamma|$ then $\mathbf{p}$ is a suffix of $\text{ap}(\delta_1)$. Otherwise, we have that $\text{ap}(\delta_1) \text{ap}(\delta_2)$ has prefix δ_1 by Property 4. This implies that $\text{ap}(\delta_2)$ begins with γ^{j-1} . Since $|\gamma| < |\mathbf{p}|$, we have that $|\mathbf{q}| = |\delta_1| - |\mathbf{p}| < |\delta_1| - |\gamma| = |\gamma^{j-1}|$, which means that $\mathbf{q}$ is a prefix of $|\gamma^{j-1}|$, and is therefore a prefix of δ_1 . But then $\delta_1 = \mathbf{q}\mathbf{p}$ and δ_1 is a cyclic shift of $\mathbf{s}$, which means that $\mathbf{s}$ is periodic with the same period as δ_1 . This contradicts $\mathbf{q}$ being the longest suffix of $\mathbf{s}$ such that $\mathbf{q}\mathbf{p}$ is a necklace. Thus $\mathbf{p}$ is a suffix of $\text{ap}(\delta_1)$. ◀

► **Lemma 8.** *The string $\text{ap}(\delta_2) \text{ap}(\delta_3)$ has prefix $\mathbf{q}$.*

Proof. First, we show that δ_2 has prefix $\mathbf{q}$. Recall that β_2 has prefix $\mathbf{q}$. Let $j = |\mathbf{p}|$. Since $\mathbf{s} = \mathbf{pq}$ is in $\mathbf{S}_k(n, w^\uparrow)$ then $\mathbf{qk}^j$ is also in $\mathbf{S}_k(n, w^\uparrow)$. Since $\mathbf{qp}$ is a necklace, then $\mathbf{qk}^j$ is a necklace by repeated application of Property 1. Thus, since δ_2 is the smallest necklace in $\mathbf{N}_k(n, w^\uparrow)$ that is lexicographically larger than β_2 , δ_2 must have prefix $\mathbf{q}$.

Now we show that $\text{ap}(\delta_2) \text{ap}(\delta_3)$ has prefix $\mathbf{q}$. If δ_2 is aperiodic then clearly $\mathbf{q}$ is a prefix of $\text{ap}(\delta_2)$. If δ_2 is periodic, then it cannot be $\mathbf{k}^n$ by the restriction on $\mathbf{s}$. Additionally, we have that $\text{ap}(\delta_2) \text{ap}(\delta_3)$ has prefix δ_2 by Property 4. Thus $\mathbf{q}$ is a prefix of $\text{ap}(\delta_2) \text{ap}(\delta_3)$. ◀

► **Corollary 9.** *The string $\mathbf{s} = \mathbf{pq}$ is a substring of $\text{ap}(\delta_1) \text{ap}(\delta_2) \text{ap}(\delta_3)$.*

To adapt **Step 2** from the previous section, we compute $T_k(n, w, \alpha)$ which counts the number of strings in $\mathbf{S}_k(n, w^\uparrow)$ whose necklaces are lexicographically smaller than a necklace α . Note that $T_k(n, w, \alpha_i) = |\text{ap}(\alpha_1) \text{ap}(\alpha_2) \cdots \text{ap}(\alpha_{i-1})|$. Let $\text{rank}'(\mathbf{s})$ denote the rank of $\mathbf{s}$ in $\mathcal{G}_k(n, w^\uparrow)$ and let $S_k(n, w^\uparrow) = |\mathbf{S}_k(n, w^\uparrow)|$. Putting this all together, we can compute $\text{rank}'(\mathbf{s})$ as follows.

$$\text{rank}'(\mathbf{s}) = \begin{cases} S_k(n, w^\uparrow) - (n - j) + 1 & \text{if } \mathbf{s} = \mathbf{k}^{n-j} \mathbf{a}_1 \cdots \mathbf{a}_j \text{ for some } 0 \leq j < n, \\ T_k(n, w, \beta_2) - |\mathbf{p}| + 1 & \text{otherwise.} \end{cases}$$

Note that is not necessary to compute δ_2 , since $T_k(n, w, \delta_2) = T_k(n, w, \beta_2)$.

It remains to analyze the running time required to compute $\text{rank}'(\mathbf{s})$. The following recurrence can be applied to enumerate $S_k(n, w^\uparrow)$ for $n, k \geq 0$ and $0 \leq w \leq kn$:

$$S_k(n, w^\uparrow) = \sum_{j=1}^k S_k(n-1, \max(0, w-j)^\uparrow), \quad (1)$$

with base cases $S_k(n, w^\uparrow) = k^n$ for $w \leq n$ and $S_k(0, w^\uparrow) = 0$ for $w > 0$. Applying dynamic programming, the value $S_k(n, w^\uparrow)$ can be computed in $\mathcal{O}(n^2 k^2)$ time and $\mathcal{O}(n^2 k)$ space. In Section 7, the following lemma is proved.

► **Lemma 10.** *Let $n, k > 1$. For any $\alpha \in \mathbf{N}_k(n)$ the value $T_k(n, w, \alpha)$ can be computed in $\mathcal{O}(n^3 k^2)$ time using $\mathcal{O}(n^3 k)$ space.*

Recall that β_2 can be computed in $\mathcal{O}(n^2)$ time using $\mathcal{O}(n)$ space. Putting this all together we obtain the following theorem.

► **Theorem 11.** *The universal cycle $\mathcal{G}_k(n, w^\uparrow)$ can be ranked in $\mathcal{O}(n^3 k^2)$ time using $\mathcal{O}(n^3 k)$ space.*

4.1 Unranking

In this section, we demonstrate how to determine the string $\mathbf{s} = \mathbf{s}_1 \mathbf{s}_2 \cdots \mathbf{s}_n$ at rank r in $\mathcal{G}_k(n, w^\uparrow)$. We consider two cases depending on the value of r .

Case 1: $S_k(n, w^\uparrow) - n + 1 < r \leq S_k(n, w^\uparrow)$. In this case, $\mathbf{s}$ is found in the wraparound. Recall that $\mathbf{a}_1 \cdots \mathbf{a}_n$ are the first n symbols in $\mathcal{G}_k(n, w^\uparrow)$ as defined in Property 2, and the last n symbols are $\mathbf{k}^n$ (from Property 3). Thus, $\mathbf{s}$ can be returned in $\mathcal{O}(n)$ time.

Case 2 : $1 \leq r \leq S_k(n, w^\uparrow) - n + 1$. By the construction of $\mathcal{G}_k(n, w^\uparrow)$ and Property 4, the ranks of the necklaces as they appear in $\mathbf{N}_k(n, w^\uparrow)$ are strictly increasing. Let $\text{SMALLESTNECK}(r)$ denote the smallest necklace in $\mathbf{N}_k(n, w^\uparrow)$ with rank greater than or equal to r in $\mathcal{G}_k(n, w^\uparrow)$. If $\gamma_1 = \text{SMALLESTNECK}(r)$, then let

8 Decoding universal cycles

r' denote the rank of γ_1 in $\mathcal{G}_k(n, w^\uparrow)$. If $r = r'$, then $s = \gamma_1$. Otherwise, let $\gamma_2 = \text{SMALLESTNECK}(r' - n)$. By Property 4, γ_2 and γ_1 appear consecutively in $\mathbf{N}_k(n, w^\uparrow)$, at most one of the two necklaces is periodic, and s is obtained by concatenating the last $r' - r$ elements from γ_2 with the first $n - (r' - r)$ elements from γ_1 .

Example 3 Let $n = 3, k = 4$, and $w = 9$, and consider $r = 3$. Recall that

$$\mathcal{G}_4(3, 9^\uparrow) = 144 \cdot 234 \cdot 243 \cdot 244 \cdot 3 \cdot 334 \cdot 344 \cdot 4.$$

The necklace $\gamma_1 = \text{SMALLESTNECK}(r) = 234$ has rank $r' = 4$ and $\gamma_2 = \text{SMALLESTNECK}(r' - n) = 144$. Since $r' - r = 1$, the string at rank $r = 3$ is the last symbol of γ_2 concatenated with the first $n - 1 = 2$ symbols of γ_1 . It is $s = 423$.

The necklace $\text{SMALLESTNECK}(r)$ can be computed by making $\mathcal{O}(n \log k)$ rank queries as illustrated in Algorithm 1. It follows the same approach as the algorithm in [SW17] adding the weight constraint. The algorithm iteratively determines each symbol of $\tau = t_1 \cdots t_n$ starting with the first symbol. The loop invariant after i iterations maintains the property that $t_1 \cdots t_i k^{n-i}$ is a necklace with rank greater than or equal to r such that $t_1 \cdots t_{i-1}(t_i - 1)k^{n-i}$ is either not in $\mathbf{N}_k(n, w^\uparrow)$, or it is in $\mathbf{N}_k(n, w^\uparrow)$ and has rank less than r . Binary search, is applied to determine the i -th symbol using at most $\log k$ ranking queries.

Algorithm 1 Compute the smallest necklace $t_1 t_2 \cdots t_n$ in $\mathbf{N}_k(n, w^\uparrow)$ with rank greater than or equal to r , where $1 \leq r \leq S_k(n, w^\uparrow) - r + 1$

```

1: function SMALLESTNECK( $r$ )
2:    $\triangleright$  Apply a binary search to find each  $t_i$ 
3:   for  $i$  from 1 to  $n$  do
4:      $min \leftarrow 1$ 
5:      $max \leftarrow k$ 
6:      $t_i \leftarrow k$ 
7:     while  $min < max$  do
8:        $prev \leftarrow t_i$ 
9:        $v \leftarrow (min + max)/2$ 
10:       $t_i \leftarrow v$ 
11:      if  $t_1 \cdots t_i k^{n-i} \in \mathbf{N}_k(n, w^\uparrow)$  and  $\text{rank}'(t_1 \cdots t_i k^{n-i}) \geq r$  then  $max \leftarrow v$ 
12:      else
13:         $t_i \leftarrow prev$ 
14:         $min \leftarrow v + 1$ 
15:   return  $t_1 t_2 \cdots t_n$ 

```

► **Theorem 12.** The universal cycle $\mathcal{G}_k(n, w^\uparrow)$ can be unranked in $\mathcal{O}(n^4 k^2 \log k)$ time using $\mathcal{O}(n^3 k)$ space.

Proof. Testing whether a string is a necklace can be determined in $\mathcal{O}(n)$ time/space [Boo80]. Thus, the running time and space for the described unranking algorithm is dominated by the $\mathcal{O}(n \log k)$ calls to the ranking function. ◀

5 An upper bound on weight

Recall that $\mathbf{S}_k(n, w_\downarrow)$ denotes the subset of $\mathbf{S}_k(n)$ containing strings with weight *at most* w . Let the *complement* of a string s , denoted by $\text{comp}(s)$, be obtained by interchanging each symbol x with $k - x + 1$. If s has weight w , then $\text{comp}(s)$ has weight $kn - w + n$.

Example 4 Let $n = 5$, $k = 4$, and consider $s = 12443$. It has weight $w = 14$. Then $\text{comp}(s) = 43112$ has weight $20 - 14 + 5 = 11$; the symbols 1 and 4 are interchanged, and the symbols 2 and 3 are interchanged.

The complement of a universal cycle for $\mathbf{S}_k(n, (kn - w + n)^\uparrow)$ is a universal cycle for $\mathbf{S}(n, w_\downarrow)$, and vice versa. Thus, $\mathcal{G}_k(n, w_\downarrow) = \text{comp}(\mathcal{G}_k(n, kn - w + n)^\uparrow)$ is a universal cycle for $\mathbf{S}_k(n, w_\downarrow)$. To rank a string s in $\mathcal{G}_k(n, w_\downarrow)$ simply return the rank of $\text{comp}(s)$ in $\mathcal{G}_k(n, (kn - w + n)^\uparrow)$.

► **Corollary 13.** *The universal cycle $\mathcal{G}_k(n, w_\downarrow)$ can be ranked in $\mathcal{O}(n^3 k^2)$ time and unranked in $\mathcal{O}(n^4 k^2 \log k)$ time using $\mathcal{O}(n^3 k)$ space.*

6 Applications to t -subsets and t -multisets

Recall that $\mathbf{Subset}(n, t)$ denotes the set of all t -subsets of $\{1, 2, \dots, n\}$. Using difference representatives for each subset, a universal cycle for $\mathbf{Subset}(n, t)$ is precisely a universal cycle for $\mathbf{S}_{n-t+1}(t, n_\downarrow)$ [CJJS25]. Thus, we can apply the results from Corollary 13 to efficiently rank a universal cycle for $\mathbf{Subset}(n, t)$.

► **Theorem 14.** *The sequence $\mathcal{G}_{n-t+1}(t, n_\downarrow)$ is a universal cycle for $\mathbf{Subset}(n, t)$ applying the difference representatives. Using $\mathcal{O}(nt^3)$ space, the sequence can be ranked in $\mathcal{O}(n^2 t^3)$ time and unranked in $\mathcal{O}(n^2 t^4 \log n)$ time.*

Example 5 The sequence $\mathcal{G}_3(3, 5_\downarrow) = 3112212111$ is a universal cycle for $\mathbf{Subset}(5, 3)$ using difference representatives. It can be efficiently decoded using its complement $\mathcal{G}_3(3, 7^\uparrow) = 1332232333$ as illustrated in the following table.

Rank	String in $\mathcal{G}_3(3, 7^\uparrow)$	Complement (diff rep)	Subset over $\{1, 2, 3, 4, 5\}$
1	133	311	$\{3, 4, 5\}$
2	332	112	$\{1, 2, 4\}$
3	322	122	$\{1, 3, 5\}$
4	223	221	$\{2, 4, 5\}$
5	232	212	$\{2, 3, 5\}$
6	323	121	$\{1, 3, 4\}$
7	233	211	$\{2, 3, 4\}$
8	333	111	$\{1, 2, 3\}$
9	331	113	$\{1, 2, 5\}$
10	313	131	$\{1, 4, 5\}$

Recall that $\mathbf{Multiset}(n, t)$ denotes the set of all t -multisets of $\{0, 2, \dots, n-1\}$. Using difference representatives for each multiset and incrementing all symbols by 1, a universal cycle for $\mathbf{Multiset}(n, t)$ is precisely a universal cycle for $\mathbf{S}_n(t, (n+t-1)_\downarrow)$ [CJJS25]. Again, we can apply the results from Corollary 13 to efficiently decode a universal cycle for $\mathbf{Multiset}(n, t)$.

► **Theorem 15.** *The sequence $\mathcal{G}_n(t, (n+t-1)_\downarrow)$ is a universal cycle for $\mathbf{Multiset}(n, t)$ applying the adjusted difference representatives. Using $\mathcal{O}(nt^3)$ space, the sequence can be ranked in $\mathcal{O}(n^2 t^3)$ time and unranked in $\mathcal{O}(n^2 t^4 \log n)$ time.*

Example 6 The sequence 2001101000 is a universal cycle for $\text{Multiset}(3, 3)$ using difference representatives. By incrementing each symbol in the sequence by one, we obtain $\mathcal{G}_3(3, 5_\downarrow) = 3112212111$. It can be efficiently decoded using its complement $\mathcal{G}_3(3, 7^\uparrow) = 1332232333$ as illustrated in the following table.

Rank	String in $\mathcal{G}_3(3, 7^\uparrow)$	Complement minus 1 (diff rep)	Multiset over $\{0, 1, 2\}$
1	133	200	$\{2, 2, 2\}$
2	332	001	$\{0, 0, 1\}$
3	322	011	$\{0, 1, 2\}$
4	223	110	$\{1, 2, 2\}$
5	232	101	$\{1, 1, 2\}$
6	323	010	$\{0, 1, 1\}$
7	233	100	$\{1, 1, 1\}$
8	333	000	$\{0, 0, 0\}$
9	331	002	$\{0, 0, 2\}$
10	313	020	$\{0, 2, 2\}$

7 Proof of Lemma 10: Enumerating $T_k(n, w, \alpha)$

In this section we demonstrate how to efficiently compute $T_k(n, w, \alpha)$, where α is a necklace in $\mathbf{N}_k(n, w^\uparrow)$. The approach follows the methods as applied in [SW17, HS19]. To simplify notation going forward assume that n, k and the necklace $\alpha = a_1 a_2 \cdots a_n$ are fixed.

7.1 Enumerating preliminary sets

In order to efficiently compute $T_k(n, w, \alpha)$, we need to efficiently determine the cardinality of two special sets of strings that have both prefix and suffix restrictions. For $t \geq j$, let $\mathbf{B}(t, j, w)$ denote the subset of strings in $\mathbf{S}_k(t, w)$ where each string has prefix $a_1 a_2 \cdots a_j$ and every non-empty suffix is greater than α . When $j = 0$, there is no prefix restriction on the strings.

Example 7 Let $\alpha = 112122$ and $k = 3$. Then the strings in $\mathbf{B}(6, 3, 11)$ are:

112133 112223 112313
 112332 112322
 112233 112323
 112332
 112333

The string in the first column can be given by $\mathbf{B}(6, 4, 11)$, while the underlined strings in the second and third columns correspond to the sets $\mathbf{B}(2, 0, 5)$ and $\mathbf{B}(2, 0, 4)$, respectively.

Let $B(t, j, w) = |\mathbf{B}(t, j, w)|$. An enumeration formula for $B(t, j, w)$, where $0 \leq j \leq t \leq n$, can be derived based on the recursive structure illustrated in the previous example. Considering the empty string, $B(0, 0, w) = 1$ if $w \leq 0$ and $B(0, 0, w) = 0$ if $w > 0$. When $t > 0$, $B(t, t, w) = 0$ because the suffix $a_1 a_2 \cdots a_t$ is

lexicographically smaller than or equal to α . Otherwise, the strings in $\mathbf{B}(t, j, w)$ can be partitioned based on the symbol in position $j+1$.

- ▷ If the $j+1$ st symbol is smaller than a_{j+1} then the suffix starting from the first index would be smaller than α , a contradiction.
- ▷ If the $j+1$ st symbol is a_{j+1} then the number of such strings is $B(t, j+1, w)$.
- ▷ If the $j+1$ st symbol is larger than a_{j+1} , then any suffix starting at index $1, 2, \dots, j+1$ is larger than α by Lemma 5. Thus, for each $x > a_{j+1}$, the remaining $t-j-1$ positions can be filled in $B(t-j-1, 0, w - \text{weight}(a_1 \cdots a_j x))$ ways.

Thus, for $0 \leq j \leq t \leq n$ and $w \geq 0$:

$$B(t, j, w) = \begin{cases} 1 & \text{if } j = t = 0 \text{ and } w \leq 0, \\ 0 & \text{if } j = t, \\ B(t, j+1, w) + \sum_{x=a_{j+1}+1}^k B(t-j-1, 0, w - \text{weight}(a_1 \cdots a_j x)) & \text{otherwise.} \end{cases}$$

Now consider the set $\mathbf{P}(t, j, w)$ which denotes the subset of $\mathbf{S}_k(t)$ where each string has weight *exactly* w and prefix $a_1 a_2 \cdots a_j$ such that every non-empty suffix is greater than α . Let $P(t, j, w) = |\mathbf{P}(t, j, w)|$. Applying the same recursive strategy applied for computing $B(t, j, w)$, but with different base cases, we obtain the following recurrence for $0 \leq j \leq t$:

$$P(t, j, w) = \begin{cases} 1 & \text{if } j = t = w = 0, \\ 0 & \text{if } j = t \text{ or } w \leq 0, \\ P(t, j+1, w) + \sum_{x=a_{j+1}+1}^k P(t-j-1, 0, w - \text{weight}(a_1 \cdots a_j x)) & \text{otherwise.} \end{cases}$$

7.2 Computing $T_k(n, w, \alpha)$

Let $\mathbf{T}_k(n, w, \alpha)$ denote the subset of strings in $\mathbf{S}_k(n, w^\dagger)$ whose necklaces are lexicographically smaller than a given necklace α . Note that $T_k(n, w, \alpha) = |\mathbf{T}_k(n, w, \alpha)|$. To compute $T_k(n, w, \alpha)$, we partition $\mathbf{T}_k(n, w, \alpha)$ into subsets following the approaches used in [SW17, HS19].

Let $\mathbf{A}(t, j)$ denote the subset of strings in $\mathbf{T}_k(n, w, \alpha)$ of the form $s = s_1 s_2 \cdots s_n$ where t is the smallest integer such that $s' = s_t s_{t+1} \cdots s_1 s_2 \cdots s_{t-1} < \alpha$ and j is the largest integer such that $a_1 \cdots a_j$ is a prefix of s' . Let $A(t, j) = |\mathbf{A}(t, j)|$. Then

$$T_k(n, w, \alpha) = \sum_{t=1}^n \sum_{j=0}^n A(t, j).$$

Example 8 The set $\mathbf{T}_3(5, 10, 12313)$ contains 40 strings from the equivalence classes represented by the eight necklaces: $\{11233, 11323, 11332, 11333, 12133, 12223, 12232, 12233\}$ that appear before 12313 in $\mathbf{N}_3(5, 10)$. These strings can be partitioned into 10 subsets of the form $\mathbf{A}(t, j)$ as follows:

$\mathbf{A}(t, j)$	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
$j = 1$	11233, 11323	31123, 31132	33112, 23113	22311, 32311	12231, 13231
	11332, 11333	21133, 31113	33112, 33112	33211, 33311	13321, 13331
$j = 2$	12233, 12223	31213, 31222	33121, 33121	13312, 22312	21331, 22231
	12232, 12232	21223, 31223	32122, 33122	23212, 23312	22321, 22331

For $j \in \{0, 3, 4, 5\}$, $\mathbf{A}(t, j) = \emptyset$ for all values of t .

Clearly $A(t, n) = 0$ for all $1 \leq t \leq n$. Otherwise for $j < n$, each set $\mathbf{A}(t, j)$ falls into one of the following two cases.

Case 1 ($t + j \leq n$). Each $s \in \mathbf{A}(t, j)$ is of the form $\mathbf{r} \mathbf{a}_1 \mathbf{a}_2 \cdots \mathbf{a}_j \mathbf{x} \mathbf{q}$ where:

- ▷ $\mathbf{r} \in \mathbf{S}_k(t-1, w')$ such that every non-empty suffix is larger than α ,
- ▷ $\mathbf{x} < \mathbf{a}_{j+1}$,
- ▷ $\mathbf{q} \in \mathbf{S}_k(n-t-j, w-w'-\text{weight}(\mathbf{a}_1 \cdots \mathbf{a}_j \mathbf{x}))$, and
- ▷ $\text{weight}(\mathbf{s}) \geq w$.

Recall that the number of possibilities for $\mathbf{r}$ with a fixed weight w' is given by $P(t-1, 0, w')$. For each $\mathbf{r}$ with weight w' and given $\mathbf{x}$, there are $S_k(n-t-j, w-w'-\text{weight}(\mathbf{a}_1 \cdots \mathbf{a}_j \mathbf{x}))$ possibilities for $\mathbf{q}$. Thus, in this case

$$A(t, j) = \sum_{\mathbf{x}=1}^{\mathbf{a}_{j+1}-1} \sum_{w'=t-1}^{k(t-1)} P(t-1, 0, w') S_k(n-t-j, w-w'-\text{weight}(\mathbf{a}_1 \cdots \mathbf{a}_j \mathbf{x})).$$

Case 2 ($t + j > n$). Each $s \in \mathbf{A}(t, j)$ is of the form $\mathbf{a}_{n-t+2} \cdots \mathbf{a}_{j-1} \mathbf{a}_j \mathbf{x} \mathbf{r} \mathbf{a}_1 \mathbf{a}_2 \cdots \mathbf{a}_{n-t+1}$ where:

- ▷ $\mathbf{x} < \mathbf{a}_{j+1}$,
- ▷ $\mathbf{r} \in \mathbf{S}(n-j-1, w-\text{weight}(\mathbf{a}_1 \cdots \mathbf{a}_j \mathbf{x}))$, and
- ▷ every non-empty suffix of $\mathbf{a}_{n-t+2} \cdots \mathbf{a}_{j-1} \mathbf{a}_j \mathbf{x} \mathbf{r}$ is larger than α .³

Let $\mathbf{q} = \mathbf{a}_{n-t+2} \cdots \mathbf{a}_{j-1} \mathbf{a}_j$. Since $\mathbf{q}$ is a substring of the necklace α , any suffix of $\mathbf{q}$ must be lexicographically greater than or equal to the prefix of α with the same length (by the definition of a necklace). Therefore we determine the *longest* suffix of $\mathbf{q}$ that is equal to the prefix of α with the same length. Suppose this suffix has length z . This implies $\mathbf{a}_{j-z+1} \cdots \mathbf{a}_{j-1} \mathbf{a}_j = \mathbf{a}_1 \mathbf{a}_2 \cdots \mathbf{a}_z$. This means any suffix of $\mathbf{s}$ starting from an index less than or equal to $|\mathbf{q}| - z$ is larger than α . Consider three sub-cases depending on $\mathbf{x}$.

- If $\mathbf{x} < \mathbf{a}_{z+1}$ then $\mathbf{a}_{j-z+1} \cdots \mathbf{a}_{j-1} \mathbf{a}_j \mathbf{x}$ is smaller than α , a contradiction.
- If $\mathbf{x} = \mathbf{a}_{z+1}$ then $\mathbf{a}_{j-z+1} \cdots \mathbf{a}_{z+1} \mathbf{r} \in \mathbf{B}(n-j+z, z+1, w-\text{weight}(\mathbf{a}_1 \cdots \mathbf{a}_{j-z}))$.
- If $\mathbf{x} > \mathbf{a}_{z+1}$ then any suffix of $\mathbf{s}$ starting from an index less than or equal to $|\mathbf{q}| + 1$ will be larger than α by Property 5. Thus, $\mathbf{r} \in \mathbf{B}(n-j-1, 0, w-\text{weight}(\mathbf{a}_1 \cdots \mathbf{a}_j \mathbf{x}))$.

Thus, if $\mathbf{a}_{j+1} > \mathbf{a}_{z+1}$

$$A(t, j) = B(n-j+z, z+1, w-\text{weight}(\mathbf{a}_1 \cdots \mathbf{a}_{j-z})) + \sum_{\mathbf{x}=\mathbf{a}_{z+1}+1}^{\mathbf{a}_{j+1}-1} B(n-j-1, 0, w-\text{weight}(\mathbf{a}_1 \cdots \mathbf{a}_j \mathbf{x})).$$

Otherwise $A(t, j) = 0$.

7.3 Analysis

By applying dynamic programming, all necessary $B(t, j, w)$ and $P(t, j, w)$ can be computed in $\mathcal{O}(n^3 k^2)$ time using $\mathcal{O}(n^3 k)$. With these values precomputed, $A(t, j)$ can be computed in $\mathcal{O}(nk^2)$ time for **Case 1** and $\mathcal{O}(k)$ time for **Case 2**, and thus $T_k(n, w, \alpha)$ can be computed in $\mathcal{O}(n^3 k^2)$ time. This proves Lemma 10.

³ This follows from the definition of t noting that $|\mathbf{a}_{n-t+2} \cdots \mathbf{a}_{j-1} \mathbf{a}_j \mathbf{x} \mathbf{r}| < n$.

8 Acknowledgment

Joe Sawada (grant RGPIN-2025-03961) and Daniel Gabrić (grant RGPIN-2026-04863) gratefully acknowledge research support from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

- BG11** A. Blanca and A. P. Godbole. On universal cycles for new classes of combinatorial structures. *SIAM Journal on Discrete Mathematics*, 25(4):1832–1842, 2011.
- BM93** J. Burns and C. Mitchell. Position sensing coding schemes. In *Cryptography and Coding III (M.J.Ganley, ed.)*, pages 31–66. Oxford University Press, 1993.
- Boo80** K. S. Booth. Lexicographically least circular substrings. *Inform. Process. Lett.*, 10(4/5):240–242, 1980.
- CDG92** F. Chung, P. Diaconis, and R. Graham. Universal cycles for combinatorial structures. *Discrete Mathematics*, 110(1):43–59, 1992.
- CGGP21** A. Cantwell, J. Geraci, A. Godbole, and C. Padilla. Graph universal cycles of combinatorial objects. *Advances in Applied Mathematics*, 127:102166, 2021.
- CJS25** C. Campbell, L. Janik-Jones, and J. Sawada. Universal cycles for k -subsets and k -multisets of an n -set, *under review*, 2025.
- db25** De Bruijn sequence and universal cycle constructions (2025). <http://debruijnsequence.org>, 2025.
- DG11** P. Diaconis and R. Graham. *Magical Mathematics: The Mathematical Ideas That Animate Great Magic Tricks*. Princeton University Press, 2011.
- Etz24** T. Etzion. *Sequences and the de Bruijn Graph*. Academic Press, 2024.
- FM78** H. Fredricksen and J. Maiorana. Necklaces of beads in k colors and k -ary de Bruijn sequences. *Discrete Math.*, 23(3):207–210, 1978.
- Fre70** H. Fredricksen. The lexicographically least de Bruijn cycle. *J. Combinatorial Theory*, 9:1–5, 1970.
- Fre82** H. Fredricksen. A survey of full length nonlinear shift register cycle algorithms. *Siam Review*, 24(2):195–221, 1982.
- GSWW20** D. Gabric, J. Sawada, A. Williams, and D. Wong. A successor rule framework for constructing k -ary de Bruijn sequences and universal cycles. *IEEE Transactions on Information Theory*, 66(1):679–687, 2020.
- HJZ09** G. Hurlbert, T. Johnson, and J. Zahl. On universal cycles for multisets. *Discrete Mathematics*, 309(17):5321–5327, 2009. Generalisations of de Bruijn Cycles and Gray Codes/Graph Asymmetries/Hamiltonicity Problem for Vertex-Transitive (Cayley) Graphs.
- HRW12** A. E. Holroyd, F. Ruskey, and A. Williams. Shorthand universal cycles for permutations. *Algorithmica*, 64(2):215–245, 2012.
- HS19** P. Hartman and J. Sawada. Ranking and unranking fixed-density necklaces and Lyndon words. *Theoretical Computer Science*, 791:36–47, 2019.
- Hur94** G. Hurlbert. On universal cycles for k -subsets of an n -set. *SIAM Journal on Discrete Mathematics*, 7(4):598–604, 1994.
- Jac93** B. Jackson. Universal cycles of k -subsets and k -permutations. *Discrete Mathematics*, 117:141–150, 07 1993.
- JSH09** B. Jackson, B. Stevens, and G. Hurlbert. Research problems on Gray codes and universal cycles. *Discrete Mathematics*, 309(17):5341–5348, 2009.
- Knu11** D. E. Knuth. *The Art of Computer Programming. Vol. 4A. Combinatorial Algorithms. Part 1*. Addison-Wesley, Upper Saddle River, NJ, 2011.
- KRR16** T. Kociumaka, J. Radoszewski, and W. Rytter. Efficient ranking of Lyndon words and decoding lexicographically minimal de Bruijn sequence. *SIAM J. Discrete Math*, 30(4):2027–2046, 2016.
- Mar34** M. H. Martin. A problem in arrangements. *Bull. Amer. Math. Soc.*, 40(12):859–864, 1934.

- MEP96** C. Mitchell, T. Etzion, and K. Paterson. A method for constructing decodable de Bruijn sequences. *IEEE Transactions on Information Theory*, 42(5):1472–1478, 1996.
- RSW12** F. Ruskey, J. Sawada, and A. Williams. De Bruijn sequences for fixed-weight binary strings. *SIAM Journal on Discrete Mathematics*, 26(2):605–617, 2012.
- Rud13** Y. Rudoy. An inductive approach to constructing universal cycles on the k -subsets of $[n]$. *The Electronic Journal of Combinatorics*, 20:P18, 2013.
- RW10** F. Ruskey and A. Williams. An explicit universal cycle for the $(n-1)$ -permutations of an n -set. *ACM Trans. Algorithms*, 6(3):1–12, July 2010.
- SSTW24** J. Sawada, J. Sears, A. Trautrim, and A. Williams. Concatenation trees: A framework for efficient universal cycle and de Bruijn sequence constructions. *arXiv preprint arXiv:2308.12405*, 2024.
- SW17** J. Sawada and A. Williams. Practical algorithms to rank necklaces, Lyndon words, and de Bruijn sequences. *Journal of Discrete Algorithms*, 43:95 – 110, 2017.
- SWW13** J. Sawada, A. Williams, and D. Wong. Universal cycles for weight-range binary strings. In *Combinatorial Algorithms: 24th International Workshop, IWOCA 2013, Rouen, France, July 10-12, 2013, Revised Selected Papers 24*, pages 388–401. Springer, 2013.
- SWW14** J. Sawada, A. Williams, and D. Wong. The lexicographically smallest universal cycle for binary strings with minimum specified weight. *Journal of Discrete Algorithms*, 28:31–40, 2014. StringMasters 2012, 2013 Special Issue (Volume 1).
- SWW16** J. Sawada, A. Williams, and D. Wong. Generalizing the classic greedy and necklace constructions of de Bruijn sequences and universal cycles. *Electron. J. Combin.*, 23(1):Paper 1.24, 20, 2016.
- Tul01** J. Tuliani. De Bruijn sequences with efficient decoding algorithms. *Discrete Mathematics*, 226(1):313 – 336, 2001.