On Reverse Engineering the Lyndon Tree

Yuto Nakashima[1], Takuya Takagi[2,3], Shunsuke Inenaga[1],
Hideo Bannai[1], and Masayuki Takeda[1]

[1] Department of Informatics, Kyushu University, Japan
{yuto.nakashima, inenaga, bannai, takeda}@inf.kyushu-u.ac.jp
[2] Graduate School of IST, Hokkaido University, Japan
tkg@ist.hokudai.ac.jp
[3] Japan Society for the Promotion of Science (JSPS), Japan

Abstract. We consider the problem of reverse engineering the Lyndon tree, i.e., given a full binary ordered tree T with n leaves as input, compute a string w of length n for which it's Lyndon tree is isomorphic to the input tree. Although the problem is easy and solvable in linear time when assuming a binary alphabet or when there is no limit on the alphabet size, how to efficiently find the smallest alphabet size for a solution string is not known. We show several new observations concerning this problem. Namely, we show that: 1) For any full binary ordered tree T, there exists a solution string w over an alphabet of size at most $h+1$, where h is the height of T. 2) For any positive n, there exists a full binary ordered tree T with n leaves, s.t. the smallest alphabet size of the solution string for T is $\lfloor \frac{n}{2} \rfloor + 1$.

1 Introduction

The problem of efficiently inferring a string from a given data structure that is defined based on the string has been considered in many contexts; for example, border array [9], directed acyclic word graph [2], suffix array [2], suffix tree [11,5,17], the run structure of a word [16], LCP array [12], etc. The motivation is to elucidate and better understand the combinatorial properties of the data structures in question, which could lead to better algorithms on constructing, representing, or using the structures.

In this paper, we consider the problem of inferring a string from its Lyndon tree [3]. A string is a *Lyndon word* [14] if it is lexicographically smaller than any of its proper suffixes. Given a Lyndon word w of length $n > 1$, (u, v) is the *standard factorization* [6,13] of w, if $w = uv$ and v is the longest proper suffix of w that is a Lyndon word, or equivalently, the lexicographically smallest proper suffix of w. It is well known that for the standard factorization (u, v) of any Lyndon word w, the factors u and v are also Lyndon words (e.g.[4,13]). The *Lyndon tree* of w is the full binary tree defined by recursive standard factorization of w; w is the root of the Lyndon tree of w, its left child is the root of the Lyndon tree of u, and its right child is the root of the Lyndon tree of v. Figure 1 shows an example of a Lyndon tree for the Lyndon word `aababaababb`. Lyndon trees have recently been shown to have connection with the structure of maximal repeats, or runs, contained in the string [1].

Releated Work

Franek et al. [8] considered the problem of reconstructing the string from its *Lyndon array*. The Lyndon array is an array of integers which contains the length of the longest Lyndon word that starts at each position. For example, for the string `aababaababb`, the Lyndon array is $(11, 2, 1, 2, 1, 6, 5, 1, 3, 1, 1)$. Since a node in the

Yuto Nakashima, Takuya Takagi, Shunsuke Inenaga, Hideo Bannai, Masayuki Takeda: On Reverse Engineering the Lyndon Tree, pp. 108–117.
Proceedings of PSC 2017, Jan Holub and Jan Žďárek (Eds.), ISBN 978-80-01-06193-0

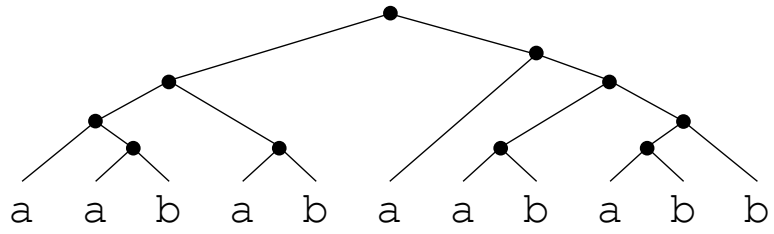

Figure 1. The Lyndon tree for the Lyndon word `aababaababb` with respect to order `a` $\prec$ `b`.

Lyndon tree is a right child of its parent if and only if the node corresponds to the longest Lyndon word that starts at that position ([1] (Lemma 22)), the Lyndon array of a Lyndon word is simply a different representation of the Lyndon tree. It is straightforward to check whether such an array corresponds to a tree topology, and if so retrieve the tree. The remaining problem is to find a string whose Lyndon tree matches this tree structure.

2 Preliminaries

2.1 Strings

Let Σ be a finite *alphabet*. An element of Σ^* is called a *string*. The set of characters contained in a string w is denoted by $\Sigma(w)$. The length of a string w is denoted by $|w|$. The empty string ε is a string of length 0, namely, $|\varepsilon| = 0$. Let Σ^+ be the set of non-empty strings, i.e., $\Sigma^+ = \Sigma^* - \{\varepsilon\}$. For a string $w = xyz$, x, y and z are called a *prefix*, *substring*, and *suffix* of w, respectively. A prefix x is and a suffix z of w are respectively called a *proper prefix*, and *proper suffix* of w, if $x \neq w$ and $z \neq w$. The i-th character of a string w is denoted by $w[i]$, where $1 \leq i \leq |w|$. For a string w and two integers $1 \leq i \leq j \leq |w|$, let $w[i..j]$ denote the substring of w that begins at position i and ends at position j. For convenience, let $w[i..j] = \varepsilon$ when $i > j$.

2.2 Binary trees

A tree is said to be a binary tree if each internal node has at most two children. In this paper, we use full binary trees and complete binary trees as the representation of Lyndon trees. A binary tree is said to be a *full binary tree* if each internal node has exactly two children. We denote the set of full binary trees with n leaves by FBT_n. A binary tree is said to be a *complete binary tree* if the tree is a full binary tree and all the leaves have the same depth. We denote the set of complete binary tree with n leaves by CBT_n.

Parenthesis representation for full binary trees We will use a parenthesis representation for full binary trees, where a node is represented as a sequence of parenthesis representations of the subtrees rooted at each children, enclosed in parentheses. For example, the full binary tree (without each character at the leaves) shown in Figure 1 can be represented as follows.

`((((()(()()))(()()))(()((()())((()())()))))`

For brevity, we use the symbol $\Diamond$ instead of `()` which represents a leaf. We will also sometimes use variables to represent a full binary tree; if T is a tree represented by `(((())(((())())))`, then the above example can also be represented as follows.

$$(((\Diamond(\Diamond\Diamond)(\Diamond\Diamond)(\Diamond T))$$

2.3 Properties on Lyndon words and Lyndon Trees

Lemma 1 (Proposition 1.3 of [7], [13]). *For any Lyndon words λ_1 and λ_2, $\lambda_1\lambda_2$ is a Lyndon word iff $\lambda_1 \prec \lambda_2$.*

It is easy to see that $\lambda_1 \prec \lambda_1\lambda_2 \prec \lambda_2$ also holds. By the definition of Lyndon trees, each node in a Lyndon tree represents a Lyndon word. For any node w in a Lyndon tree, let $str(w)$ be the Lyndon word which is represented by w. From Lemma 1 and the definition of Lyndon trees, for any internal node w in a Lyndon tree (let $u(v)$ be the left(right) child of w, respectively), $str(u) \prec str(w) \prec str(v)$ holds. Assume that λ is a Lyndon word which corresponds to a Lyndon tree. We can extend this lexicographic relation between $str(u)$ (or $str(w)$) and $str(v)$ to the lexicographic relation between suffixes of λ at each beginning position. In other word, the suffix of λ which begins at beginning position of $str(u)$ is lexicographically smaller than the suffix of λ which begins at beginning position of $str(v)$. For any nodes u, v s.t. v is not a sibling of u and $str(u)$ is followed by $str(v)$ in λ, $str(u) \succeq str(v)$ holds. This relation also can be extended to the relation between two suffixes. In this paper, we denote the Lyndon tree of a Lyndon word λ by $LTree(\lambda)$.

3 Reverse Engineering of the Lyndon Trees

Our reverse engineering problem on Lyndon trees is formalized as follows.

Problem 2. Given a full binary tree T with n leaves, compute a Lyndon word of length n s.t. its Lyndon tree is isomorphic to T.

For any ordered tree T_1 and T_2, we will write $T_1 \equiv_I T_2$ if T_1 and T_2 are isomorphic. Firstly, we summarize known results for our problem in Section 3.1. Secondly, we give an upper bound on the alphabet size of an output string in Section 3.2. Thirdly, we also give a lower bound in Section 3.3. Finally, we consider our problem restricted to complete binary trees in Section 3.4.

3.1 Algorithms on restricted alphabet

The following two Lemmas have been shown by Franek et al. [8] in a slightly different context.

Lemma 3 (algorithm on binary alphabet). *For any $T \in FBT_n$, we can compute a Lyndon word λ s.t. $|\Sigma(\lambda)| \leq 2$ and $LTree(\lambda) \equiv_I T$ in $O(n)$ time. (If no Lyndon word exists, return false.)*

Lemma 4 (algorithm on any alphabet). *For any $T \in FBT_n$, we can compute a Lyndon word λ s.t. $|\Sigma(\lambda)| = n$ and $LTree(\lambda) \equiv_I T$ in $O(n)$ time. (Lyndon word w always exists.)*

3.2 Upper bounds

We consider *Suffix Ordered Tree* of a full binary tree T, denoted by $SOT(T)$. We call the tree obtained by removing all leaves (as well as their incoming edges) from a full binary tree T the *internal tree* of T. $SOT(T)$ is a full binary tree which is isomorphic to the internal tree of T. For any node u in $SOT(T)$, let u' be the node in T which corresponds to u, and $pos(u)$ be the number i s.t. the leftmost leaf in subtree rooted at the right child of u' is the i-th leaf from left. For any node u in $SOT(T)$, u is labeled by $pos(u)$. Figure 2 shows an example of $SOT(T)$.

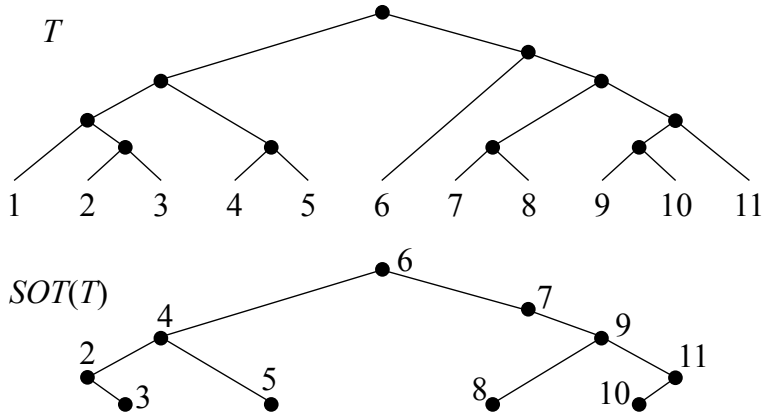

Figure 2. The suffix ordered tree for the full binary tree T.

This tree is known to be related to the *(inverse) suffix array* [15], as shown in the following Lemma. For any string w, we denote the suffix array (resp. inverse suffix array) by $SA(w)$ (resp. $ISA(w)$).

Lemma 5 ([10]). *For any Lyndon word λ, the internal tree of $LTree(\lambda)$ is isomorphic to the Cartesian tree of $ISA(\lambda)[2..|\lambda|]$.*

From Lemma 5, $SOT(T)$ represents necessary conditions w.r.t. lexicographic order of proper suffixes of a Lyndon word λ s.t. $LTree(\lambda) \equiv_I T$. For example, the suffix of λ at position 6 has to be the lexicographically smallest proper suffix of λ. Also, the suffix of λ at position 9 has to be lexicographically smaller than suffixes of λ at position 8, 11 and 10. By the definition of Lyndon words, the suffix at position 1 is always the smallest suffix of λ. Thus, for any Lyndon word λ, the suffix array of λ satisfies suffix orders represented by $SOT(T)$ iff $LTree(\lambda) \equiv_I T$. Then we can obtain the following upper bound of the alphabet size of λ.

Lemma 6 (upper bound). *For any $T \in FBT_n$ of height h, there exists a Lyndon word λ s.t. $|\Sigma(\lambda)| \leq h+1$ and $LTree(\lambda) \equiv_I T$.*

Proof. We consider the string λ obtained by the following operations. For any node u, we assigned a character c which is lexicographically larger than the character assigned to the parent. If u is labeled by i, $\lambda[i] = c$. Let $\lambda[1]$ be the smallest character in λ s.t. the character does not occur at any other positions.

Then the suffix array of λ does not contradict to $SOT(T)$, and λ includes at most $h+1$ distinct characters (since there exist h nodes on the longest path in $SOT(T)$). □

3.3 Lower bounds

Lemma 7 (lower bound for even length). *For any positive integer $k \geq 1$, there exists a full binary tree $T \in FBT_{2k}$ s.t.* $\min\{|\Sigma(\lambda)| : T \equiv_I LTree(\lambda)\} = k+1$.

Proof. We prove the lemma by induction on k. Although it suffices to show one tree for each k, we describe all the trees that we have discovered. For $k = 1$ consider the tree $(\Diamond\Diamond) \in FBT_2$, and for $k = 2$, consider the tree $((\Diamond\Diamond)(\Diamond\Diamond))$ or $((\Diamond(\Diamond\Diamond))\Diamond)$. It can be checked by exhaustive enumeration of strings λ of length $2k$ and $|\Sigma(\lambda)| \leq k+1$ that these trees satisfy the statement of the lemma. Next, assume that the lemma holds for all $k \leq i$ for some i and let $T \in FBT_{2i}$ be a full binary tree that satisfies the statement for $k = i$. Let $g(T)$ be the full binary tree $((\Diamond T)\Diamond)$ for any full binary tree T. We claim that the full binary tree $g(T) \in FBT_{2i+2}$ satisfies the statement.

Suppose there exists a Lyndon word $c_1\lambda c_2$ ($c_1, c_2 \in \Sigma$) of length $2i+2$ s.t. $g(T) \equiv_I LTree(c_1\lambda c_2)$. From the induction hypothesis, we have that $|\Sigma(\lambda)| = i+1$ and thus $|\Sigma(c_1\lambda c_2)| \geq i+1$. Due to the structure of $g(T)$, it must also be that $LTree(\lambda) \equiv_I T$. Since $g(T)$ is the Lyndon tree of $c_1\lambda c_2$, $c_1 \prec c_2 \preceq \lambda[1]$ holds. To prove $|\Sigma(c_1\lambda c_2)| \geq i+2$, assume to the contrary that $|\Sigma(c_1\lambda c_2)| \leq i+1$. Then, it must be that $c_1 = \lambda[1]$. However, this implies that $c_1 = c_2$ and contradicts that $c_1\lambda c_2$ is a Lyndon word. Thus, $|\Sigma(c_1\lambda c_2)| \geq i+2$. On the other hand, if we let λ be a string implied by the lemma for $k = i$, $c_1 \prec \lambda[1]$ a new character not in $\Sigma(\lambda)$, and $c_2 = \lambda[1]$, it is easy to see that $c_1\lambda c_2$ is a Lyndon word, $\Sigma(c_1\lambda c_2) = i+2$ and $g(T) \equiv_I LTree(c_1\lambda c2)$. Thus, the statement holds for $k = i+1$, proving the lemma. □

We define the set G of full binary trees described in Lemma 7 as follows : $G = \{g^k(T) \mid T \in \{((\Diamond\Diamond)(\Diamond\Diamond)), g((\Diamond\Diamond))\}, k \geq 1\}$.

Lemma 8 (lower bound for odd length). *For any positive integer $k \geq 1$, there exists a full binary tree $T \in FBT_{2k+1}$ s.t.* $\min\{|\Sigma(\lambda)| : T \equiv_I LTree(\lambda)\} = k+1$.

Proof. We prove the lemma by induction on k. Although it suffices to show one tree for each k, we describe all the trees that we have discovered. For $k = 1$ consider the tree $(\Diamond(\Diamond\Diamond))$ or $((\Diamond\Diamond)\Diamond)$ in FBT_3, for $k = 2$, consider one of the 8 trees in FBT_5, for $k = 3$, consider one of the 22 trees in FBT_7, and for $k = 4$, consider one of the 34 trees in FBT_9. It can be checked by exhaustive enumeration of strings λ of length $2k+1$ and $|\Sigma(\lambda)| \leq k+1$ that these trees satisfy the statement of the lemma. Next, assume that the lemma holds for all $k \leq i$ for some $i > 3$. We claim that all full binary trees in FBT_{2i+3} obtained by the construction described below, satisfy the lemma.

1. Let $T \in FBT_{2i+1}$ be a full binary tree that satisfies the statement for $k = i$. Consider the full binary tree $g(T) \in FBT_{2i+3}$ and suppose there exists a Lyndon word $c_1\lambda c_2$ s.t. $g(T) \equiv_I LTree(c_1\lambda c_2)$. Then, by similar arguments as in Lemma 7, we can see that $|\Sigma(c_1\lambda c_2)| \geq i+2$, while c_1, c_2 can be chosen so that $|\Sigma(c_1\lambda c_2)| = i+2$ and $g(T) \equiv_I LTree(c_1\lambda c_2)$.
2. Let $T \in FBT_{2i+2} \cap G$, and λ a Lyndon word s.t. $T \equiv_I LTree(\lambda)$. By Lemma 7, $|\Sigma(\lambda)| \geq i+2$. Consider the full binary tree $(\Diamond T) \in FBT_{2i+3}$ or $(T\Diamond) \in FBT_{2i+3}$. Suppose there exists a Lyndon word $c_1\lambda$ (resp. λc_2) s.t. $(\Diamond T) \equiv_I LTree(c_1\lambda)$ (resp. $(T\Diamond) \equiv_I LTree(\lambda c_2)$). If we let $c_1 = \lambda[1]$ (resp. $c_2 = \lambda[|\lambda|]$), it is easy to see that $c_1\lambda$ (resp. λc_2) is a Lyndon word and $(\Diamond T) \equiv_I LTree(c_1\lambda)$ (resp. $(T\Diamond) \equiv_I LTree(\lambda c_2)$). Then, $|\Sigma(c_1\lambda)| = |\Sigma(\lambda c_2)| = i+2$.

3. Let $T \in FBT_{2i} \cap G$, and λ a Lyndon word s.t. $T \equiv_I LTree(\lambda)$. By Lemma 7, $|\Sigma(\lambda)| \geq i+1$. Consider the full binary tree $T' = ((\Diamond\Diamond)(\Diamond T)) \in FBT_{2i+3}$ and suppose there exists a Lyndon word $c_1c_2c_3\lambda$ s.t. $T' \equiv_I LTree(c_1c_2c_3\lambda)$. From the structure of the tree, it must be that $c_1 \preceq c_3 \prec c_2$, $c_3 \preceq \lambda[1]$, and $c_1c_2 \preceq c_3\lambda[1]$. If $c_3 = \lambda[1]$, then $c_1 \prec \lambda[1]$ must hold. Thus, we have that either $c_1 \prec \lambda[1]$ or $c_3 \prec \lambda[1]$, i.e., at least one of c_1 or c_3 has to be a new smaller character not contained in $\Sigma(\lambda)$, and thus $|\Sigma(c_1c_2c_3\lambda)| \geq k+1$. On the other hand, if we choose $c_1 \prec c_3 = \lambda[1] \prec c_2 = \lambda[|\lambda|]$, $c_1c_2c_3\lambda$ is a Lyndon word s.t. $|\Sigma(c_1c_2c_3\lambda)| = k+1$ and $T' \equiv_I LTree(c_1c_2c_3\lambda)$.
4. Let $T \in FBT_{2i} \cap G$, and λ a Lyndon word s.t. $T \equiv_I LTree(\lambda)$. By Lemma 7, $|\Sigma(\lambda)| \geq i+1$. Consider the full binary tree $T' = (((\Diamond\Diamond)T)\Diamond) \in FBT_{2i+3}$, and suppose there exists a Lyndon word $c_1c_2\lambda c_3$ s.t. $T' \equiv_I LTree(c_1c_2\lambda c_3)$. From the structure of the tree, it must be that $c_1 \prec c_3 \preceq \lambda[1]$. Thus c_1 has to be a new smaller character not contained in $\Sigma(\lambda)$, and thus $|\Sigma(c_1c_2\lambda c_3)| \geq i+2$. On the other hand, if we choose $c_1 \prec c_3 = \lambda[1] \prec c_2 = \lambda[|\lambda|]$, $c_1c_2\lambda c_3$ is a Lyndon word s.t. $|\Sigma(c_1c_2\lambda c_3)| = i+2$ and $T' \equiv_I LTree(c_1c_2\lambda c_3)$.
5. Let $T \in FBT_{2i} \cap G$, and λ a Lyndon word s.t. $T \equiv_I LTree(\lambda)$. By Lemma 7, $|\Sigma(\lambda)| \geq i+1$. Consider the full binary tree $T' = ((T(\Diamond\Diamond))\Diamond) \in FBT_{2i+3}$, and suppose there exists a Lyndon word $\lambda c_1c_2c_3$ s.t. $T' \equiv_I LTree(\lambda c_1c_2c_3)$. Since $T = g(T'')$ for some $T'' \in G$, it follows from the arguments in Lemma 7 that the structure of T implies that $\lambda[1] \prec \lambda[|\lambda|] \preceq \lambda[2]$. Notice that since $\lambda[2..|\lambda|-1]$ is a Lyndon word, $\lambda[1]$ and $\lambda[|\lambda|]$ are the two smallest characters in $\Sigma(\lambda)$. From the structure of T', it must be that $\lambda[1] \prec c_3 \preceq c_1 \prec \lambda[|\lambda|]$, implying that $c_1, c_3 \notin \Sigma(\lambda)$ and $|\Sigma(\lambda c_1c_2c_3)| \geq i+2$. One the other hand, if we choose $\lambda[1] \prec c_1 = c_3 \prec \lambda[|\lambda|] = c_2c$, $\lambda c_1c_2c_3$ is a Lyndon word s.t. $|\Sigma(\lambda c_1c_2c_3)| = i+2$ and $T' \equiv_I LTree(\lambda c_1c_2c_3)$.
6. Let $T \in FBT_{2i-2} \cap G$, and λ a Lyndon word s.t. $T \equiv_I LTree(\lambda)$. By Lemma 7, $|\Sigma(\lambda)| \geq i$. Consider the full binary tree $T' = (((\Diamond(\Diamond T))(\Diamond\Diamond))\Diamond) \in FBT_{2i+3}$, and suppose there exists a Lyndon word $c_1c_2\lambda c_3c_4c_5$ s.t. $T \equiv_I LTree(c_1c_2\lambda c_3c_4c_5)$. From the structure of T', it must be that $c_1 \prec c_5 \preceq c_3 \preceq c_2 \preceq \lambda[1]$, $c_3 \prec c_4$, and $c_3c_4 \preceq c_2\lambda[1]$. If $c_3 = \lambda[1]$, this implies $c_2 = \lambda[1]$, but then $c_3c_4 \preceq c_2\lambda[1]$ cannot hold. Thus we have that $c_1 \prec c_3 \prec \lambda[1]$, and thus $|\Sigma(c_1c_2\lambda c_3c_4c_5)| \geq i+2$. On the other hand, if we choose $c_1 \prec c_5 = c_3 = c_2 \prec \lambda[1] = c_4$, $c_1c_2\lambda c_3c_4c_5$ is a Lyndon word s.t. $|\Sigma(c_1c_2\lambda c_3c_4c_5)| = i+2$ and $T' \equiv_I LTree(c_1c_2\lambda c_3c_4c_5)$.

□

We can obtain the following theorem by Lemma 7 and Lemma 8.

Theorem 9. *For any positive integer $n \geq 1$, there exists a full binary tree $T \in FBT_n$ s.t.* $\min\{|\Sigma(\lambda)| : T \equiv_I LTree(\lambda)\} = \lfloor \frac{n}{2} \rfloor + 1$.

Conjectures on lower bound We conjecture that any full binary tree which satisfies the above theorem is one of the trees described in Lemma 7 and Lemma 8. For any k, we have two types of trees which satisfies Lemma 7. In Lemma 8, for any k, there exist 12 new types of trees due to case 2-6. This implies that $\lfloor \frac{n}{2} \rfloor + 1$ is also an upper bound.

3.4 Problem on complete binary trees

Here, we restrict the input of our problem and consider a complete binary tree with 2^k leaves. We obtain the following theorem.

Theorem 10. *For any integer $k \geq 0$, there exists a Lyndon word λ s.t. $|\Sigma(\lambda)| \leq 4$ and $LTree(\lambda) \equiv_I CBT_{2^k}$.*

To prove this theorem, we consider a homomorphism f defined as follows.

$$f(\mathtt{a}) = \mathtt{ac}, f(\mathtt{b}) = \mathtt{ad}, f(\mathtt{c}) = \mathtt{bc}, f(\mathtt{d}) = \mathtt{bd}$$

We show an example in Figure 3. By the definition of f, we have the following lemma.

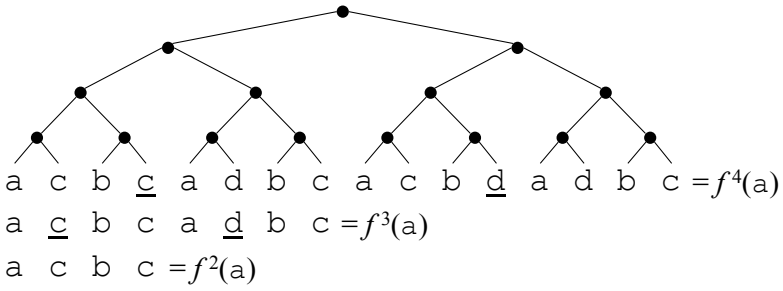

Figure 3. The Lyndon tree of $f^4(\mathtt{a})$.

Lemma 11. *For any $k \geq 3$, the following properties hold.*

1. $f^{k+1}(\mathtt{a}) = f^k(\mathtt{a}) \cdot f^k(\mathtt{a})[1..\frac{|f^k(\mathtt{a})|}{2} - 1] \cdot \mathtt{d} \cdot f^k(\mathtt{a})[\frac{|f^k(\mathtt{a})|}{2} + 1..|f^k(\mathtt{a})|]$.
2. *the length of longest common prefix of $f^k(\mathtt{a})$ and any of its proper suffix is less than $\frac{|f^k(\mathtt{a})|}{4}$.*

Proof.

1. We prove this by induction on k. For $k = 3$, $f^4(\mathtt{a}) = \mathtt{acbcadbcacbdadbc}$ satisfies the statement by $f^3(\mathtt{a}) = \mathtt{acbcadbc}$. We assume that

$$f^{k+1}(\mathtt{a}) = f^k(\mathtt{a}) \cdot f^k(\mathtt{a})[1..\frac{|f^k(\mathtt{a})|}{2} - 1] \cdot \mathtt{d} \cdot f^k(\mathtt{a})[\frac{|f^k(\mathtt{a})|}{2} + 1..|f^k(\mathtt{a})|]$$

holds for all $3 \leq k \leq i$ for some i. Since f derives two characters from each character,

$$f^{i+2}(\mathtt{a})[1..\frac{|f^{i+2}(\mathtt{a})|}{2}]$$

is derived from

$$f^{i+1}(\mathtt{a})[1..\frac{|f^{i+1}(\mathtt{a})|}{2}] = f^i(\mathtt{a}).$$

Thus

$$f^{i+2}(\mathtt{a})[1..\frac{|f^{i+2}(\mathtt{a})|}{2}] = f(f^i(\mathtt{a})) = f^{i+1}(\mathtt{a}).$$

Similarly,

$$\begin{aligned} f^{i+2}(\mathtt{a})[\frac{3}{4}|f^{i+2}(\mathtt{a})| + 1..|f^{i+2}(\mathtt{a})|] &= f(f^i(\mathtt{a})[\frac{|f^i(\mathtt{a})|}{2} + 1..|f^i(\mathtt{a})|]) \\ &= f^{i+1}(\mathtt{a})[\frac{|f^{i+1}(\mathtt{a})|}{2} + 1..|f^{i+1}(\mathtt{a})|]. \end{aligned}$$

Let

$$X = f^{i+1}(\mathtt{a})[1..\frac{|f^{i+1}(\mathtt{a})|}{4} - 1]$$
$$= f^{i+1}(\mathtt{a})[\frac{|f^{i+1}(\mathtt{a})|}{2} + 1..\frac{3}{4}|f^{i+1}(\mathtt{a})| - 1].$$

Then

$$f^{i+2}(\mathtt{a})[\frac{|f^{i+2}(\mathtt{a})|}{2} + 1..\frac{3}{4}|f^{i+2}(\mathtt{a})|] = f(X)\mathtt{bd}.$$

It is clear that the last character of $f^k(\mathtt{a})$ is $\mathtt{c}$ for any $k \geq 2$. Thus

$$f(X)\mathtt{b} = f^{i+1}(\mathtt{a})[1..\frac{|f^{i+1}(\mathtt{a})|}{2} - 1].$$

Therefore,

$$f^{i+2}(\mathtt{a}) = f^{i+1}(\mathtt{a}) \cdot f^{i+1}(\mathtt{a})[1..\frac{|f^{i+1}(\mathtt{a})|}{2} - 1] \cdot \mathtt{d} \cdot f^{i+1}(\mathtt{a})[\frac{|f^{i+1}(\mathtt{a})|}{2} + 1..|f^{i+1}(\mathtt{a})|],$$

and the statement holds for $i + 1$.

2. From the above arguments, the longest substring of $f^k(\mathtt{a})$ which is also a prefix is

$$f^k(\mathtt{a})[\frac{|f^k(\mathtt{a})|}{2} + 1..\frac{3}{4}|f^k(\mathtt{a})| - 1].$$

Thus the statement holds.

□

Lemma 12. *Let $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$ be an ordered alphabet of size 4 s.t. $\mathtt{a} \prec \mathtt{b} \prec \mathtt{c} \prec \mathtt{d}$. For any integer $k \geq 0$, $LTree(f^k(\mathtt{a})) \equiv_I CBT_{2^k}$.*

Proof. We prove the lemma by induction on k. For $k = 1$ consider the string $f^1(\mathtt{a}) = \mathtt{ac}$, for $k = 2$, consider the string $f^2(\mathtt{a}) = \mathtt{acbc}$, and for $k = 3$, consider the string $f^3(\mathtt{a}) = \mathtt{acbcadbc}$. We can see the strings satisfy the statement of the lemma. Next, assume that the lemma holds for all $k \leq i$ for some i. We claim that $LTree(f^{i+1}(\mathtt{a})) \equiv_I CBT_{2^{i+1}}$ holds.

From Lemma 11,

$$f^{i+1}(\mathtt{a}) = f^i(\mathtt{a}) \cdot f^i(\mathtt{a})[1..\frac{|f^i(\mathtt{a})|}{2} - 1] \cdot \mathtt{d} \cdot f^i(\mathtt{a})[\frac{|f^i(\mathtt{a})|}{2} + 1..|f^i(\mathtt{a})|],$$

and thus

$$SA(f^{i+1}(\mathtt{a})[\frac{|f^{i+1}(\mathtt{a})|}{2} + 1..|f^{i+1}(\mathtt{a})|]) = SA(f^i(\mathtt{a})).$$

Thus

$$f^{i+1}(\mathtt{a})[\frac{|f^{i+1}(\mathtt{a})|}{2} + 1..|f^{i+1}(\mathtt{a})|]$$

is a Lyndon word and $LTree(f^{i+1}(\mathtt{a})[\frac{|f^{i+1}(\mathtt{a})|}{2}+1..|f^{i+1}(\mathtt{a})|]) \equiv_I LTree(f^i(\mathtt{a})) \equiv_I CBT_{2^i}$ holds. For any proper suffix w of $f^i(\mathtt{a})$, w is lexicographically larger than

$$f^{i+1}(\mathtt{a})[\frac{|f^{i+1}(\mathtt{a})|}{2} + 1..|f^{i+1}(\mathtt{a})|],$$

and $f^i(\mathtt{a})$ is lexicographically smaller than

$$f^{i+1}(\mathtt{a})[\frac{|f^{i+1}(\mathtt{a})|}{2}+1..|f^{i+1}(\mathtt{a})|].$$

From the induction hypothesis, $LTree(f^i(\mathtt{a})) \equiv_I LTree(f^{i+1}(\mathtt{a})[1..\frac{|f^{i+1}(\mathtt{a})|}{2}]) \equiv_I CBT_{2^i}$. Therefore, $LTree(f^{i+1}(\mathtt{a})) \equiv_I CBT_{2^{i+1}}$ and the statement holds for $k = i+1$. □

4 Conclusions and open question

We considered reverse engineering problems on Lyndon trees. We showed that: 1) For any full binary ordered tree T, there exists a solution string w over an alphabet of size at most $h+1$, where h is the height of T. 2) For any positive n, there exists a full binary ordered tree T with n leaves, s.t. the smallest alphabet size of the solution string for T is $\lfloor \frac{n}{2} \rfloor + 1$. We also conjectured that the trees described in Lemmas 7 and 8 are the only trees that satisfy the statements. We discovered the property on Lyndon trees which are isomorphic to complete binary trees.

Our remaining interest which is most important is an algorithm to reconstruct a Lyndon word s.t. the Lyndon tree is isomorphic to an input full binary tree and the alphabet size is smallest possible.

References

1. H. Bannai, T. I, S. Inenaga, Y. Nakashima, M. Takeda, and K. Tsuruta: *The "Runs" Theorem.* CoRR, abs/1406.0263 2014.
2. H. Bannai, S. Inenaga, A. Shinohara, and M. Takeda: *Inferring strings from graphs and arrays*, in Mathematical Foundations of Computer Science 2003, 28th International Symposium, MFCS 2003, Bratislava, Slovakia, August 25-29, 2003, Proceedings, 2003, pp. 208–217.
3. H. Barcelo: *On the action of the symmetric group on the free Lie algebra and the partition lattice.* Journal of Combinatorial Theory, Series A, 55(1) 1990, pp. 93–129.
4. F. Bassino, J. Clément, and C. Nicaud: *The standard factorization of Lyndon words: an average point of view.* Discrete Mathematics, 290(1) 2005, pp. 1–25.
5. B. Cazaux and E. Rivals: *Reverse engineering of compact suffix trees and links: A novel algorithm.* Journal of Discrete Algorithms, 28 2014, pp. 9 – 22.
6. K. T. Chen, R. H. Fox, and R. C. Lyndon: *Free differential calculus. iv. the quotient groups of the lower central series.* Annals of Mathematics, 68(1) 1958, pp. 81–95.
7. J. Duval: *Factorizing words over an ordered alphabet.* J. Algorithms, 4(4) 1983, pp. 363–381.
8. F. Franek, J. W. Daykin, J. Holub, A. S. M. S. Islam, and W. F. Smyth: *Reconstructing a string from its Lyndon arrays.* Theoretical Computer Science, 2017, p. in press.
9. F. Franek, S. Gao, W. Lu, P. Ryan, W. Smyth, Y. Sun, and L. Yang: *Verifying a border array in linear time.* J. Comb. math. Comb. Comput., 42 2002, pp. 223–236.
10. C. Hohlweg and C. Reutenauer: *Lyndon words, permutations and trees.* Theor. Comput. Sci., 307(1) 2003, pp. 173–178.
11. T. I, S. Inenaga, H. Bannai, and M. Takeda: *Inferring strings from suffix trees and links on a binary alphabet.* Discrete Applied Mathematics, 163, Part 3 2014, pp. 316 – 325, Stringology Algorithms.
12. J. Karkkainen, M. Piatkowski, and S. J. Puglisi: *String inference from longest-common-prefix array*, in Proc. ICALP, 2017, to appear.
13. M. Lothaire: *Combinatorics on Words*, Addison-Wesley, 1983.
14. R. C. Lyndon: *On Burnside's problem.* Transactions of the American Mathematical Society, 77 1954, pp. 202–215.
15. U. Manber and G. Myers: *Suffix arrays: A new method for on-line string searches.* SIAM Journal on Computing, 22(5) 1993, pp. 935–948.

16. W. Matsubara, A. Ishino, and A. Shinohara: *Inferring strings from runs*, in Proceedings of the Prague Stringology Conference 2010, Prague, Czech Republic, August 30 - September 1, 2010, 2010, pp. 150–160.
17. T. A. Starikovskaya and H. W. Vildhøj: *A suffix tree or not a suffix tree?* J. Discrete Algorithms, 32 2015, pp. 14–23.