Ranking Bracelets in Polynomial Time

Duncan Adamson
Leverhulme Research Centre for Functional Materials Design, Department of Computer Science, University of Liverpool, UK

Vladimir V. Gusev
Leverhulme Research Centre for Functional Materials Design, Department of Computer Science, University of Liverpool, UK

Igor Potapov
Department of Computer Science, University of Liverpool, UK

Argyrios Deligkas
Department of Computer Science, Royal Holloway University of London, UK

Abstract

The main result of the paper is the first polynomial-time algorithm for ranking bracelets. The time-complexity of the algorithm is $O(k^2 \cdot n^4)$, where k is the size of the alphabet and n is the length of the considered bracelets. The key part of the algorithm is to compute the rank of any word with respect to the set of bracelets by finding three other ranks: the rank over all necklaces, the rank over palindromic necklaces, and the rank over enclosing apalindromic necklaces. The last two concepts are introduced in this paper. These ranks are key components to our algorithm in order to decompose the problem into parts. Additionally, this ranking procedure is used to build a polynomial-time unranking algorithm.

2012 ACM Subject Classification Mathematics of computing → Combinatorics on words

Keywords and phrases Bracelets, Ranking, Necklaces

Digital Object Identifier 10.4230/LIPIcs.CPM.2021.4

Related Version *Full Version*: `https://arxiv.org/abs/2104.04324`

Funding *Igor Potapov*: Partially funded by EP/R018472/1 and Royal Society Leverhulme Trust Senior Research Fellowship.

Acknowledgements The authors thank the Leverhulme Trust for funding this research via the Leverhulme Research Centre for Functional Materials Design and the reviewers for their helpful comments that improved the quality of the paper.

1 Introduction

Counting, ordering, and generating basic discrete structures such as strings, permutations, set-partitions, etc. are fundamental tasks in computer science. A variety of such algorithms are assembled in the fourth volume of the prominent series "The art of computer programming" by D. Knuth [10]. Nevertheless, this research direction remains very active [8].

If the structures under consideration are linearly ordered, e.g. a set of words under the dictionary (lexicographic) order, then a unique integer can be assigned to every structure. The *rank* (or index) of a structure is the number of structures that are smaller than it. The *ranking* problem asks to compute the rank of a given structure, while the *unranking* problem corresponds to its reverse: compute the structure of a given rank. Ranking has been studied for various objects including partitions [19], permutations [13, 14], combinations [18], etc. Unranking has similarly been studied for objects such as permutations [14] and trees [7, 15].

1.	aaaaaaaa	7.	aaaababb	13.	aaabbabb	19.	aababbbb	25.	ababbabb
2.	aaaaaaab	8.	aaaabbbb	14.	aaabbbbb	20.	aabbaabb	26.	ababbbbb
3.	aaaaaabb	9.	aaabaaab	15.	aabaabab	21.	aabbabbb	27,	abbabbbb
4.	aaaaabab	10.	aaabaabb	16.	aabaabbb	22.	aabbbbbb	28.	abbbabbb
5.	aaaaabbb	11.	aaababab	17.	aabababb	23.	abababab	29.	abbbbbbb
6.	aaaabaab	12.	aaabbabb	18.	aababbab	24.	abababbb	30.	bbbbbbbb

Figure 1 List of all bracelets of length 8 over the alphabet $\{a, b\}$.

Both ranking and unranking are straightforward for the set of all words over a finite alphabet (assuming the standard lexicographic order), but they immediately cease to be so, as soon as additional symmetry is introduced. One of such examples is a class of necklaces [6]. A *necklace*, also known as a *cyclic word*, is an equivalence class of all words under the cyclic shift operation. Necklaces are classical combinatorial objects and they remain an object of study in other contexts such as total search problems [5] or circular splicing systems [4].

The *rank* of a word w for a given set S and its ordering is the number of words in S that are smaller than w. Often the set is a class of words, for instance all words of a given length over some alphabet. The first class of cyclic words to be ranked were *Lyndon words* - fixed length aperiodic cyclic words - by Kociumaka et. al. [11] who provided an $O(n^3)$ time algorithm. An algorithm for ranking necklaces - fixed length cyclic words - was given by Kopparty et. al. [12], without tight bounds on the complexity. A quadratic algorithm for ranking necklaces was provided by Sawada et al. [16].

This paper answers the open problem of ranking *bracelets*, posed by Sawada and Williams [16]. Bracelets are necklaces that are minimal under both *cyclic shifts* and *reflections*. Figure 1 provides an example of the ranks of length 8 bracelets over a binary alphabet. Bracelets have been studied extensively, with results for counting and generation in both the normal and fixed content cases [9, 17].

This paper presents the first algorithm for ranking bracelets of length n over an alphabet of size k in polynomial time, with a time complexity of $O(k^2 \cdot n^4)$. This algorithm is further used to unrank bracelets in $O(n^5 \cdot k^2 \cdot \log(k))$. time. These polynomial time algorithms improve upon the exponential time brute-force algorithm.

2 Preliminaries

2.1 Definitions and Notation

Let Σ be a finite alphabet. We denote by Σ^* the set of all words over Σ and by Σ^n the set of all words of length n. For the remainder of this paper, let $k = |\Sigma|$. The notation $\bar{w}$ is used to clearly denote that the variable w is a word. The length of a word $\bar{u} \in \Sigma^*$ is denoted $|\bar{u}|$. We use $\bar{u}_i$, for any $i \in \{1 \dots |\bar{u}|\}$ to denote the i^{th} symbol of $\bar{u}$. The *reversal* operation on a word $\bar{w} = \bar{w}_1\bar{w}_2 \dots \bar{w}_n$, denoted by $\bar{w}^R$, returns the word $\bar{w}_n \dots \bar{w}_2\bar{w}_1$.

In the present paper we assume that Σ is linearly ordered. Let $[n]$ return the ordered set of integers from 1 to n inclusive. Given two words $\bar{u}, \bar{v} \in \Sigma^*$ where $|\bar{u}| \leq |\bar{v}|$, $\bar{u} = \bar{v}$ if and only if $|\bar{u}| = |\bar{v}|$ and $\bar{u}_i = \bar{v}_i$ for every $i \in [|\bar{u}|]$. A word $\bar{u}$ is *lexicographically smaller* than $\bar{v}$ if there exists an $i \in [|\bar{u}|]$ such that $\bar{u}_1\bar{u}_2 \dots \bar{u}_{i-1} = \bar{v}_1\bar{v}_2 \dots \bar{v}_{i-1}$ and $\bar{u}_i < \bar{v}_i$. For example, given the alphabet $\Sigma = \{a, b\}$ where $a < b$, the word *aaaba* is smaller than *aabaa* as the first two symbols are the same and a is smaller than b. For a given set of words $\mathbf{S}$, the rank of $\bar{v}$ with respect to $\mathbf{S}$ is the number of words in $\mathbf{S}$ that are smaller than $\bar{v}$.

The *rotation* of a word $\bar{w} = \bar{w}_1\bar{w}_2\dots\bar{w}_n$ by $r \in [n-1]$ returns the word $\bar{w}_{r+1}\dots\bar{w}_n\bar{w}_1\dots\bar{w}_r$, and is denoted by $\langle\bar{w}\rangle_r$, i.e. $\langle\bar{w}_1\bar{w}_2\dots\bar{w}_n\rangle_r = \bar{w}_{r+1}\dots\bar{w}_n\bar{w}_1\dots\bar{w}_r$. Under the rotation operation, $\bar{u}$ is equivalent to $\bar{v}$ if $\bar{v} = \langle\bar{w}\rangle_r$ for some r. The t^{th} power of a word $\bar{w} = \bar{w}_1\dots\bar{w}_n$, denoted $\bar{w}^t$, is equal to $\bar{w}$ repeated t times. For example $(aab)^3 = aabaabaab$. A word $\bar{w}$ is *periodic* if there is some word $\bar{u}$ and integer $t \geq 2$ such that $\bar{u}^t = \bar{w}$. Equivalently, word $\bar{w}$ is *periodic* if there exists some rotation $0 < r < |\bar{w}|$ where $\bar{w} = \langle\bar{w}\rangle_r$. A word is *aperiodic* if it is not periodic. The *period* of a word $\bar{w}$ is the length of the smallest word $\bar{u}$ for which there exists some value t for which $\bar{w} = \bar{u}^t$.

A *cyclic word*, also called a *necklace*, is the equivalence class of words under the rotation operation. For notation, a word $\bar{w}$ is written as $\tilde{\mathbf{w}}$ when treated as a necklace. Given a necklace $\tilde{\mathbf{w}}$, the *necklace representative* is the lexicographically smallest element of the set of words in the equivalence class $\tilde{\mathbf{w}}$. The necklace representative of $\tilde{\mathbf{w}}$ is denoted $\langle\tilde{\mathbf{w}}\rangle$, and the r^{th} shift of the necklace representative is denoted $\langle\tilde{\mathbf{w}}\rangle_r$. The reversal operation on a necklace $\tilde{\mathbf{w}}$ returns the necklace $\tilde{\mathbf{w}}^R$ containing the reversal of every word $\bar{u} \in \tilde{\mathbf{w}}$, i.e. $\tilde{\mathbf{w}}^R = \{\bar{u}^R : \bar{u} \in \tilde{\mathbf{w}}\}$. Given a word $\bar{w}$, $\langle\bar{w}\rangle$ will denote the necklace representative of the necklace containing $\bar{w}$, i.e. the representative of $\tilde{\mathbf{u}}$ where $\bar{w} \in \tilde{\mathbf{u}}$.

A *subword* of the cyclic word $\bar{w}$, denoted $\bar{w}_{[i,j]}$ is the word $\bar{u}$ of length $|\bar{w}|+j-i-1 \bmod |\bar{w}||)$ such that $\bar{u}_a = \bar{w}_{i-1+a \bmod |\bar{w}|}$. For notation $\bar{u} \sqsubseteq \bar{w}$ denotes that $\bar{u}$ is a subword of $\bar{w}$. Further, $\bar{u} \sqsubseteq_i \bar{w}$ denotes that $\bar{u}$ is a subword of $\bar{w}$ of length i. If $\bar{w} = \bar{u}\bar{v}$, then $\bar{u}$ is a prefix and $\bar{v}$ is a suffix. A prefix or suffix of a word $\bar{u}$ is *proper* if its length is smaller than $|\bar{u}|$. For notation, the tuple $\mathbf{S}(\bar{v},\ell)$ is defined as the set of all subwords of $\bar{v}$ of length ℓ. Formally let $\mathbf{S}(\bar{v},\ell) = \{\bar{s} \sqsubseteq \bar{v} : |\bar{s}| = \ell\}$. Further, $\mathbf{S}(\bar{v},\ell)$ is assumed to be in lexicographic order, i.e. $\mathbf{S}(\bar{v},\ell)_1 \geq \mathbf{S}(\bar{v},\ell)_2 \geq \dots \mathbf{S}(\bar{v},\ell)_{|\bar{v}|}$.

A *bracelet* is the equivalence class of words under the combination of the rotation and the reversal operations. In this way a bracelet can be thought of as the union of two necklace classes $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{w}}^R$, hence $\hat{\mathbf{w}} = \tilde{\mathbf{w}} \cup \tilde{\mathbf{w}}^R$. Given a bracelet $\hat{\mathbf{w}}$, the *bracelet representative* of $\hat{\mathbf{w}}$, denoted by $[\hat{\mathbf{w}}]$, is the lexicographically smallest word $\bar{u} \in \hat{\mathbf{w}}$.

A necklace $\tilde{\mathbf{w}}$ is *palindromic* if $\tilde{\mathbf{w}} = \tilde{\mathbf{w}}^R$. This means that the reflection of every word in $\tilde{\mathbf{w}}$ is in $\tilde{\mathbf{w}}^R$, i.e. given $\bar{u} \in \tilde{\mathbf{w}}, \bar{u}^R \in \tilde{\mathbf{w}}^R$. Note that for any word $\bar{w} \in \tilde{\mathbf{a}}$, where $\tilde{\mathbf{a}}$ is a palindromic necklace, either $\bar{w} = \bar{w}^R$, or there exists some rotation i for which $\langle\bar{w}\rangle_i = \bar{w}^R$.

Let $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ be a pair of necklaces belonging to the same bracelet class. For simplicity assume that $\langle\tilde{\mathbf{u}}\rangle < \langle\tilde{\mathbf{v}}\rangle$. The bracelet $\hat{\mathbf{u}}$ *encloses* a word $\bar{w}$ if $\langle\tilde{\mathbf{u}}\rangle < \bar{w} < \langle\tilde{\mathbf{v}}\rangle$. An example of this is the bracelet $\hat{\mathbf{u}} = aabc$ which encloses the word $\bar{w} = aaca$ as $aabc < aaca < aacb$. The set of all bracelets which enclose $\bar{w}$ are referred to as the set of bracelets *enclosing* $\bar{w}$.

2.2 Bounding Subwords

For both the palindromic and enclosing cases the number of necklaces smaller than $\bar{v} \in \Sigma^n$ is computed by iteratively counting the number of words of length n for which no subword is smaller than $\bar{v}$. The set of such words, denoted by $\mathbf{S}_n$, will be analysed iteratively as well, since it can have an exponential size. In order to relate $\mathbf{S}_i$ to $\mathbf{S}_{i+1}$, we will split $\mathbf{S}_i$ into parts using the positions of length i subwords of $\bar{v}$ with respect to the lexicographic order on $\mathbf{S}_i$. Informally, every $\bar{w} \in \mathbf{S}_i$ can be associated with the unique lower bound from $\mathbf{S}(\bar{v},i)$, which will be used to identify the parts leading us to the following definition.

▶ **Definition 1.** *Let $\bar{w},\bar{v} \in \Sigma^*$ where $|\bar{w}| \leq |\bar{v}|$. The word $\bar{w}$ is* bounded *(resp. strictly bounded) by $\bar{s} \sqsubseteq_{|\bar{w}|} \bar{v}$, if $\bar{s} \leq \bar{w}$ (resp. $\bar{s} < \bar{w}$) and there is no $\bar{u} \sqsubseteq_{|\bar{w}|} \bar{v}$ such that $\bar{s} < \bar{u} \leq \bar{w}$.*

The aforementioned parts $\mathbf{S}_i(\bar{s})$ contain all words $\bar{w} \in \mathbf{S}_i$ such that $\bar{s} \sqsubseteq_{|\bar{w}|} \bar{v}$. The key observation is that words of the form $x\bar{w}$ for all $\bar{w} \in \mathbf{S}_i$ and some fixed symbol $x \in \Sigma$ belong to the same set $\mathbf{S}_{i+1}(\bar{s}')$, where $\bar{s}' \sqsubseteq \bar{v}$. The same holds true for words of the form $\bar{w}x$.

Thus, we can compute the corresponding $\bar{s}'$ for all pairs of $\bar{s}$ and x in order to derive sizes of $\mathbf{S}_{i+1}(\bar{s}')$. Moreover, this relation between $\bar{s}$, x and $\bar{s}'$ is independent of i allowing us to store this information in two $n^2 \times k$ arrays XW and WX. Both arrays will be indexed by the words $\bar{s} \sqsubseteq \bar{v}$ and characters $x \in \Sigma$. Given a word $\bar{w}$ strictly bounded by $\bar{s}$, $XW[\bar{s}, x]$ will contain the word $\bar{s}' \sqsubseteq_{|\bar{s}|+1} \bar{v}$ strictly bounding $x\bar{w}$. Similarly, $WX[\bar{s}, x]$ will contain the word $\bar{s}' \sqsubseteq_{|\bar{s}|+1} \bar{v}$ strictly bounding $\bar{w}x$. By precomputing these arrays, the cost of determining these words can be avoided during the ranking process.

▶ **Proposition 2.** *Let $\bar{v} \in \Sigma^n$. The array $XW[\bar{s} \sqsubseteq \bar{v}, x \in \Sigma]$ such that $XW[\bar{s}, x]$ strictly bounds $x\bar{w}$ for every $\bar{w}$ strictly bounded by $\bar{s}$ can be computed in time $O(k \cdot n^3 \cdot \log(n))$.*

▶ **Proposition 3.** *Let $\bar{v} \in \Sigma^n$. The array $WX[\bar{s} \sqsubseteq \bar{v}, x \in \Sigma]$, such that $WX[\bar{s}, x]$ strictly bounds $\bar{w}x$ for every $\bar{w}$ strictly bounded by $\bar{s}$, can be computed in $O(k \cdot n^3 \cdot \log(n))$ time.*

3 Ranking Bracelets

The main result of the paper is the first algorithm for ranking bracelets. In this paper, we tacitly assume that we are ranking a word $\bar{v}$ of length n. The time-complexity of the ranking algorithm is $O(k^2 \cdot n^4)$, where k is the size of the alphabet and n is the length of the considered bracelets. The key part of the algorithm is to compute the rank of the word $\bar{v}$ with respect to the set of bracelets by finding three other ranks: the rank over all necklaces, the rank over palindromic necklaces, and the rank over enclosing apalindromic necklaces.

A bracelet can correspond to two apalindromic necklaces, or to exactly one palindromic necklace. If a bracelet $\hat{\mathbf{b}}$ corresponds to two necklaces $\tilde{\mathbf{l}}_b$ and $\tilde{\mathbf{r}}_b$, then it is important to take into account the lexicographical positions of these two necklaces $\tilde{\mathbf{l}}_b$ and $\tilde{\mathbf{r}}_b$ with respect to a given word $\bar{v}$. There are three possibilities: $\tilde{\mathbf{l}}_b$ and $\tilde{\mathbf{r}}_b$ could be less than $\bar{v}$; $\tilde{\mathbf{l}}_b$ and $\tilde{\mathbf{r}}_b$ encloses $\bar{v}$, e.g. $\tilde{\mathbf{l}}_b < \bar{v} < \tilde{\mathbf{r}}_b$, or both of necklaces $\tilde{\mathbf{l}}_b$ and $\tilde{\mathbf{r}}_b$ are greater than $\bar{v}$. This is visualised in Figure 2. Therefore the number of bracelets smaller than a given word w can be calculated by adding the number of palindromic necklaces less than $\bar{v}$, enclosing bracelets smaller than $\bar{v}$ and half of all other apalindromic and non-enclosing necklaces smaller than $\bar{v}$. Let us define the following notation is used for the rank of $\bar{v} \in \Sigma^n$ for sets of bracelets and necklaces.

- $RN(\bar{v})$ denotes the rank of $\bar{v}$ with respect to the set of *necklaces* of length n over Σ.
- $RP(\bar{v})$ denotes the rank of $\bar{v}$ with respect to the set of *palindromic necklaces* over Σ.
- $RB(\bar{v})$ denotes the rank of $\bar{v}$ with respect to the set of *bracelets* of length n over Σ.
- $RE(\bar{v})$ denotes the rank of $\bar{v}$ with respect to the set of *bracelets enclosing* $\bar{v}$.

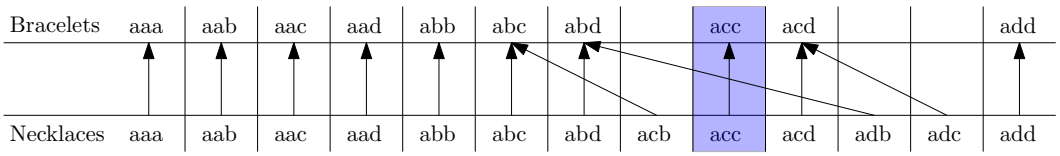

Figure 2 In this example the top line represents the set of bracelets and the bottom line the set of necklaces, with arrows indicated which necklace corresponds to which bracelet. Assuming we wish to rank the word *acc* (highlighted), *abc* and *acb* are apalindromic necklaces smaller than *acc*, while *abd* encloses *acc*. All other necklaces are palindromic.

In Lemma 4 below, we show that $RB(\bar{v})$ can be expressed via $RN(\bar{v})$, $RP(\bar{v})$ and $RE(\bar{v})$. The problem of computing $RN(\bar{v})$ has been solved in quadratic time [16], so the goal of the paper is to design efficient procedures for computing $RP(\bar{v})$ and $RE(\bar{v})$.

▶ **Lemma 4.** *The rank of a word $\bar{v} \in \Sigma^n$ with respect to the set of bracelets of length n over the alphabet Σ is given by $RB(\bar{v}) = \frac{1}{2}(RN(\bar{v}) + RP(\bar{v}) + RE(\bar{v}))$.*

Proof. Simply dividing the number of necklaces by 2 will undercount the number of bracelets, while doing nothing will overcount. Therefore to get the correct number of bracelets, those bracelets corresponding to only 1 necklace must be accounted for. A bracelet $\hat{\mathbf{a}}$ will correspond to 2 necklaces smaller than $\bar{v}$ if and only if $\hat{\mathbf{a}}$ does not enclose $\bar{v}$ and $\hat{\mathbf{a}}$ is apalindromic. Therefore the number of bracelets corresponding to 2 necklaces is $\frac{1}{2}(RN(\bar{v}) - RP(\bar{v}) - RE(\bar{v}))$. The number of bracelets enclosing $\bar{v}$ is equal to $RE(\bar{v})$. The number of bracelets corresponding to palindromic necklaces is equal to $RP(\bar{v})$. Therefore the total number of bracelets is $\frac{1}{2}(RN(\bar{v}) - RP(\bar{v}) - RE(\bar{v})) + RP(\bar{v}) + RE(\bar{v}) = \frac{1}{2}(RN(\bar{v}) + RP(\bar{v}) + RE(\bar{v}))$. ◀

Lemma 4 provides the basis for ranking bracelets. Theorem 5 uses Lemma 4 to get the complexity of the ranking process. The remainder of this paper will prove Theorem 5, starting with the complexity of ranking among palindromic necklaces in Section 4 followed by the complexity of ranking enclosing bracelets in Section 5.

▶ **Theorem 5.** *Given a word $\bar{v} \in \Sigma^n$, the rank of $\bar{v}$ with respect to the set of bracelets of length n over the alphabet Σ, $RB(\bar{v})$, can be computed in $O(k \cdot n^4)$ time.*

The remainder of this paper will prove Theorem 5. For simplicity, the word $\bar{v}$ is assumed to be a necklace representation. It is well established how to find the lexicographically largest necklace smaller than or equal to some given word. Such a word can be found in quadratic time using an algorithm form [16]. Note that the number of necklaces less than or equal to $\bar{v}$ corresponds to the number of necklaces less than or equal to the lexicographically largest necklace smaller than $\bar{v}$. From Lemma 4 it follows that to rank $\bar{v}$ with respect to the set of bracelets, it is sufficient to rank $\bar{v}$ with respect to the set of necklaces, palindromic necklaces, and enclosing bracelets. The rank with respect to the set of palindromic necklaces, $RP(\bar{v})$ can be computed in $O(k \cdot n^3)$ using the techniques given in Theorem 25 in Section 4. The rank with respect to the set of enclosing bracelets, $RE(\bar{v})$ can be computed in $O(k \cdot n^4)$ as shown in Theorem 30 in Section 5. As each of these steps can be done independently of each other, the total complexity is $O(k \cdot n^4)$.

This complexity bound is a significant improvement over the naive method of enumerating all bracelets, requiring exponential time in the worst case. New intuition is provided to rank the palindromic and enclosing cases. The main source of complexity for the problem of ranking comes from having to consider the lexicographic order of the word under reflection. New combinatorial results and algorithms are needed to count the bracelets in these cases.

Before showing in detail the algorithmic results that allow bracelets to be efficiently ranked, it is useful to discus the high level ideas. Lemma 4 shows our approach to ranking bracelets by dividing the problem into the problems of ranking necklaces, palindromic necklaces and enclosing bracelets. For both palindromic necklaces and enclosing bracelets, we derive a *canonical form* using the combinatorial properties of these objects.

Using these canonical forms, the number of necklaces smaller than $\bar{v}$ is counted in an iterative manner. In the palindromic case, this is done by counting the number of necklaces greater than $\bar{v}$, and subtracting this from the total number of palindromic necklaces. In the enclosing case, this is done by directly counting the number of necklaces smaller than $\bar{v}$. For both cases, the counting is done by way of a tree comprised of the set of all prefixes of words of the canonical form. By partitioning the internal vertices of the trees based on the number of children of the vertices, the number of words of the canonical form may be derived in an efficient manner, forgoing the need to explicitly generate the tree. This allows the size of these partitions to be computed through a dynamic programming approach. It follows from these partitions how to count the number of leaf nodes, corresponding to the canonical form.

▶ **Theorem 6.** *The z^{th} bracelet of length n over Σ can be computed in $O(n^5 \cdot k^2 \cdot \log(k))$.*

Theorem 6 is proven by using the ranking technique as a black box alongside a simple binary search in the same manner as [16].

4 Computing the rank $RP(\bar{v})$

To rank palindromic necklaces, it is crucial to analyse their combinatorial properties. This section focuses on providing results on determining unique words representing palindromic necklaces. We study two cases depending on whether the length n of a palindromic necklace is even or odd. The reason for this division can be seen by considering examples of palindromic necklaces. If equivalence under the rotation operation is not taken into account, then a word is palindromic if $\bar{w} = \bar{w}^R$. If the length n of $\bar{w}$ is odd, then if $\bar{w} = \bar{w}^R$, $\bar{w}$ can be written as $\bar{\phi}x\bar{\phi}^R$, where $\bar{\phi} \in \Sigma^{(n-1)/2}$ and $x \in \Sigma$. For example, the word *aaabaaa* is equal to $\bar{\phi}x\bar{\phi}^R$, where $\bar{\phi} = aaa$ and $x = b$. If the length n of w is even, then if $\bar{w} = \bar{w}^R$, $\bar{w}$ can be written as $\bar{\psi}\bar{\psi}^R$, where $\bar{\psi} \in \Sigma^{n/2}$. For example the word *aabbaa* is equal to $\bar{\psi}\bar{\psi}^R$, where $\bar{\psi} = aab$.

Once rotations are taken into account, the characterisation of palindromic necklaces becomes more difficult. It is clear that any necklace $\tilde{\mathbf{a}}$ that contains a word of the form $\bar{\phi}x\bar{\phi}^R$ or $\bar{\phi}\bar{\phi}^R$ is palindromic. However this check does not capture every palindromic necklace. Let us take, for example, the necklace $\tilde{\mathbf{a}} = ababab$, which contains two words *ababab* and *bababa*. While *ababab* can neither be written as $\bar{\phi}x\bar{\phi}^R$ nor $\bar{\phi}\bar{\phi}^R$, it is still palindromic as $\langle ababab^R \rangle = \langle bababa \rangle = ababab$. Therefore a more extensive test is required. As the structure of palindromic words without rotation is different depending on the length being either odd or even, it is reasonable to split the problem of determining the structure of palindromic necklaces into the cases of odd and even length.

The number of palindromic necklaces are counted by computing the number of these characterisations. This is done by constructing trees containing every prefix of these characterisations. As each vertex corresponds to the prefix of a word, the leaf nodes of these trees correspond to the words in the characterisations. By partitioning the tree in an intelligent manner, the number of leaf nodes and therefore number of these characterisations can be computed. In the odd case this corresponds directly to the number of palindromic necklaces, while in the even case a small transformation of these sets is needed.

4.1 Odd Length Palindromic Necklaces

Starting with the odd-length case, Proposition 7 shows that every palindromic necklace of odd length contains **exactly one word** that can be written as $\bar{\phi}x\bar{\phi}^R$ where $\bar{\phi} \in \Sigma^{(n-1)/2}$ and $x \in \Sigma$. This fact is used to rank the number of bracelets by constructing a tree representing every prefix of a word of the form $\bar{\phi}x\bar{\phi}^R$ that belongs to a bracelet greater than $\bar{v}$.

▶ **Proposition 7.** *A necklace $\tilde{\mathbf{w}}$ of odd length n is palindromic if and only if there exists exactly one word $\bar{u} = \bar{\phi}x\bar{\phi}^R$ such that $\bar{v} \in \tilde{\mathbf{w}}$, where $\bar{\phi} \in \Sigma^{(n-1)/2}$ and $x \in \Sigma$.*

Proof Sketch. If a necklace $\tilde{\mathbf{w}}$ contains a word of the form $\bar{\phi}x\bar{\phi}^R$, then clearly $\tilde{\mathbf{w}}$ is palindromic. The other direction follows a counting argument. In order for $\tilde{\mathbf{w}}$ to be palindromic, the reflection of every word in $\tilde{\mathbf{w}}$ must also be in $\tilde{\mathbf{w}}$. As $\tilde{\mathbf{w}}$ contains an odd number of words, there must exist a word that is its own reflection, i.e. $\bar{u} = \bar{u}^R$. As the length of the word is odd, the only time this can occur is when $\bar{u} = \bar{\phi}x\bar{\phi}^R$. We show uniqueness by observing that no two words in $\tilde{\mathbf{w}}$ can be mapped to the same palindromic word. ◀

▶ **Corollary 8.** *The number of palindromic necklaces of odd length n over Σ equals $k^{(n+1)/2}$.*

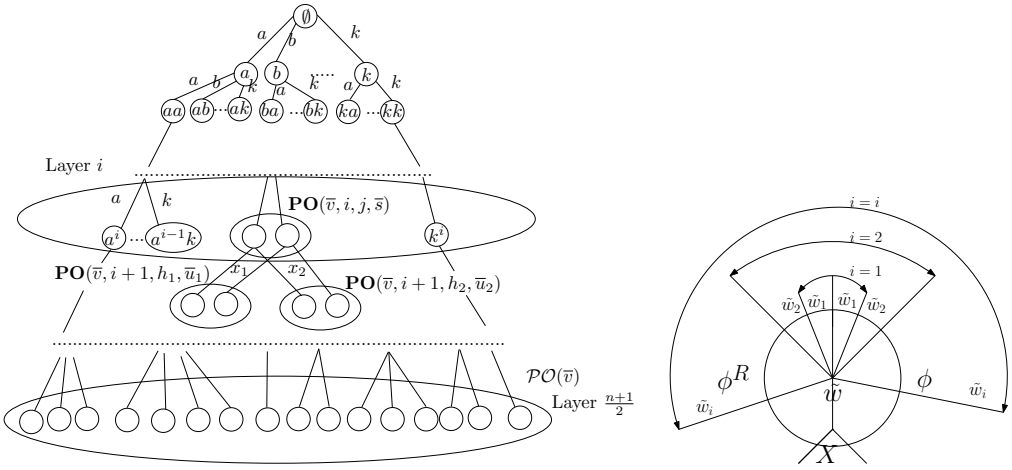

Figure 3 (Left) The relationship between $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ with the tree $\mathcal{TO}(\bar{v})$ and $\mathbf{PO}(\bar{v})$. (right) Example of the order for which characters are assigned. Note that at each step the choices for the symbol $\tilde{\mathbf{w}}_i$ is constrained in the no subword of $\tilde{\mathbf{w}}_{[1,i]}\tilde{\mathbf{w}}^R_{[1,i]}$ is greater than or equal to $\bar{v}$.

The problem now becomes to rank a word $\bar{v}$ with respect to the odd length palindromic necklaces utilising their combinatorial properties. Let $\bar{v} \in \Sigma^n$ be a word of odd length n. We define the set $\mathcal{PO}(\bar{v})$, where $\mathcal{PO}$ stands for palindromic odd length. The set $\mathcal{PO}(\bar{v})$ contains one word representing each palindromic bracelet of odd length n that is greater than $\bar{v}$.

$$\mathcal{PO}(\bar{v}) := \left\{ \bar{w} \in \Sigma^n : \bar{w} = \bar{\phi} x \bar{\phi}^R, \text{ where } \langle \bar{w} \rangle > \bar{v},\ \bar{\phi} \in \Sigma^{(n-1)/2},\ x \in \Sigma \right\}.$$

As each word will correspond to a unique palindromic necklace of length n greater than $\bar{v}$, and every palindromic necklace greater than $\bar{v}$ will correspond to a word in $\mathcal{PO}(\bar{v})$, the number of palindromic necklaces greater than $\bar{v}$ is equal to $|\mathcal{PO}(\bar{v})|$. Using this set the number of necklaces less than $\bar{v}$ can be counted by subtracting the size of $\mathcal{PO}(\bar{v})$ from the total number of odd length palindromic necklaces, equal to $k^{(n+1)/2}$ (Corollary 8).

High level idea for the Odd Case. Here we provide a high level idea for the approach we follow for computing $\mathcal{PO}(\bar{v})$. Let $\bar{v}$ have a length n. Since $\mathcal{PO}(\bar{v})$ only contains words of the form $\bar{\phi} x \bar{\phi}^R$, where $\bar{\phi} \in \Sigma^{(n-1)/2}$ and $x \in \Sigma$, we have that $\bar{w}_i = \bar{w}_{n-i}$ for every $\bar{w} \in \mathcal{PO}(\bar{v})$. As the lexicographically smallest rotation of every $\bar{w} \in \mathcal{PO}(\bar{v})$ must be greater than $\bar{v}$, it follows that any word rotation of $\bar{w}$ must be greater than $\bar{v}$ and therefore every subword of $\bar{w}$ must also be greater than or equal to the prefix of $\bar{v}$ of the same length. This property is used to compute the size of $\mathcal{PO}(\bar{v})$ by iteratively considering the set of prefixes of each word in $\mathcal{PO}(\bar{v})$ in increasing length representing them with the tree $\mathcal{TO}(\bar{v})$. As generating $\mathcal{TO}(\bar{v})$ directly would require an exponential number of operations, a more sophisticated approach is needed for the calculation of $|\mathcal{PO}(\bar{v})|$ based on partial information.

As the tree $\mathcal{TO}(\bar{v})$ is a tree of prefixes, vertices in $\mathcal{TO}(\bar{v})$ are referred to by the prefix they represent. So $\bar{u} \in \mathcal{TO}(\bar{v})$ refers to the unique vertex in $\mathcal{TO}(\bar{v})$ representing $\bar{u}$. The root vertex of $\mathcal{TO}(\bar{v})$ corresponds to the empty word. Every other vertex $\bar{u} \in \mathcal{TO}(\bar{v})$ corresponds to a word of length i, where i is the distance between $\bar{u}$ and the root vertex. Given two vertices $\bar{p}, \bar{c} \in \mathcal{TO}(\bar{v})$, $\bar{p}$ is the parent vertex of a child vertex $\bar{c}$ if and only if $\bar{c} = \bar{p}x$ for some symbol $x \in \Sigma$. The i^{th} layer of $\mathcal{TO}(\bar{v})$ refers to all representing words of length i in $\mathcal{TO}(\bar{v})$.

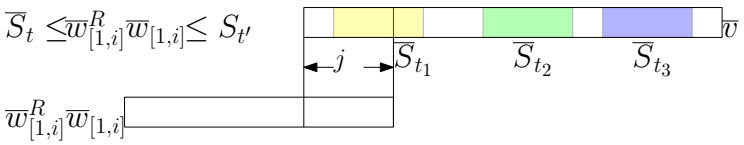

Figure 4 Visual representation of the properties of $\bar{w}^R_{[1,i]}\bar{w}_{[1,i]} \in \mathbf{PO}(\bar{v}, i, j, \bar{s})$.

The size of $\mathcal{PO}(\bar{v})$ is equivalent to the number of unique prefixes of length $\frac{n+1}{2}$ of words of the palindromic form $\bar{\phi}x\bar{\phi}^R$ in $\mathcal{PO}(\bar{v})$. This set of prefixes corresponds to the vertices in the layer $\frac{n+1}{2}$ of $\mathcal{TO}(\bar{v})$. Therefore the maximum depth of $\mathcal{TO}(\bar{v})$ is $\frac{n+1}{2}$.

To speed up computation, each layer of $\mathcal{TO}(\bar{v})$ is partitioned into sets that allow the size of $\mathcal{PO}(\bar{v})$ to be efficiently computed. This partition is chosen such that the size of the sets in layer $i+1$ can be easily derived from the size of the sets in layer i. As these sets are tied to the tree structure, the obvious property to use is the number of children each vertex has. As each vertex $\bar{u} \in \mathcal{TO}(\bar{v})$ represents a prefix of some word $\bar{w} \in \mathcal{PO}(\bar{v})$, the number of children of $\bar{u}$ is the number of symbols $x \in \Sigma$ such that $\bar{u}x$ is a prefix of some word in $\mathcal{PO}(\bar{v})$. Recall that every word in $\bar{w} \in \mathcal{PO}(\bar{v})$ has the form $\bar{\phi}x\bar{\phi}^R$, and that there is no subword of $\bar{w}$ that is less than $\bar{v}$. Therefore if $\bar{u} \in \mathcal{TO}(\bar{v})$, there must be no subword of $\bar{u}^R\bar{u}$ that is less than $\bar{v}$. Hence the number of children of $\bar{u}$ is the number of symbols $x \in \Sigma$ such that no subword of $x\bar{u}^R\bar{u}x$ is less than the prefix of $\bar{v}$ of the same length. As $\bar{u}^R\bar{u}$ has no subword less than $\bar{v}$, $x\bar{u}^R\bar{u}x$ will only have a subword that is less than $\bar{v}$ if either (1) $x\bar{u}^R\bar{u}x < \bar{v}$ or (2) there exists some suffix of length j such that $(\bar{u}^R\bar{u})_{[2i-j,2i]} = \bar{v}_{[1,j]}$ and $x < \bar{v}_{j+1}$. For the first condition, let $\bar{s} \sqsubseteq_{2i} \bar{v}$. By the definition of strictly bounding subwords (Definition 1), $x\bar{u}^R\bar{u}x < \bar{v}$ if and only if $x\bar{s}x < \bar{v}$. Note that this ignores any word $\bar{u}$ where $\bar{u}^R\bar{u} \sqsubseteq \bar{v}$. The restriction to strictly bounded words is to avoid the added complexity caused by Proposition 2, where the word that bounds $x\bar{s}x$ might not be the word that bounds $x\bar{u}^R\bar{u}x$. For the second property, let j be the length of the longest suffix of $\bar{u}^R\bar{u}$ that is a prefix of $\bar{v}$. From Lemma 1 due to Sawada and Williams [16], there is some suffix of $\bar{u}^R\bar{u}x$ that is smaller than $\bar{v}$ if and only if $x < \bar{v}_{j+1}$. The i^{th} layer of $\mathcal{TO}(\bar{v})$ is partitioned into n^2 sets $\mathbf{PO}(\bar{v}, i, j, \bar{s})$, for every $i \in [\frac{n+1}{2}], j \in [2i]$ and $\bar{s} \sqsubseteq_{2i} \bar{v}$.

▶ **Definition 9.** *Let $i \in [\frac{n+1}{2}], j \in [2i]$ and $\bar{s} \sqsubseteq_{2i} \bar{v}$. The set $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ contains every prefix $\bar{u} \in \mathcal{TO}(\bar{v})$ of length i where (1) the longest suffix of $\bar{u}^R_{[1,i]}\bar{u}_{[1,i]}$ which is a prefix of $\bar{v}$ has a length of j and (2) The word $\bar{u}^R_{[1,i]}\bar{u}_{[1,i]}$ is strictly bounded by $\bar{s}$.*

An overview of the properties used by $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ is given in Figures 3 and 4. It follows from the earlier observations that each vertex in $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ has the same number of children. Lemma 10 strengthens this observation, showing that given $\bar{a}, \bar{b} \in \mathbf{PO}(\bar{v}, i, j, \bar{s})$, $\bar{a}x \in \mathbf{PO}(\bar{v}, i+1, j', \bar{s}')$ if and only if $\bar{b}x \in \mathbf{PO}(\bar{v}, i+1, j', \bar{s}')$.

The remainder of this section establishes how to count the size of $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ and the number of children vertices for each vertex in $\mathbf{PO}(\bar{v}, i, j, \bar{s})$. The first step is to formally prove that all vertices in $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ have the same number of children vertices. This is shown in Lemma 10 by proving that given two vertices $\bar{a}, \bar{b} \in \mathbf{PO}(\bar{v}, i, j, \bar{s})$, if the vertex $\bar{a}' = \bar{a}x$ for $x \in \Sigma$ belongs to the set $\mathbf{PO}(\bar{v}, i+1, j', \bar{s}')$, so to does $\bar{b}' = \bar{b}x$.

▶ **Lemma 10.** *Let $\bar{a}, \bar{b} \in \mathbf{PO}(\bar{v}, i, j, \bar{s})$ and let $x \in \Sigma$. If the vertex $\bar{a}' = \bar{a}x$ belongs to $\mathbf{PO}(\bar{v}, i+1, j', \bar{s}')$, the vertex $\bar{b}' = \bar{b}x$ also belongs to $\mathbf{PO}(\bar{v}, i+1, j', \bar{s}')$. Furthermore the value of j' and $\bar{s}'$ can be computed in constant time from the values of $j, \bar{s}$ and x.*

Proof Sketch. The arrays XW and WX are used to determine the value of $\bar{s}'$. The longest suffix of both $x\bar{a}x$ and $x\bar{b}x$ that is a prefix of $\bar{v}$ is determined by the value of j and x, either $j+1$, if $x = \bar{v}_{j+1}$, or 0 otherwise. Hence if $\bar{a}x \in \mathbf{PO}(\bar{v}, i+1, j', \bar{s}')$, $\bar{b}x \in \mathbf{PO}(\bar{v}, i+1, j', \bar{s}')$. Further the values of j' and $\bar{s}'$ can be determined in constant time. ◀

Computing the size of $\mathbf{PO}(\bar{v}, i, j', \bar{s}')$. Lemma 10, provides enough information to compute the size of $\mathbf{PO}(\bar{v}, i, j', \bar{s}')$ once the size of $\mathbf{PO}(\bar{v}, i-1, j, \bar{s})$ has been computed for each value of $j \in [2(i-1)]$ and $\bar{s} \in \mathbf{S}(\bar{v}, 2(i-1))$. At a high level, the idea is to create an array, $SizePO$, storing the size of the $\mathbf{PO}(\bar{v}, i, j', \bar{s}')$ for every value of $i \in [\frac{n-1}{2}], j \in [2i]$ and $\bar{s} \sqsubseteq_{2i} \bar{v}$. For simplicity, let the value of $SizePO[i, j, \bar{s}]$ be the size of $|\mathbf{PO}(\bar{v}, i, j, \bar{s})|$.

Lemma 11 formally provides the method of computing $SizePO[i, j, \bar{s}]$ for every $j \in [2i]$ and $\bar{s} \sqsubseteq_{2i} \bar{v}$ once $SizePO[i-1, j', \bar{s}']$ has been computed for every $j' \in [2i-2]$ and $\bar{s} \sqsubseteq_{2i-2} \bar{v}$. Observe that each vertex $a \in \mathbf{PO}(\bar{v}, i, j', \bar{s}')$ represents a prefix $\bar{a}'x$ where $\bar{a}'$ is either in $\mathbf{PO}(\bar{v}, i-1, j, \bar{s})$, for some value of j and $\bar{s}$, or $\bar{a}' \sqsubseteq \bar{v}$. Using this, the high level idea is to derive the values of j' and $\bar{s}'$ for each $j \in [2(i-1)], \bar{s} \in \mathbf{S}(\bar{v}, 2(i-1))$ and $x \in \Sigma$. Once the values j' and $\bar{s}'$ have been derived, the value of $SizePO[i, j', \bar{s}']$ is increased by the size of $\mathbf{PO}(\bar{v}, i-1, j, \bar{s})$. Repeating this for every value of $j, \bar{s}$ and x will leave the value of $SizePO[i, j', \bar{s}']$ as the number of vertices in $\mathbf{PO}(\bar{v}, i, j', \bar{s}')$ representing words of the form $\bar{a}x$ where $\bar{a} \not\sqsubseteq \bar{v}$. As each set $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ may have children in at most k sets $\mathbf{PO}(\bar{v}, i+1, j', \bar{s}')$, the number of vertices in $\mathbf{PO}(\bar{v}, i+1, j', \bar{s}')$ with a parent vertex in $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ can be computed in $O(k \cdot n^2)$ by looking at every argument of $j \in [2i]$ and $\bar{s} \sqsubseteq_{2i} \bar{v}$.

To account for the vertices in $\mathbf{PO}(\bar{v}, i, j', \bar{s}')$ of the form $\bar{b}x$ where $\bar{b}^R\bar{b} \sqsubseteq \bar{v}$, a similar process is applied to each pair $\bar{s} \in \mathbf{S}(\bar{v}, 2(i-1))$ and $x \in \Sigma$. For each pair, the values $\bar{s}'$ and j' are derived in the same manner as Lemma 10 utilising the tables XW and WX. Once derived, the value of $SizePO[i, j', \bar{s}']$ is increased by one, to account for the vertex $\bar{s}x$. As the values of j' and $\bar{s}'$ can be computed in $O(n)$ time from the value of x and $\bar{s}$, the number of vertices in $\mathbf{PO}(\bar{v}, i+1, j', \bar{s}')$ where the parent vertex is a subword of $\bar{v}$ can be computed in $O(k \cdot n^2)$ time.

▶ **Lemma 11.** *Given the size of $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ for $i \in \left[\frac{n-3}{2}\right]$ and every $j \in [2i], \bar{s} \sqsubseteq_{2i} \bar{v}$, the size of $\mathbf{PO}(\bar{v}, i+1, j', \bar{s}')$ for every $j' \in [2i+2], \bar{s}' \sqsubseteq_{2i+2} \bar{v}$ can be computed in $O(k \cdot n^2)$ time.*

Proof Sketch. $SizePO[i+1, j', \bar{s}']$ is computed by looking at every argument of $j \in [2i], \bar{s} \sqsubseteq_{2i} \bar{v}$ and $x \in \Sigma$. For each set of arguments, $\bar{j}'$ and $\bar{s}'$ are derived by Lemma 10, and the size of $SizePO[i+1, j', \bar{s}']$ is increased by $SizePO[i, j, \bar{s}]$. Similarly, for each $x \in \Sigma$ and $\bar{s} \sqsubseteq_{2i} \bar{v}$, $\bar{j}'$ and $\bar{s}'$ are derived and the size of $SizePO[i+1, j', \bar{s}']$ is increased by 1. ◀

Once the size of $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ has been computed for every $i \in [\frac{n-1}{2}], j \in [2i], \bar{s} \in \mathbf{S}(\bar{v}, 2i)$, the final step is to compute $|\mathcal{PO}(\bar{v})|$. The high level idea is to determine the number of vertices in $\mathcal{PO}(\bar{v})$ are children of a vertex in $\mathbf{PO}(\bar{v}, \frac{n-1}{2}, j, \bar{s})$.The set $\mathbf{X}(\bar{v}, j, \bar{s}) \subseteq \Sigma$ is introduced to help with this goal. Let $\mathbf{X}(\bar{v}, j, \bar{s})$ contain every symbol $x \in \Sigma$ such that $\bar{a}x\bar{a}^R \in \mathcal{PO}(v)$ where $\bar{a} \in \mathbf{PO}(\bar{v}, \frac{n-1}{2}, j, \bar{s})$. By the definition of $\mathbf{X}(\bar{v}, j, \bar{s})$, $|\mathbf{X}(\bar{v}, j, \bar{s})| \cdot |\mathbf{PO}(\bar{v}, \frac{n-1}{2}, j, \bar{s})|$ equals the number of words $\bar{w} \in \mathcal{PO}(\bar{v})$ where $(\bar{w}_1 \dots \bar{w}_{(n-1)/2}) \in \mathbf{PO}(\bar{v}, i, j, \bar{s})$. Lemma 12 shows how to compute the size of $\mathbf{X}(\bar{v}, j, \bar{s})$ in $O(k \cdot n)$ time.

▶ **Lemma 12.** *Let $\mathbf{X}(\bar{v}, j, \bar{s})$ contain every symbol in Σ such that $\bar{a}x\bar{a}^R \in \mathcal{PO}(v)$ where $\bar{a} \in \mathbf{PO}(\bar{v}, \frac{n-1}{2}, j, \bar{s})$. The size of $\mathbf{X}(\bar{v}, j, \bar{s})$ can be computed in $O(k \cdot n)$ time.*

Proof Sketch. The size of $\mathbf{X}(\bar{v}, j, \bar{s})$ is computed by checking if $x \in \mathbf{X}(\bar{v}, j, \bar{s})$ for each $x \in \Sigma$, where j and $\bar{s}$ are used to bound x from below. As x must be such that $\bar{v}_{[1,j]}x\bar{s} > \bar{v}$, $x \geq \bar{v}_{j+1}$. Further, if $\bar{s} < \bar{v}_{[j+2,n]}$ then $x > \bar{v}_{j+1}$. ◀

Converting *SizePO* to $|\mathcal{PO}(\bar{v})|$. The final step in computing $\mathcal{PO}(\bar{v})$ is to convert the cardinality of $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ to the size of $\mathcal{PO}(\bar{v})$. Lemma 13 provides a formula for counting the size of $\mathcal{PO}(\bar{v})$. Combining this formula with the techniques given in Lemma 11 an algorithm for computing the size of $\mathcal{PO}(\bar{v})$ directly follows.

It follows from Lemma 10 that the number of words in $\mathcal{PO}(\bar{v})$ with a prefix in $\mathbf{PO}\left(\bar{v}, \frac{n-1}{2}, j, \bar{s}\right)$ is equal to the cardinality of $\mathbf{PO}\left(\bar{v}, \frac{n-1}{2}, j, \bar{s}\right)$ multiplied by the size of $\mathbf{X}(\bar{v}, j, \bar{s})$. Similarly the number of words in $\mathcal{PO}(\bar{v})$ with a prefix $\bar{u}$ of length $\frac{n-1}{2}$ where $\bar{u}^R\bar{u} \sqsubseteq \bar{v}$ can be determined using $\mathbf{X}(\bar{v}, j, \bar{u}^R\bar{u})$. The main difference in this case is that if $\bar{u}^R\bar{u} = \bar{v}_{[j+2,n+j]}$, where j is the length of the longest suffix of $\bar{u}^R\bar{u}$ that is a prefix of $\bar{v}$, then the number of words in $\mathcal{PO}(\bar{v})$ where $\bar{u}$ is a prefix is 1 fewer than for the number of words strictly bounded by $\bar{u}^R\bar{u}$, i.e. $|\mathbf{X}(\bar{v}, J(\bar{s}, \bar{v}), \bar{s})| - 1$. Lemma 13 provides the procedure to compute $|\mathcal{PO}(\bar{v})|$.

▶ **Lemma 13.** *Let $J(\bar{s}, \bar{v})$ return the length of the longest suffix of $\bar{s}$ that is a prefix of $\bar{v}$. The size of $\mathcal{PO}(\bar{v})$ is equal to*

$$\sum_{\bar{s} \in \mathbf{S}(\bar{v}, n-1)} \left(\sum_{j=1}^{n-1} |\mathbf{X}(\bar{v}, j, \bar{s})| \cdot |\mathbf{PO}\left(\bar{v}, \tfrac{n-1}{2}, j, \bar{s}\right)| \right) + \begin{cases} 0 & \bar{s} \neq \phi\phi^R \\ |\mathbf{X}(\bar{v}, J(\bar{s}, \bar{v}), \bar{s})| & \bar{s} \neq \bar{v}_{[j+2,n+j]} \\ |\mathbf{X}(\bar{v}, J(\bar{s}, \bar{v}), \bar{s})| - 1 & \bar{s} = \bar{v}_{[j+2,n+j]} \end{cases}$$

Further this can be computed in $O(k \cdot n^3 \cdot \log(n))$ time.

Proof Sketch. By the definition of $\mathbf{X}(\bar{v}, j, \bar{s})$, the number of words in $\mathcal{PO}(\bar{v})$ with a parent vertex in $\mathbf{PO}(\bar{v}, \frac{n-1}{2}, j, \bar{s})$ equals $|\mathbf{X}(\bar{v}, j, \bar{s})|$. Similarly, given $\bar{s} \sqsubseteq_{n-1} \bar{v}$, the number words in $\mathcal{PO}(\bar{v})$ of the form $\bar{s}_{[(n+1)/2,n-1]}x\bar{s}_{[1,(n-1)/2]}$ are equal to either $|\mathbf{X}(\bar{v}, j, \bar{s})|$, if $\bar{s} \neq \bar{v}_{[j+2,n+j]}$, or $|\mathbf{X}(\bar{v}, j, \bar{s})| - 1$ if $\bar{s} = \bar{v}_{[j+2,n+j]}$. ◀

4.2 Even Length Palindromic Necklaces

Section 4.1 shows how to rank $\bar{v}$ within the set of odd length palindromic necklaces. This leaves the problem of counting even length palindromic necklaces. As in the odd case, the first step is to determine how to characterise these words. Proposition 14 shows that every palindromic necklace will have at least one word of either the form $\bar{\phi}\bar{\phi}^R$, where $\bar{\phi} \in \Sigma^{n/2}$, or $x\bar{\phi}y\bar{\phi}^R$, where $x, y \in \Sigma$ and $\bar{\phi} \in \Sigma^{(n/2)-1}$. Proposition 14 is strengthened by Propositions 15 and 16, showing that each palindromic necklace of even length will have no more than two words of either form. Lemmas 21, 22, 23 and 24 use these results a similar manner to Section 4.1 to count the number of palindromic necklaces of even length.

▶ **Proposition 14.** *A necklace $\tilde{\mathbf{w}}$ of even length n is palindromic if and only if there exists some word $\bar{u} \in \tilde{\mathbf{w}}$ where either (1) $\bar{u} = x\bar{\phi}y\bar{\phi}^R$ where $x, y \in \Sigma$ and $\bar{\phi} \in \Sigma^{(n/2)-1}$, or (2) $\bar{u} = \bar{\phi}\bar{\phi}^R$ where $\bar{\phi} \in \Sigma^{n/2}$.*

Proof Sketch. Recall that if the necklace $\tilde{\mathbf{w}}$ is palindromic, then for any word $\bar{u} \in \tilde{\mathbf{w}}$, $\bar{u}^R \in \tilde{\mathbf{w}}$. As such, $\bar{u} = \langle \bar{u}^R \rangle_r$ for some rotation r. It follows from this that $\bar{u}_1 = \bar{u}_{n-r}, \bar{u}_2 = u_(n - r - 1) \ldots \bar{u}_j = \bar{u}_{n-r-j+1}$. The word $\bar{u}$ is split into the subwords $\bar{a} = \bar{u}_{[1,n-r]}$ and $\bar{b}_{[n-r+1,n]}$. As $\bar{a} = \bar{a}^R$ and $\bar{b} = \bar{b}^R$, the smaller subword can be lengthened by appending the first and last symbol of the longer word while maintaining this property, i.e. $\bar{b}' = \bar{a}_1\bar{b}\bar{a}_{|\bar{a}|} = \bar{b}'^R$. Repeating this until either both words have length $\frac{n}{2}$, or one has length $\frac{n}{2} - 1$, allows $\bar{u}$ to be rewritten, using these words to derive the values of $\bar{\phi} \in \Sigma^{n/2}$, or $x, y \in \Sigma$ and $\bar{\phi} \in \Sigma^{(n/2)-1}$. ◀

▶ **Proposition 15.** *The word $\bar{u} \in \Sigma^*$ equals both $x\bar{\phi}y\bar{\phi}^R = \bar{\psi}\bar{\psi}^R$ if and only if $\bar{u} = x^n$.*

▶ **Proposition 16.** *For an even length palindromic necklace* $\tilde{\mathbf{a}}$ *there are at most two words* $\bar{w}, \bar{u} \in \tilde{\mathbf{a}}$ *where either (1)* $\bar{w}$ *and* $\bar{u}$ *are of the form* $x\bar{\phi}y\bar{\phi}^R$ *where* $x, y \in \Sigma$ *and* $\bar{\phi} \in \Sigma^{(n/2)-1}$ *or (2)* $\bar{w}$ *and* $\bar{u}$ *are of the form* $\bar{\phi}\bar{\phi}^R$ *where* $\bar{\phi} \in \Sigma^{n/2}$.

Proof Sketch. For both $\bar{w} = x\bar{\phi}y\bar{\phi}^R$, and $\bar{w} = \bar{\phi}\bar{\phi}^R$, a similar approach is used. For $\bar{w} = x\bar{\phi}y\bar{\phi}^R$ assume that $\langle\bar{w}\rangle_r = \bar{u}$ for the smallest rotation r such that $\langle\bar{w}\rangle_r \neq \bar{w}$ and $\langle\bar{w}\rangle_r = x\bar{\phi}y\bar{\phi}^R$. By showing that the period of $\bar{w}$ must be $2r$, it follows that if there is some other rotation s such that $\bar{w} \neq \langle\bar{w}\rangle_s \neq \langle\bar{w}\rangle_r$ and $\langle\bar{w}\rangle_s$ is of the form $x\bar{\phi}y\bar{\phi}^R$, s must be less than r contradicting the initial assumption. For $\bar{w} = \bar{\phi}\bar{\phi}^R$, a similar process is done, showing that not only must there be no more than two words of the form $\bar{\phi}\bar{\phi}^R$, but that if $\bar{w} = \bar{\phi}\bar{\phi}^R$ then $\bar{u} = \bar{\psi}\bar{\psi}^R$ if and only if $\bar{\phi} = \bar{\psi}^R$. ◀

Propositions 14, 15 and 16 show that every palindromic necklace of even length has 1 or 2 words of either the form $x\bar{\phi}y\bar{\phi}^R$ or $\bar{\phi}\bar{\phi}^R$. To count the number of words of each form, the problem is split into two sub problems, counting words of the form $x\bar{\phi}y\bar{\phi}^R$ and counting the number of words of the form $\bar{\phi}\bar{\phi}^R$. This is done using the same basic ideas as in Section 4.1. Two new sets $\mathcal{PE}(\bar{v})$ and $\mathcal{PS}(\bar{v})$ are introduced, serving the same function as $\mathcal{PO}(\bar{v})$ for words of the from $x\bar{\phi}y\bar{\phi}^R$ and $\bar{\phi}\bar{\phi}^R$ respectively.

$$\mathcal{PE}(\bar{v}) := \left\{\bar{w} \in \Sigma^n : \bar{w} = x\bar{\phi}y\bar{\phi}^R, \text{ where } \langle\bar{w}\rangle > \bar{v}, \bar{\phi} \in \Sigma^{(n/2)-1}, x, y, \in \Sigma\right\}$$

$$\mathcal{PS}(\bar{v}) := \left\{\bar{w} \in \Sigma^n : \bar{w} = \bar{\phi}\bar{\phi}^R, \text{ where } \langle\bar{w}\rangle > \bar{v}, \bar{\phi} \in \Sigma^{(n/2)-1}\right\}$$

Unlike the set $\mathcal{PO}(\bar{v})$ in Section 4.1 the sets $\mathcal{PE}(\bar{v})$ and $\mathcal{PS}(\bar{v})$ do not correspond directly to bracelets greater than $\bar{v}$. For notation let $\mathcal{GE}(\bar{v})$ and $\mathcal{GS}(\bar{v})$ denote the number of bracelets greater than $\bar{v}$ of the form $x\bar{\phi}y\bar{\phi}^R$ and $\bar{\phi}\bar{\phi}^R$ respectively. The number of even length necklaces greater than $\bar{v}$ equals $\mathcal{GE}(\bar{v}) + \mathcal{GS}(\bar{v}) - (k - \bar{v}_1)$, where $k - \bar{v}_1$ denotes the number of symbols in Σ greater than $\bar{v}_1$. Before showing how to compute the size of these sets, it is useful to first understand how they are used to compute the rank amongst even length palindromic necklaces. Lemmas 19 and 18 shows how to covert the cardinalities of these sets into the number of even length palindromic necklaces smaller than $\bar{v}$. The main idea is to use the observations given by Propositions 14 and 16 to determine how many even length palindromic necklaces have either one or two words of the form $x\bar{\phi}y\bar{\phi}^R$ or $\bar{\phi}\bar{\phi}^R$.

▶ **Proposition 17.** *Let* $l = \frac{n+2}{4}$ *if* $\frac{n}{2}$ *is odd or* $l = \frac{n}{4}$ *if* $\frac{n}{2}$ *is even. The number of even length palindromic necklaces is given by* $\frac{1}{2}\left(k^{n/2}(k+2) + k^l\right) - k$.

Proof Sketch. This equation is derived by first determining the number of necklaces that has only one word of the form $x\bar{\phi}y\bar{\phi}^R$, from which the first $k^{n/2}$ term comes from. The number of necklaces with two representatives can be computed by subtracting the number of necklaces with one representative from the number of words of the form $x\bar{\phi}y\bar{\phi}^R$, giving $\frac{1}{2}(k^{n/2+1} - k^{n/2})$ By adding these two values together, the total number of necklaces of this form can be counted as $\frac{1}{2}(k^{n/2+1} - k^{n/2}) + k^{n/2} = \frac{1}{2}(k^{n/2+1} + k^{n/2})$. The number of necklaces with two words of the form $\bar{\phi}\bar{\phi}^R$ is counted by determining the number of necklaces with only one word of the form $\bar{\phi}\bar{\phi}^R$, giving the k^l term. Subtracting this from the number of words of the form $\bar{\phi}\bar{\phi}^R$, giving a total of $\frac{1}{2}(k^{n/2} - k^l) + k^l = \frac{1}{2}(k^{n/2} + k^l)$ necklaces. Finally, to avoid over counting words of the form x^n, the k necklaces of this form are subtracted, giving a total of $\frac{1}{2}\left(k^{n/2+1} + k^{n/2} + k^{n/2} + k^l\right) - k$. ◀

▶ **Lemma 18.** *The number of necklaces greater than* $\bar{v}$ *containing at least one word of the form* $x\bar{\phi}y\bar{\phi}^R$ *is given by* $GE(\bar{v}) = \frac{1}{2}\left(|\mathcal{PE}(\bar{v})| + \begin{cases} |\mathcal{PO}(\bar{v}_{[1,n/2]})| & \frac{n}{2} \text{ is odd.} \\ GE(\bar{v}_{[1,n/2]}) & \frac{n}{2} \text{ is even.} \end{cases}\right)$.

Proof Sketch. This claim is shown by dividing necklaces greater than $\bar{v}$ in to two sets, those with one word of the form $x\bar{\phi}y\bar{\phi}^R$, and those with two. By subtracting the set of necklaces with only a single such word from the total set, the number of necklaces with two such representatives are counted. The equation comes from adding the size of these sets. ◀

▶ **Lemma 19.** *The number of necklaces greater than $\bar{v}$ containing at least one word of the form $\bar{\phi}\bar{\phi}^R$ is given by $GS(\bar{v}) = \frac{1}{2}\left(|\mathcal{PE}(\bar{v})| + \begin{cases} |\mathcal{PO}(\bar{v})| & \frac{n}{2} \text{ is odd.} \\ GS(\bar{v}_{[1,n/2]}) & \frac{n}{2} \text{ is even.} \end{cases}\right)$.*

Proof Sketch. This claim is shown by dividing necklaces greater than $\bar{v}$ in to two sets, those with one word of the form $\bar{\phi}\bar{\phi}^R$, and those with two. By subtracting the set of necklaces with only a single such word from the total set, the number of necklaces with two such representatives can be counted. The final equation comes from adding the size of these sets together. ◀

High Level Idea for the Even Case. Lemmas 18 and 19 show how to use the sets $\mathcal{PS}(\bar{v})$ and $\mathcal{PE}(\bar{v})$ to get the number of necklaces of the form $x\bar{\phi}y\bar{\phi}^R$ and $\bar{\phi}\bar{\phi}^R$ respectively. This leaves the problem of computing the size of both sets. This is achieved in a manner similar to the one outlined in Section 4.1. At a high level the idea is to use two trees analogous to $\mathcal{TO}(\bar{v})$ as defined in Section 4.1. The tree $\mathcal{TE}(\bar{v})$ is introduced to compute the cardinality of $\mathcal{PE}(\bar{v})$ and the tree $\mathcal{TS}(\bar{v})$ is introduced to compute the cardinality of $\mathcal{PS}(\bar{v})$. As in Section 4.1, the trees $\mathcal{TE}(\bar{v})$ and $\mathcal{TS}(\bar{v})$ contain every prefix of a word in $\mathcal{PS}(\bar{v})$ or $\mathcal{PE}(\bar{v})$ respectively. The leaf vertices of these trees correspond to the words in these sets.

To compute the size of $\mathcal{PE}(\bar{v})$ using $\mathcal{TE}(\bar{v})$, the same approach as in Section 4.1 is used. A word $\bar{u}$ of length less than $\frac{n}{2}$ is a prefix of some word in $\mathcal{PE}(\bar{v})$ if and only if no subword of $(\bar{u}_{[1,|\bar{u}|-1]})^R\bar{u}$ is less than the prefix of $\bar{v}$ of the same length. This is slightly different from the odd case, where $\bar{u} \in \mathcal{PE}(\bar{v})$ if and only if there is no subword of $\bar{u}^R\bar{u}$ smaller than the prefix of $\bar{v}$ of the same length. To account for this difference the sets $\mathbf{PE}(\bar{v}, i, j, \bar{s})$ are introduced as analogies to the sets $\mathbf{PO}(\bar{v}, i, j, \bar{s})$.

▶ **Definition 20.** *Let $i \in [\frac{n+1}{2}], j \in [2i]$ and $\bar{s} \sqsubseteq_{2i} \bar{v}$. The set $\mathbf{PE}(\bar{v}, i, j, \bar{s})$ contains every word $\bar{u} \in \mathcal{TE}(\bar{v})$ of length i where (1) the longest suffix of $(\bar{u}_{[1,i-1]})^R\bar{u}_{[1,i]}$ which is a prefix of $\bar{v}$ has a length of j and (2) the word $(\bar{u}_{[1,i-1]})^R\bar{u}_{[1,i]}$ is strictly bounded by $\bar{s} \sqsubseteq_{2i-1} \bar{v}$.*

As in Section 4.1, the size of $\mathbf{PE}(\bar{v}, i, j, \bar{s})$ is computed via dynamic programming. The array $SizePE$ is introduced, storing the size of $\mathbf{PE}(\bar{v}, i, j, \bar{s})$ for every value of $i \in \left[\frac{n}{2}\right], j \in [2i-1]$ and $\bar{s} \sqsubseteq_{2i-1} \bar{v}$. Let $SizePE$ be and $n \times n \times n$ array such that $SizePE[i, j, \bar{s}] = |\mathbf{PE}(\bar{v}, i, j, \bar{s})|$. Lemma 21 shows that the techniques used in Lemma 11 can be used to compute $SizePE$ in $O(k \cdot n^3 \log(n))$ time. This is done by proving that the properties established by Lemma 10 regarding the relationship between the sets $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ also hold for the sets $\mathbf{PE}(\bar{v}, i, j, \bar{s})$. As words in $\mathcal{PS}(\bar{v})$ are of the form $\bar{\phi}\bar{\phi}^R$, a word $\bar{u}$ is in $\mathcal{TS}(\bar{v})$ if and only if no subword of $\bar{u}^R\bar{u}$ is less than the prefix of $\bar{v}$ of the same length. Note that this corresponds to the same requirement as the odd case. As such the internal vertices in the tree $\mathcal{TS}(\bar{v})$ may be partitioned in the same way as those of $\mathcal{TO}(\bar{v})$. Lemma 23 shows how to convert the array $SizePO$ as defined is Section 4.1 to the size of $\mathcal{PS}(\bar{v})$.

▶ **Lemma 21.** *Given $\bar{u}, \bar{w} \in \mathbf{PE}(\bar{v}, i, j, \bar{s})$ and $x \in \Sigma$. If $\bar{u}x \in \mathbf{PE}(\bar{v}, i+1, j', \bar{s}')$ then $\bar{v}x \in \mathbf{PE}(\bar{v}, i+1, j', \bar{s}')$. Further the values of j' and $\bar{s}'$ can be computed in constant time from the values of $j, \bar{s}$ and x. Therefore the array $SizePE[i, j, \bar{s}]$ can be computed for every value $i \in \left[\frac{n}{2}\right], j \in [2i-1]$ and $\bar{s} \sqsubseteq_{2i-1} \bar{v}$ in $O(k \cdot n^3 \cdot \log(n))$ time.*

Proof Sketch. By proving that all properties of $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ from Lemma 10 hold for $\mathbf{PE}(\bar{v}, i, j, \bar{s})$, the approach in Lemma 11 is used to compute $SizePE[i, j, \bar{s}]$ for every $i \in \left[\frac{n}{2}\right], j \in [2i-1], \bar{s} \sqsubseteq_{2i-1} \bar{v}$ in $O(k \cdot n^3 \cdot \log(n))$ time. ◀

▶ **Lemma 22.** *Let $\bar{v} \in \Sigma^n$. The size of $\mathcal{PE}(\bar{v})$ can be computed in $O(k \cdot n^3 \cdot \log(n))$ time.*

Proof Sketch. The size of $\mathcal{PE}(\bar{v})$ is computed in a similar manner to $\mathcal{PO}(\bar{v})$ using $SizePE$ in the same manner as $SizePO$. As before there are two cases to consider for each $\bar{w} \in \mathcal{PE}(\bar{v})$. The first case is where $\bar{w}_{[1,(n/2)]} \in \mathbf{PE}(\bar{v}, \frac{n}{2}, j, \bar{s})$ for some set of arguments $j \in [n-1], \bar{s} \sqsubseteq_{n-1} \bar{v}$. The second is where $\bar{w}_{[1,(n/2)-1]})^R \bar{w}_{[1,(n/2)]} \sqsubseteq_{n-1} \bar{v}$. In both cases the same approach as used in Lemma 13 can be used, determining the number of symbols $x \in \Sigma$ where $\langle (\bar{w}_{[1,(n/2)-1]})^R \bar{w}_{[1,(n/2)]} x \rangle \in \mathcal{PS}(\bar{v})$, using the values of j and $\bar{s}$ rather than directly computing this value for each word. ◀

The size of $\mathcal{PS}(\bar{v})$ is calculated in a similar manner. As the words in $\mathcal{PS}(\bar{v})$ are of the form $\bar{\phi}\bar{\phi}^R$, the prefixes of length i correspond to subwords of length $2i$ with the form $\bar{u}^R\bar{u}$. Note that these are the same as the prefixes used in Section 4.1 for odd length palindromic necklaces. As such, the sets $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ are used to partition internal vertices of the tree $\mathcal{TS}(\bar{v})$. Lemma 23 shows how to use these sets to compute the size of $\mathcal{PS}(\bar{v})$.

▶ **Lemma 23.** *Let $\bar{v} \in \Sigma^n$. The size of $\mathcal{PS}(\bar{v})$ can be computed in $O(k \cdot n^3 \cdot \log(n))$ time.*

Proof Sketch. The size of $\mathcal{PS}(\bar{v})$ is computed using two cases. As before, for every word $\bar{w} \in \mathcal{PS}(\bar{v})$ either $(\bar{w}_{[1,(n/2)-1]})^R \bar{w}_{[1,(n/2)-1]} \sqsubseteq \bar{v}$. or there exists some set $\mathbf{PO}(\bar{v}, \frac{n}{2}-1, j, \bar{s})$ such that $\bar{w}_{[1,(n/2)-1]} \in \mathbf{PS}(\bar{v}, \frac{n}{2}-1, j, \bar{s})$. Following the same approach as in Lemmas 13 and 22, set of arguments $j \in [n-2], \bar{s} \sqsubseteq_{n-2} \bar{v}$ are used to compute the number of symbols $x \in \Sigma$ such that for $\bar{u} \in \mathbf{PS}(\bar{v}, \frac{n}{2}-1, j, \bar{s})$, $\langle \bar{u}x\bar{u}^R \rangle > \bar{v}$. Similarly, for every subword $\bar{u}\bar{u}^R \sqsubseteq_{n-2} \bar{v}$, the number of symbols $x \in \Sigma$ where $\langle \bar{u}x\bar{u}^R \rangle > \bar{v}$. ◀

Combining Lemmas 22 and 23 with Lemmas 18 and 19 provides the tools to compute the rank of $\bar{v}$ among even length palindromic necklaces. Lemma 24 shows how to combine these values to get the rank of $\bar{v}$ among even length palindromic necklaces.

▶ **Lemma 24.** *The rank of $\bar{v} \in \Sigma^n$ among even length palindromic necklaces can be computed in $O(k \cdot n^3 \cdot \log(n)^2)$ time.*

Proof. From Proposition 17, the number of even length palindromic necklaces is equal to $\frac{1}{2}\left(k^{n/2+1} + 2k^{n/2} + k^l\right) - k$, where $l = \frac{n+2}{4}$ if $\frac{n}{2}$ is odd, or $l = \frac{n}{4}$ if $\frac{n}{2}$ is even. Lemma 18 provides an equation to count the number of necklaces greater than $\bar{v}$ containing at least one word of the form $x\bar{\phi}y\bar{\phi}^R$. The equation given by Lemma 18 requires the size of $\mathcal{PE}(\bar{v})$ to be computed, needing at most $O(k \cdot n^3 \cdot \log(n))$ operations, and either $|\mathcal{PE}(\bar{v}_{[1,n/2]})|$ or $GE(\bar{v}_{[1,n/2]})$. As both $|\mathcal{PE}(\bar{v})|$ and $|\mathcal{PO}(\bar{v})|$ require $O(k \cdot n^3 \cdot \log(n))$ operations, the total complexity comes from the number of such sets that must be considered. As the prefixes of $\bar{v}$ that need to be computed is no more than $\log_2(n)$, the total complexity of computing $GE(\bar{v})$ is $O(k \cdot n^3 \cdot \log^2(n))$. Similarly as the complexity of computing $\mathcal{PS}(\bar{v})$ is $O(k \cdot n^3 \cdot \log(n))$, the complexity of computing $GS(\bar{v})$ is $O(k \cdot n^3 \cdot \log^2(n))$. ◀

▶ **Theorem 25.** *Give a word $\bar{v} \in \Sigma^n$, the rank of $\bar{v}$ with respect to the set of palindromic necklaces, $RP(\bar{v})$, can be computed in $O(k \cdot n^3 \cdot \log^2(n))$ time.*

Proof. The number of odd length palindromic necklaces is given by Proposition 8 as $k^{(n-1)/2}$. Lemma 13 shows that the size of set $\mathcal{PO}(\bar{v})$, corresponding to the number of odd length palindromic bracelets, can be computed in $O(k \cdot n^3 \cdot \log(n))$ time. By subtracting the size of

$\mathcal{PO}(\bar{v})$ from $k^{(n-1)/2}$, the rank of $\bar{v}$ can be computed in $O(k \cdot n^3 \cdot \log(n))$ time. Lemma 24 shows that of $RP(\bar{v})$ can be computed in $O(k \cdot n^3 \cdot \log^2(n))$ time if the length of $\bar{v}$ is even. Hence the total complexity is $O(k \cdot n^3 \cdot \log^2(n))$. ◀

5 Enclosing Bracelets

Following Lemma 4 and Theorem 25, the remaining problem is counting the number of enclosing words. This section will provide a technique to count the number of necklaces enclosing some word $\bar{v}$. As in the palindromic case, the structure of these words will first be analysed so that a more efficient algorithm can be derived.

▶ **Proposition 26.** *The bracelet representation of every bracelet $\hat{\mathbf{w}}$ enclosing the word $\bar{v} \in \Sigma^n$ can be written as $\bar{v}_{[1,i]}x\bar{\phi}$ where; $x \in \Sigma$ is a symbol that is strictly smaller than $\bar{v}_{[i+1]}$, and $\bar{\phi} \in \Sigma^*$ is a word such that every rotation of $(\bar{v}_{[1,i]}x\bar{\phi})^R$ is greater than $\bar{v}$.*

Proof Sketch. The claim is shown by proving that any word not of this form has a rotation of the reflection smaller than $\bar{v}$, contradicting the assumption that the bracelet encloses $\bar{v}$. ◀

▶ **Proposition 27.** *Given a bracelet $\hat{\mathbf{w}}$ enclosing the word $\bar{v} \in \Sigma^n$ of the form $\bar{v}_{[1,j]}x\bar{\phi}$ as given in Proposition 26. The value of x must be greater than or equal to $\bar{v}_{[(j+1) \bmod l]}$ where l is the length of the longest Lyndon word that is a prefix of $\bar{v}_{[1,j]}$.*

High Level Idea for the Enclosing Case. Similar to Sections 4.1 and 4.2, the main idea is to use the structure given in Proposition 26 as a basis for counting the number of enclosing bracelets. For each value of i and x, the number of possible values of $\bar{\phi}$ are counted. This is done in a recursive manner, working backwards from the last symbol. For each combination of i and x, the key properties to observe are that (1) every suffix of $\bar{\phi}$ must be greater than or equal to $\bar{v}_{[1,i]}x$ and (2) every rotation of $\bar{\phi}^R x\bar{v}^R_{[1,i]}$ is greater than $\bar{v}$.

These observations are used to create a tree, $\mathcal{TEN}(\bar{v}, i, x)$, where each vertex represents a suffix of some possible value of $\bar{\phi}$. Equivalently, the vertices of $\mathcal{TEN}(\bar{v}, i, x)$ can be thought of as representing the prefixes of $\bar{\phi}^R$. The leaf vertices of $\mathcal{TEN}(\bar{v}, i, x)$ represent the possible values of $\bar{\phi}$. As in Section 4, each layer of $\mathcal{TEN}(\bar{v}, i, x)$ is grouped into sets based on the lexicographical value of the reflection of the suffixes, and the prefixes of the suffixes. Let $t \in [|\bar{w}| - i]$, $j \in [t + i + 1]$ and $\bar{s} \sqsubseteq_{t+i+1} \bar{v}$. For the t^{th} layer of $\mathcal{TEN}(\bar{v}, i, x)$, the set $\mathcal{E}(\bar{v}, i, x, j, \bar{s})$ is introduced containing a subset of the vertices at layer t. The idea is to use the values of j and $\bar{s}$ to divide the prefixes at layer t by lexicographic value and suffix respectively. Let $\bar{u} \in \mathcal{E}(\bar{v}, i, x, j, \bar{s})$ be a suffix of some word $\bar{w}$ such that $\bar{v}_{[1,i]}x\bar{w}$ is a bracelet enclosing $\bar{v}$. To ensure that the necklace represented by the reflection is strictly greater than $\bar{v}$, j is used to track the longest prefix of $\bar{u}^R$ that is a prefix of $\bar{v}$. To ensure that there is no rotation of $x\bar{v}^R_{[1,i]}\bar{w}^R$, the subword $\bar{s} \sqsubseteq_t \bar{v}$ is used to bound the value of $\bar{u}^R$. Formally, $\mathcal{E}(\bar{v}, i, x, j, \bar{s})$ contains every suffix $\bar{u} \in \mathcal{TEN}(\bar{v}, i, x)$ of length i where (1) the longest prefix of $\bar{u}^R$ that is also a prefix of $\bar{v}$ and (2) the subword $\bar{s} \sqsubseteq_t \bar{v}$ bounds $\bar{u}^R$.

As in Section 4 the number of leaf vertices are calculated by determining the size of the sets $\mathcal{E}(\bar{v}, i, x, j, \bar{s})$ at layer $|\bar{v}| - i - 2$, and the number of children of each set. To determine the size of the sets, two key observations must be made. The first is that given the word $\bar{u} \in \mathcal{E}(\bar{v}, i, x, j, \bar{s})$ and the symbol $y \in \Sigma$, if $y\bar{u} \in \mathcal{TEN}(\bar{v}, i, x)$ then there exists some pair $j' \in [n], \bar{s}' \sqsubseteq_{|\bar{u}|+1} \bar{v}$ such that $y\bar{u} \in \mathcal{E}(\bar{v}, i, x, j', \bar{s}')$. Secondly, if $y\bar{u} \in \mathcal{E}(\bar{v}, i, x, j', \bar{s}')$, then $y\bar{w} \in \mathcal{E}(\bar{v}, i, x, j', \bar{s}')$ for every $\bar{w} \in \mathcal{E}(\bar{v}, i, x, j, \bar{s})$. These observations are proven in Lemma 28, as well as showing how to determine the values of j' and $\bar{s}'$.

▶ **Lemma 28.** *Given $\bar{u} \in \mathcal{E}(\bar{v}, i, x, j, \bar{s})$ and symbol $y \in \Sigma$, the pair $j' \in [n], \bar{s}' \sqsubseteq_{|\bar{u}|+1} \bar{v}$ such that $y\bar{u} \in \mathcal{E}(\bar{v}, i, x, j', \bar{s}')$ can be computed in constant time. Further, if $y\bar{u} \in \mathcal{E}(\bar{v}, i, x, j', \bar{s}')$, then $y\bar{w} \in \mathcal{E}(\bar{v}, i, x, j', \bar{s}')$ for every $\bar{w} \in \mathcal{E}(\bar{v}, i, x, j, \bar{s})$.*

Proof Sketch. The proof follows the same arguments as Lemma 10: the value of j' is either $j+1$ if $y = \bar{v}_{j+1}$, or 0 otherwise. Then, the value of $\bar{s}'$ is derived using the array XW. ◀

From Lemma 28, the size of $\mathcal{E}(\bar{v}, i, x, j, \bar{s})$ are computed using the sizes of $\mathcal{E}(\bar{v}, i, x, j', \bar{s}')$ for $j' \in [0, n]$ and $\bar{s}' \in \mathbf{S}(\bar{v}, |\bar{s}|+1)$. To compute the value of $\mathcal{E}(\bar{v}, i, x, j, \bar{s})$, an array SE of size $k \times n \times n \times n^2$ is introduced such that the value of $SE[x, i, j, \bar{s}] = |\mathcal{E}(\bar{v}, i, x, j, \bar{s})|$.

▶ **Lemma 29.** *Let $\bar{v} \in \Sigma^n$. Let SE be a $n \times n^2$ array such that $SE[x, i, j, \bar{s}] = |\mathcal{E}(\bar{v}, i, x, j, \bar{s})|$ for $j \in [0, n]$ and $\bar{s} \sqsubseteq \bar{v}$. Every value of $SE[x, i, j, \bar{s}]$ is computed in $O(k^2 \cdot n^4)$ time.*

Proof Sketch. SE is computed using Lemma 28. The value of $SE[x, i, j, \bar{s}]$ is computed starting with $|\bar{s}| = n-1$ for every value of x, i, and j. Then, by iteratively decreasing the length of $\bar{s}$, the value of $SE[x, i, j, \bar{s}]$ for each of the $n^3 \cdot k$ arguments of $x, i, j, \bar{s}$ can be computed in $O(n \cdot k)$ time; hence we need $O(k^2 \cdot n^4)$ time in total. ◀

Once SE has been computed, the number of enclosing words can be computed using SE and each valid combination of i and x. This is done in a direct manner. Note that the number of possible values of $\bar{\phi}$ such that $\bar{v}_{[1,i]}x\bar{\phi}$ represents a bracelet enclosing $\bar{v}$ is equal to $SE[x, i, j, \bar{s}]$ where j is the longest suffix of $\bar{v}_{[2,i]}x$ that is a prefix of $\bar{v}$ and $\bar{s}$ is the subword that bounds $x\bar{v}^R_{[1,i]}$. As both values can be computed naively in $O(n^2)$ operations, the complexity of this problem comes predominately from computing SE.

▶ **Theorem 30.** *The number of bracelets enclosing $\bar{v} \in \Sigma^n$ can be computed in $O(n^4 \cdot k^2)$.*

Proof. From Lemma 29 the array SE may be computed in $O(n^4 \cdot k^2)$ operations. Using SE, let $i \in [1, n]$ and $x \in \Sigma$. Further let l be the length of the longest Lyndon word that is a prefix of $\bar{v}_{[1,i]}$. If the value of x is less than $\bar{v}_{i+1 \bmod l}$ or greater than or equal to $\bar{v}_{i+1}$ then there is no bracelet represented by $\bar{v}_{[1,i]}x\bar{\phi}$. Similarly if $x\bar{v}^R_{[1,i]} < \bar{v}_{[1,i+1]}$, then any bracelet of the form $\bar{v}_{1,i}x\bar{\phi}$ does not enclose $\bar{v}$. Otherwise, the number of enclosing bracelets represented by $\bar{v}_{[1,i]}x\bar{\phi}$ is equal to $SE[x, i, j, \bar{s}']$ where j is the longest suffix of $\bar{v}_{[2,i]}x$ that is a prefix of $\bar{v}$ and $\bar{s}$ is the subword that bounds $x\bar{v}^R_{[1,i]}$. By summing the value of $SE[x, i, j, \bar{s}']$ for each value of $i \in [1, n]$ and $x \in \Sigma$ such that $\bar{v}_{[1,i]}x$ is the prefix of the representation of some bracelet enclosing $\bar{v}$ gives the number of enclosing bracelets. Therefore

$$RE(\bar{v}) = \sum_{i \in [1, n-1]} \sum_{x \in \Sigma} \begin{cases} 0 & x\bar{v}^R_{[1,i]} < \bar{v} \\ 0 & x \leq \bar{v}_{i+1 \bmod l} \text{ or } x > \bar{v}_{i+1} \\ SE[x, i, j, \bar{s}'] & \textit{Otherwise.} \end{cases}$$ ◀

Proof of Theorem 5. The tools are now available to prove Theorem 5 and show that it is possible to rank a word $\bar{v} \in \Sigma^n$ with respect to the set of bracelets of length n over the alphabet Σ in $O(k^2 \cdot n^4)$ steps. To rank bracelets, it is sufficient to use the results of ranking $\bar{v}$ with respect to necklaces, palindromic necklaces and bracelets enclosing $\bar{v}$, combining them as shown in Lemma 4. Sawada et. al. provided an algorithm to rank v with respect to necklaces in $O(n^2)$ time. It follows from Theorem 25 that the rank with respect to palindromic necklaces can be computed in $O(k \cdot n^3)$ time. Theorem 30 shows that the rank with respect to bracelets enclosing v can be computed in $O(k^2 \cdot n^4)$ time. As combining these results can be done in $O(1)$ steps, therefore the overall complexity is $O(k^2 \cdot n^4)$. ◀

Conclusions and Future Work. In this work we have presented an algorithm for the ranking of bracelets in $O(k \cdot n^4)$ time. Additionally, we have presented a complimentary $O(n^4 \cdot k^2 \cdot \log(k))$ time algorithm for unranking. This expands upon the previous work on ranking necklaces and Lyndon words in $O(n^2)$ time, and unranking in $O(n^3)$ time. This leaves the question of if there is a faster algorithm for ranking bracelets, which may be found by deriving a faster algorithm to count the number of enclosing bracelets.

In addition to the importance of the results from the perspective of combinatorics on words, a practical application of combinatorial necklaces and bracelets can be found in the field of chemistry, since they provide discrete representation of periodic motives in crystals. The problems on finding diverse and representative samples of languages of necklaces and bracelets has served as a heuristic in the exploration of the space of crystal structures [2, 3], since the problem is considered to be NP-hard [1]. The essential component for building representative samples require efficient procedures for ranking bracelets.

References

1 Duncan Adamson, Argyrios Deligkas, Vladimir V. Gusev, and Igor Potapov. On the Hardness of Energy Minimisation for Crystal Structure Prediction. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12011 LNCS, pages 587–596. Springer, January 2020. doi:10.1007/978-3-030-38919-2_48.

2 Duncan Adamson, Argyrios Deligkas, Vladimir V. Gusev, and Igor Potapov. The K-Centre Problem for Necklaces. *CoRR*, May 2020. arXiv:2005.10095.

3 C. Collins, M. S. Dyer, M. J. Pitcher, G. F. S. Whitehead, M. Zanella, P. Mandal, J. B. Claridge, G. R. Darling, and M. J. Rosseinsky. Accelerated discovery of two crystal structure types in a complex inorganic phase field. *Nature*, 546(7657):280–284, 2017.

4 Clelia De Felice, Rocco Zaccagnino, and Rosalba Zizza. Unavoidable sets and circular splicing languages. *Theoretical Computer Science*, 658:148–158, 2017. Formal Languages and Automata: Models, Methods and Application In honour of the 70th birthday of Antonio Restivo. doi:10.1016/j.tcs.2016.09.008.

5 Aris Filos-Ratsikas and Paul W. Goldberg. The complexity of splitting necklaces and bisecting ham sandwiches. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 638–649. ACM, 2019. doi:10.1145/3313276.3316334.

6 R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete mathematics : a foundation for computer science*. Addison-Wesley, 1994.

7 U. I. Gupta, D. T. Lee, and C. K. Wong. Ranking and unranking of B-trees. *Journal of Algorithms*, 4(1):51–60, March 1983. doi:10.1016/0196-6774(83)90034-2.

8 Elizabeth Hartung, Hung Phuc Hoang, Torsten Mütze, and Aaron Williams. Combinatorial generation via permutation languages. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1214–1225. SIAM, 2020. doi:10.1137/1.9781611975994.74.

9 S. Karim, J. Sawada, Z. Alamgir, and S. M. Husnine. Generating bracelets with fixed content. *Theoretical Computer Science*, 475:103–112, March 2013. doi:10.1016/j.tcs.2012.11.024.

10 Donald E. Knuth. *The Art of Computer Programming: Combinatorial Algorithms, Part 1*. Addison-Wesley Professional, 1st edition, 2011.

11 T. Kociumaka, J. Radoszewski, and W. Rytter. Computing k-th Lyndon word and decoding lexicographically minimal de Bruijn sequence. In *Symposium on Combinatorial Pattern Matching*, pages 202–211. Springer, 2014.

12 S. Kopparty, M. Kumar, and M. Saks. Efficient indexing of necklaces and irreducible polynomials over finite fields. *Theory of Computing*, 12(1):1–27, 2016.

13 Martin Mareš and Milan Straka. Linear-time ranking of permutations. In Lars Arge, Michael Hoffmann, and Emo Welzl, editors, *Algorithms – ESA 2007*, pages 187–193, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

14 Wendy Myrvold and Frank Ruskey. Ranking and unranking permutations in linear time. *Information Processing Letters*, 79(6):281–284, 2001. doi:10.1016/S0020-0190(01)00141-7.

15 J. M. Pallo. Enumerating, Ranking and Unranking Binary Trees. *The Computer Journal*, 29(2):171–175, February 1986. doi:10.1093/comjnl/29.2.171.

16 J. Sawada and A. Williams. Practical algorithms to rank necklaces, Lyndon words, and de Bruijn sequences. *Journal of Discrete Algorithms*, 43:95–110, 2017.

17 Joe Sawada. Generating bracelets in constant amortized time. *SIAM Journal on Computing*, 31(1):259–268, January 2001. doi:10.1137/S0097539700377037.

18 Toshihiro Shimizu, Takuro Fukunaga, and Hiroshi Nagamochi. Unranking of small combinations from large sets. *Journal of Discrete Algorithms*, 29:8–20, 2014. doi:10.1016/j.jda.2014.07.004.

19 S. G. Williamson. Ranking algorithms for lists of partitions. *SIAM Journal on Computing*, 5(4):602–617, 1976. doi:10.1137/0205039.