On the Size of Lempel-Ziv and Lyndon Factorizations*

Juha Kärkkäinen[1], Dominik Kempa[2], Yuto Nakashima[3], Simon J. Puglisi[4], and Arseny M. Shur[5]

1 **Helsinki Institute for Information Technology (HIIT), Helsinki, Finland; and Department of Computer Science, University of Helsinki, Helsinki, Finland**
juha.karkkainen@cs.helsinki.fi
2 **Helsinki Institute for Information Technology (HIIT), Helsinki, Finland; and Department of Computer Science, University of Helsinki, Helsinki, Finland**
dominik.kempa@cs.helsinki.fi
3 **Department of Informatics, Kyushu University, Fukuoka, Japan; and Japan Society for the Promotion of Science, Japan**
yuto.nakashima@inf.kyushu-u.ac.jp
4 **Helsinki Institute for Information Technology (HIIT), Helsinki, Finland; and Department of Computer Science, University of Helsinki, Helsinki, Finland**
simon.puglisi@cs.helsinki.fi
5 **Dept. of Algebra and Discrete Mathematics, Ural Federal University, Ekaterinburg, Russia**
arseny.shur@urfu.ru

Abstract

Lyndon factorization and Lempel-Ziv (LZ) factorization are both important tools for analysing the structure and complexity of strings, but their combinatorial structure is very different. In this paper, we establish the first direct connection between the two by showing that while the Lyndon factorization can be bigger than the non-overlapping LZ factorization (which we demonstrate by describing a new, non-trivial family of strings) it is always less than twice the size.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Pattern Matching, G.2.1 [Combinatorics] Combinatorial Algorithms

Keywords and phrases Lempel-Ziv factorization, Lempel-Ziv parsing, LZ, Lyndon word, Lyndon factorization, Standard factorization

Digital Object Identifier 10.4230/LIPIcs.STACS.2017.45

1 Introduction

Given a string (or word) x, a *factorization* of x partitions x into substrings $f_1, f_2, \ldots f_t$, such that $x = f_1 f_2 \ldots f_t$. In the past 50 years or so, dozens of string factorizations have been studied, some purely out of combinatorial interest (e.g. [12, 1, 22, 2]) and others because the internal structure that they reveal allows the design of efficient string processing algorithms. Perhaps the two most important factorizations in string processing are the Lempel-Ziv (LZ) factorization [26] and the Lyndon factorization[1] [10].

* This research was partially supported by the Academy of Finland through grant 294143 and by the RFBR grant 16-01-00795.

[1] Also known as the Standard factorization.

The LZ factorization has its origins in data compression, and is still used in popular file compressors[2] and as part of larger software systems (see, e.g., [7, 16] and references therein). More recently it has been used in the design of compressed data structures for indexed pattern matching [13] and other problems [5]. Each factor f_i in the LZ factorization must be as long as possible and must be either the first occurrence of a letter in x or occur in $f_1 \dots f_{i-1}$.[3] The Lyndon factorization, on the other hand, was first studied in the context of combinatorics on words [20, Sect. 5], and later found use in algorithms; for example, in a bijective variant of the Burrows-Wheeler transform [14, 18], in suffix sorting [21] and in repetition detection [4]. Each factor f_i in the Lyndon factorization must be a Lyndon word: a string that is lexicographically smaller than all its proper suffixes; and the factors must be lexicographically non-increasing. Lyndon words themselves have deep combinatorial properties [20] and have wide application [6, 9, 11, 15, 18, 19, 20, 23].

For some problems each factorization (Lempel-Ziv or Lyndon) leads to quite different solutions. Perhaps the best known example of this is the computation of all the maximal repetitions – also known as the "runs" – in a string. In 1999 Kolpakov and Kucherov proved that $\rho(n)$, the number of runs in a string of length n, is $O(n)$, and showed how to exploit the structure of the LZ factorization to compute all the runs in linear time [17]. Much more recently, Bannai et al. [3] used properties of the Lyndon factorization to obtain a much simpler constructive proof that $\rho(n) < n$. This later result also leads to a straight-forward linear-time algorithm for computing the runs from the Lyndon factorization [4].

Our overarching motivation in this paper is to obtain a deeper understanding of how these two fundamental factorizations – Lempel-Ziv and Lyndon – relate. Toward this aim, we ask: *by how much can the sizes of the factorizations of the same word differ*? Here the size of the Lempel-Ziv factorization $s = p_1 \cdots p_z$ is z and the size of the Lyndon factorization $s = f_1^{e_1} \cdots f_m^{e_m}$, where each e_i is positive and each f_i is lexicographically strictly greater than f_{i+1}, is m. For most strings, the number of Lyndon factors is much smaller. Indeed, any string has a rotation with a Lyndon factorization of size one. So the actual question is how big can m be with respect to z. For a lower bound, we show that there are strings with $m = z + \Theta(\sqrt{z})$. Our main result is the upper bound: the inequality $m < 2z$ holds for all strings. This result improves significantly a previous, indirect bound by I et al. [25], who showed that the number of Lyndon factors cannot be more than the size of the smallest straight line program (SLP). Since the smallest SLP is at most a logarithmic factor bigger than the LZ factorization [24, 8], this establishes an indirect, logarithmic factor bound, which we improve to a constant factor two.

2 Basic Notions

We consider finite strings over an alphabet $\Sigma = \{a_1, \dots, a_n\}$, which is linearly ordered: $a_1 \prec a_2 \prec \cdots \prec a_n$. For strings, we use the array notation: $s = s[1..|s|]$, where $|s|$ stands for the length of s. The empty string ε has length 0. Any pair i, j such that $1 \le i \le j \le |s|$ specifies a *substring* $s[i..j]$ in s. A string u equal to some $s[i..j]$ is a *factor* of s (a *prefix*, if $i = 1$, and a *suffix*, if $j = |s|$). A prefix or suffix of s is called *proper* if it is not equal to s. A factor u may be equal to several substrings of s, referred to as *occurrences* of u in s. The occurrences of a given factor u are totally ordered by their positions in s, so we can speak about "leftmost" or "previous" occurrence. By u^k we denote the concatenation of k copies of string u. If $k = 0$ we define $u^k = \varepsilon$.

[2] For example `gzip`, `p7zip`, `lz4`, and `snappy` all have the LZ factorization at their core.
[3] This is the non-overlapping version of the LZ factorization.

A string u over Σ is lexicographically smaller or equal than a string v (denoted by $u \preceq v$) if either u is a prefix of v or $u = xaw_1$, $v = xbw_2$ for some strings x, w_1, w_2 and some letters $a \prec b$. In the latter case, we refer to this occurrence of a (resp., of b) as the *mismatch* of u with v (resp., of v with u). A string w is called a *Lyndon word* if w is lexicographically smaller than all its non-empty proper suffixes. The *Lyndon factorization* of a string s is its unique (see [10]) factorization $s = f_1^{e_1} \cdots f_m^{e_m}$ such that each f_i is a Lyndon word, $e_i \geq 1$, and $f_i \succ f_{i+1}$ for all $1 \leq i < m$. We call each f_i a *Lyndon factor* of s, and each $F_i = f_i^{e_i}$ a *Lyndon run* of s. The size of the Lyndon factorization is m, the number of distinct Lyndon factors, or equivalently, the number of Lyndon runs.

The *non-overlapping LZ factorization* (see [26]) of a string s is its factorization $s = p_1 \cdots p_z$ built left to right in a greedy way by the following rule: each new factor (also called an *LZ phrase*) p_i is either the leftmost occurrence of a letter in s or the longest prefix of $p_i \cdots p_z$ which occurs in $p_1 \cdots p_{i-1}$.

3 Upper Bound

The aim of this section is to prove the following theorem.

▶ **Theorem 1.** *Every string s having Lyndon factorization $s = f_1^{e_1} \cdots f_m^{e_m}$ and non-overlapping LZ factorization $s = p_1 \cdots p_z$ satisfies $m < 2z$.*

Let us fix an arbitrary string s and relate all notation $(f_i, e_i, F_i, p_i, m, z)$ to s. The main line of the proof is as follows. We identify occurrences of some factors in s that must contain a boundary between two LZ phrases. Non-overlapping occurrences contain different boundaries, so our aim is to prove the existence of more than $m/2$ such occurrences. We start with two basic facts; the first one is obvious.

▶ **Lemma 2.** *For any strings u, v, w_1, w_2, the relation $uw_1 \prec v \prec uw_2$ implies that u is a prefix of v.*

▶ **Lemma 3.** *The inequality $j < i$ implies $f_j \succ F_i$.*

Proof. We prove that $f_j \succ f_i^k$ for any k, arguing by induction on k. The base case $k = 1$ follows from the definitions. Let $f_j \succ f_i^{k-1}$. In the case of mismatch, $f_j \succ f_i^k$ holds trivially. Otherwise, $f_j = f_i^{k-1}x$ for some $x \neq \varepsilon$. If $x = f_i$ or $x \prec f_i$, then $x \prec f_j$, and so f_j is not a Lyndon word. Hence $x \succ f_i$ and thus $f_j = f_i^{k-1}x \succ f_i^k$. Thus, the inductive step holds. ◀

The next lemma locates the leftmost occurrences of the Lyndon runs and their products.

▶ **Lemma 4.** *Let $d \geq 1$ and $1 \leq i \leq m - d + 1$, and assume that $F_iF_{i+1} \cdots F_{i+d-1}$ has an occurrence to the left of the trivial one in s. Then:*
1. *The leftmost occurrence of $F_iF_{i+1} \cdots F_{i+d-1}$ is a prefix of f_j for some $j < i$;*
2. *$F_iF_{i+1} \cdots F_{i+d-1}$ is a prefix of every f_k with $j < k < i$.*

Proof. (1) Let j be the smallest integer such that the leftmost occurrence of $F_iF_{i+1} \cdots F_{i+d-1}$ in s overlaps F_j. Suppose first that the leftmost occurrence of $F_iF_{i+1} \cdots F_{i+d-1}$ is not entirely contained inside a single occurrence of f_j. Then there exists a non-empty suffix u of f_j that is equal to some prefix of one of the factors $f_i, \ldots, f_{i+d-1}$, say $f_{i'}$. We cannot have $u = f_j$ because then $f_j \preceq f_{i'}$ which is impossible since $j < i'$. Thus u must be a proper suffix of f_j. But then $u \preceq f_{i'} \prec f_j$, which contradicts f_j being a Lyndon word.

Suppose then that the leftmost occurrence of $F_iF_{i+1} \cdots F_{i+d-1}$ in s is entirely contained inside f_j but is not its prefix, i.e., $f_j = vF_iF_{i+1} \cdots F_{i+d-1}w$ for some strings $v \neq \varepsilon$ and w.

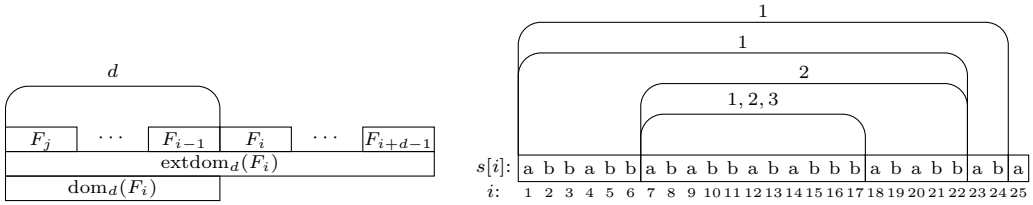

Figure 1 Left: Graphical notation used to illustrate $\mathrm{dom}_d(F_i) = F_j \cdots F_{i-1}$. Also shown is $\mathrm{extdom}_d(F_i) = F_j \cdots F_{i+d-1}$. Right: all non-empty domains for the example string with the Lyndon factorization of size 5. Note that due to Lemma 6 there are no non-trivial intersections between domains.

Since f_j is a Lyndon word we have $f_j \prec F_i F_{i+1} \cdots F_{i+d-1} w$. Consider the position of the mismatch of $F_i F_{i+1} \cdots F_{i+d-1} w$ with f_j. If the mismatch occurs inside $F_i F_{i+1} \cdots F_{i+d-1}$, we can write $f_j = F_i \cdots F_{i'-1} f_{i'}^e x$ where $i \le i' < i + d$, $0 \le e < e_i$, and x is a suffix of f_j that satisfies $x \prec f_{i'} \prec f_j$, which contradicts f_j being a Lyndon word. On the other hand, the mismatch inside w implies that f_j begins with $F_i F_{i+1} \cdots F_{i+d-1}$, contradicting the assumption that the inspected occurrence of $F_i F_{i+1} \cdots F_{i+d-1}$ is the leftmost in s.

(2) We prove this part by induction on d. Let $d = 1$. By Lemma 3 we have $f_j \succ f_k \succ F_i$. Since f_j begins with F_i by statement 1, so does f_k by Lemma 2. Assume now that the claim holds for all $d' < d$. From the inductive assumption F_i and $F_{i+1} \cdots F_{i+d-1}$ are both prefixes of f_k. Let y, y', and z be such that $f_j = F_i F_{i+1} \cdots F_{i+d-1} y$, $f_k = F_{i+1} \cdots F_{i+d-1} y' = F_i z$. We have $j < k$ and thus $f_k \prec f_j$ must hold which, since F_i is a prefix of both f_j and f_k, implies $z \prec F_{i+1} \cdots F_{i+d-1} y$. On the other hand, since f_k is a Lyndon word, we have $f_k = F_{i+1} \cdots F_{i+d-1} y' \prec z$. By Lemma 2, $F_{i+1} \cdots F_{i+d-1} y' \prec z \prec F_{i+1} \cdots F_{i+d-1} y$ implies that $F_{i+1} \cdots F_{i+d-1}$ is a prefix of z or equivalently that $F_i F_{i+1} \cdots F_{i+d-1}$ is a prefix of f_k. ◀

3.1 Domains

Lemma 4 motivates the following definition.

▶ **Definition 5.** Let $d \ge 1$ and $1 \le i \le m - d + 1$. We define the *d-domain* of Lyndon run F_i as the substring $\mathrm{dom}_d(F_i) = F_j F_{j+1} \cdots F_{i-1}$, $j \le i$ of s, where F_j is the Lyndon run (which exists by Lemma 4) starting at the same position as the leftmost occurrence of $F_i F_{i+1} \cdots F_{i+d-1}$ in s. Note that if $F_i F_{i+1} \cdots F_{i+d-1}$ does not have any occurrence to the left of the trivial one then $\mathrm{dom}_d(F_i) = \varepsilon$. The integers d and $i - j$ are called the *order* and *size* of the domain, respectively.

The *extended d-domain* of F_i is the substring $\mathrm{extdom}_d(F_i) = \mathrm{dom}_d(F_i) F_i \cdots F_{i+d-1}$ of s.

Lemma 4 implies two easy properties of domains presented below as Lemma 6. These properties lead to a convenient graphical notation to illustrate domains (see Fig. 1).

▶ **Lemma 6.** *Let* $\mathrm{dom}_d(F_i) = F_j \cdots F_{i-1}$, $j \le i$. *Then:*
- *For any* $d' > d$, $\mathrm{dom}_{d'}(F_i)$ *is a suffix of* $\mathrm{dom}_d(F_i)$*;*
- *For any* $d' \ge 1$, $\mathrm{dom}_{d'}(F_k)$ *is a substring of* $\mathrm{dom}_d(F_i)$ *if* $j \le k < i$.

▶ **Definition 7.** Consider $\mathrm{dom}_d(F_i)$ for some $d \ge 1$, $1 \le i \le m - d + 1$, and let $\alpha = F_i \cdots F_{i+d-1}$. We say that the leftmost occurrence of α in s is *associated* with $\mathrm{dom}_d(F_i)$.

For example, in Fig. 1, $s[7..9]$ is associated with $\mathrm{dom}_2(s[23..24])$; $s[7..17]$ is associated with $\mathrm{dom}_1(s[7..17])$ even though it is not shown, since $\mathrm{dom}_1(s[7..17]) = \varepsilon$. Observe that due

to Lemma 6 this implies that $\mathrm{dom}_d(s[7..17]) = \varepsilon$ for any $d > 1$, and hence for example the substring of s associated with $\mathrm{dom}_2(s[7..17])$ is $s[7..22]$.

A substring $s[i..i+k]$, $k \geq 0$ is said to *contain an LZ phrase boundary* if some phrase of the LZ-factorization of s begins in one of the positions $i, \ldots, i+k$. Clearly, non-overlapping substrings contain different phrase boundaries. Furthermore, if the substring of s does not have any occurrence to the left (in particular, if it is the leftmost occurrence of a single symbol), it contains an LZ phrase boundary, thus we obtain the following easy observation.

▶ **Lemma 8.** *Each substring associated with a domain contains an LZ phrase boundary.*

3.2 Tandem Domains

▶ **Definition 9.** Let $d \geq 1$ and $1 \leq i \leq m-d$. A pair of domains $\mathrm{dom}_{d+1}(F_i)$, $\mathrm{dom}_d(F_{i+1})$ is called a *tandem domain* if $\mathrm{dom}_{d+1}(F_i) \cdot F_i = \mathrm{dom}_d(F_{i+1})$ or, equivalently, if $\mathrm{extdom}_{d+1}(F_i) = \mathrm{extdom}_d(F_{i+1})$. Note that we permit $\mathrm{dom}_{d+1}(F_i) = \varepsilon$.

For example, $\mathrm{dom}_3(s[18..22])$, $\mathrm{dom}_2(s[23..24])$ is a tandem domain in Fig. 1, because we have $\mathrm{extdom}_3(s[18..22]) = \mathrm{extdom}_2(s[23..24]) = s[7..25]$.

▶ **Definition 10.** Let $\mathrm{dom}_{d+1}(F_i)$, $\mathrm{dom}_d(F_{i+1})$ be a tandem domain. Since $F_{i+1} \cdots F_{i+d}$ is a prefix of F_i by Lemma 4, we let $F_i = F_{i+1} \cdots F_{i+d}x$. The leftmost occurrence of $F_i \cdots F_{i+d}$ in s can thus be written as $F_{i+1} \cdots F_{i+d}xF_{i+1} \cdots F_{i+d}$. We say that this particular occurrence of the factor $xF_{i+1} \cdots F_{i+d}$ is *associated* with the tandem domain $\mathrm{dom}_{d+1}(F_i)$, $\mathrm{dom}_d(F_{i+1})$.

▶ Remark. Note that the above definition permits $\mathrm{dom}_{d+1}(F_i) = \varepsilon$. If $\mathrm{dom}_{d+1}(F_i) \neq \varepsilon$, then α, the substring of s associated with $\mathrm{dom}_{d+1}(F_i)$, $\mathrm{dom}_d(F_{i+1})$, is (by Lemma 4) a substring of F_j, where F_j, $j < i$ is the leftmost Lyndon run inside $\mathrm{dom}_{d+1}(F_i)$. Otherwise, α overlaps at least two Lyndon runs. In both cases, however, α is a substring of $\mathrm{extdom}_{d+1}(F_i)$.

▶ **Lemma 11.** *Each substring associated with a tandem domain contains an LZ phrase boundary.*

Proof. Let $\mathrm{dom}_{d+1}(F_i)$, $\mathrm{dom}_d(F_{i+1})$ be a tandem domain and let $u = xF_{i+1} \cdots F_{i+d}$ be the associated substring of s. Suppose to the contrary that u contains no LZ phrase boundaries. Then some LZ-phrase p_t contains u and the letter preceding u. Since we consider a non-overlapping LZ variant, the previous occurrence of p_t in s must be a substring of $p_1 \cdots p_{t-1}$. Note, however, that u is preceded in s by the leftmost occurrence of $F_{i+1} \cdots F_{i+d}$, which is the prefix of F_j (see Definition 10). Thus, the leftmost occurrence of u in s either immediately precedes the associated substring, or overlaps it, or coincides with it. This, however, rules out the possibility that the previous occurrence of p_t occurs in $p_1 \cdots p_{t-1}$, a contradiction. ◀

We say that a tandem domain $\mathrm{dom}_{d+1}(F_i)$, $\mathrm{dom}_d(F_{i+1})$ is *disjoint* from a tandem domain $\mathrm{dom}_{e+1}(F_k)$, $\mathrm{dom}_e(F_{k+1})$ if all $i, i+1, k, k+1$ are different, i.e., $i+1 < k$ or $k+1 < i$.

▶ **Lemma 12.** *Substrings associated with disjoint tandem domains do not overlap each other.*

Proof. Let $\mathrm{dom}_{d+1}(F_i)$, $\mathrm{dom}_d(F_{i+1})$ and $\mathrm{dom}_{e+1}(F_k)$, $\mathrm{dom}_e(F_{k+1})$ be tandem domains called the d-tandem and e-tandem, respectively. Without the loss of generality let $i+1 < k$.

Case 1: $\mathrm{dom}_{d+1}(F_i) \neq \varepsilon$ and $\mathrm{dom}_{e+1}(F_k) \neq \varepsilon$. First observe that if the d-tandem and e-tandem begin with different Lyndon runs, then the associated substrings trivially do not overlap by the above Remark. Assume then that all considered domains start with F_j, $j < i$. By Definition 10 we can write F_j as $F_j = F_{i+1} \cdots F_{i+d}xF_{i+1} \cdots F_{i+d}y$, where $|F_{i+1} \cdots F_{i+d}x| = |F_i|$ and $xF_{i+1} \cdots F_{i+d}$ is the substring of s associated with the d-tandem.

Similarly we have $F_j = F_{k+1} \cdots F_{k+e} x' F_{k+1} \cdots F_{k+e} y'$ where $|F_{k+1} \cdots F_{k+e} x'| = |F_k|$ and $x' F_{k+1} \cdots F_{k+e}$ is the substring of s associated with the e-tandem. However, by Lemma 4, $F_k \cdots F_{k+e}$ is a prefix of F_{i+1} and thus $|F_{k+1} \cdots F_{k+e} x' F_{k+1} \cdots F_{k+e}| \leq |F_{i+1}|$, i.e., the substring of s associated with the e-tandem is inside the prefix F_{i+1} of F_j and thus is on the left of the substring associated with the d-tandem.

Case 2: $\mathrm{dom}_{d+1}(F_i) = F_j \cdots F_{i-1}$, $j < i$, and $\mathrm{dom}_{e+1}(F_k) = \varepsilon$. In this case the substring associated with the e-tandem begins in F_k by the above Remark and thus is on the right of the substring associated with the d-tandem.

Case 3: $\mathrm{dom}_{d+1}(F_i) = \varepsilon$ and $\mathrm{dom}_{e+1}(F_k) = \varepsilon$. This is only possible if $i + d < k$ since otherwise F_k (and thus also $F_{k+1} \cdots F_{k+e}$) occurs in F_i, contradicting $\mathrm{dom}_e(F_{k+1}) = F_k$. Then, $\mathrm{extdom}_{d+1}(F_i)$ does not overlap $\mathrm{extdom}_{e+1}(F_k)$, and the claim holds by above Remark.

Case 4: $\mathrm{dom}_{d+1}(F_i) = \varepsilon$ and $\mathrm{dom}_{e+1}(F_k) = F_j \cdots F_{k-1}$, $j < k$. Then, the substring of s associated with e-tandem is a substring of F_j. If $i > j$, then clearly $\mathrm{extdom}_{d+1}(F_i)$ does not overlap F_j. On the other hand, if $i < j$, it must also hold $i + d < j$ since otherwise F_j (and thus also $F_k \cdots F_{k+e}$) occurs in F_i, contradicting $\mathrm{dom}_{e+1}(F_k) = F_j \cdots F_{k-1}$, and thus again, $\mathrm{extdom}_{d+1}(F_i)$ does not overlap F_j. In both cases the Remark above implies the claim. Finally, if $i = j$, we must also have $i + 1 < k$ from the assumption about the disjointness of d- and e-tandem. By Lemma 4 we can write $F_i = F_{i+1} \cdots F_{i+d} x$, $F_{i+1} = F_k \cdots F_{k+e} x'$ and hence also $F_i \cdots F_{i+d} = F_k \cdots F_{k+e} x' F_{i+2} \cdots F_{i+d} x F_{i+1} \cdots F_{i+d}$. In this decomposition, the substring associated with the e-tandem occurs inside the prefix $F_k \cdots F_{k+e}$, and the substring associated with the d-tandem is the suffix $x F_{i+1} \cdots F_{i+d}$, which proves the claim. ◀

3.3 Groups

We now generalize the concept of tandem domain.

▶ **Definition 13.** Let $d \geq 1$, $2 \leq p \leq m$, and $1 \leq i \leq m - d - p + 2$. A set of p domains $\mathrm{dom}_{d+p-1}(F_i)$, $\mathrm{dom}_{d+p-2}(F_{i+1})$, …, $\mathrm{dom}_d(F_{i+p-1})$ is called a *p-group* if for all $t = 0, \ldots, p-2$ the equality $\mathrm{dom}_{d+p-1-t}(F_{i+t}) \cdot F_{i+t} = \mathrm{dom}_{d+p-2-t}(F_{i+t+1})$ holds or, equivalently, $\mathrm{extdom}_{d+p-1}(F_i) = \ldots = \mathrm{extdom}_d(F_{i+p-1})$. Note that we permit $\mathrm{dom}_{d+p-1}(F_i) = \varepsilon$.

▶ **Lemma 14.** *Substrings associated with tandem domains from the same group do not overlap each other.*

Proof. Consider a p-group, $p \geq 3$ and assume first that $p = 3$. By Lemma 4 we have $F_i = F_{i+1} \cdots F_{i+d+1} x'$ and $F_{i+1} = F_{i+2} \cdots F_{i+d+1} x$ for some words x' and x. We can thus write the leftmost occurrence of $F_i \cdots F_{i+d+1}$ in s as $F_{i+2} \cdots F_{i+d+1} x F_{i+2} \cdots F_{i+d+1} x' F_{i+1} \cdots F_{i+d+1}$. It is easy to see that those occurrences of $x F_{i+2} \cdots F_{i+d+1}$ and $x' F_{i+1} \cdots F_{i+d+1}$ are associated with (resp.) tandem domains $\mathrm{dom}_{d+1}(F_{i+1})$, $\mathrm{dom}_d(F_{i+2})$ and $\mathrm{dom}_{d+2}(F_i)$, $\mathrm{dom}_{d+1}(F_{i+1})$, and thus the claim holds.

For $p > 3$ it suffices to consider all subgroups of size three, in left-to-right order, to verify that the substrings associated with all tandem domains occur in reversed order as a contiguous substring and thus no two substrings overlap each other. ◀

The above Lemma is illustrated in Fig. 2. It also motivates the following definition which generalizes the concept of associated substring from tandem domains to groups.

▶ **Definition 15.** Consider a p-group $\mathrm{dom}_{d+p-1}(F_i)$, $\mathrm{dom}_{d+p-2}(F_{i+1})$, …, $\mathrm{dom}_d(F_{i+p-1})$ for some $p \geq 2$. From Lemma 4, $F_{i+p-1} \cdots F_{i+p+d-2}$ is a prefix of F_i. Thus, the leftmost occurrence of $F_i \cdots F_{i+p+d-2}$ in s can be written as $F_{i+p-1} \cdots F_{i+p+d-2} x F_{i+1} \cdots F_{i+p+d-2}$.

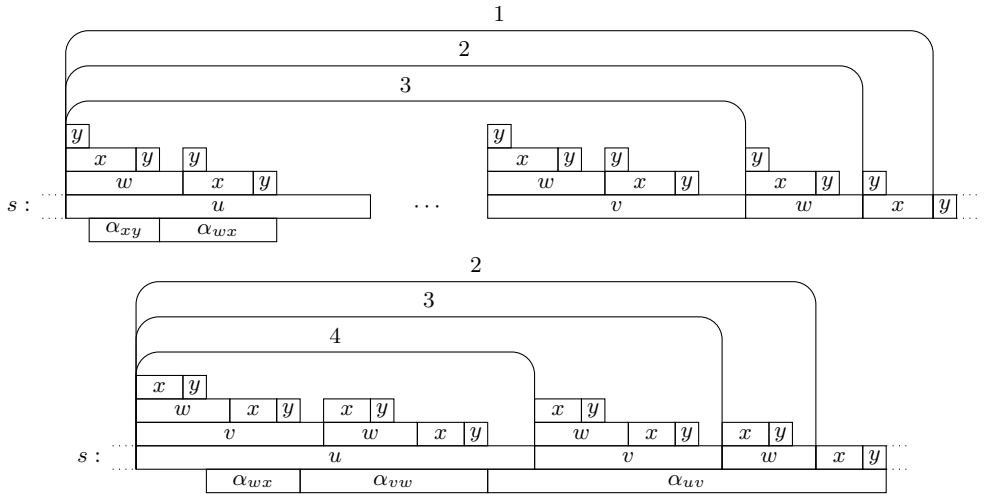

Figure 2 Illustration of Lemma 14. In the examples u, v, w, x, y are Lyndon runs from the Lyndon factorization of s. The top figure shows a 3-group: $\mathrm{dom}_3(w) = u \cdots v$, $\mathrm{dom}_2(x) = u \cdots vw$, $\mathrm{dom}_1(y) = u \cdots vwx$. α_{wx} is a substring associated with the tandem domain $\mathrm{dom}_3(w)$, $\mathrm{dom}_2(x)$, and α_{xy} is a substring associated with the tandem domain $\mathrm{dom}_2(x)$, $\mathrm{dom}_1(y)$. Observe that the substrings associated with tandem domains occur as a contiguous substring and in reverse order (compared to the order of the corresponding tandem domains in s). The bottom figure shows a 4-group: $\mathrm{dom}_5(u) = \varepsilon$, $\mathrm{dom}_4(v) = u$, $\mathrm{dom}_3(w) = uv$, $\mathrm{dom}_2(x) = uvw$ and demonstrates the case when the leftmost domain in a group is empty.

We say that this particular occurrence of the substring $xF_{i+1} \cdots F_{i+p+d-2}$ is *associated* with the p-group $\mathrm{dom}_{d+p-1}(F_i)$, $\mathrm{dom}_{d+p-2}(F_{i+1})$, …, $\mathrm{dom}_d(F_{i+p-1})$.

It is easy to derive a formal proof of the following Lemma from the proof of Lemma 14.

▶ **Lemma 16.** *The substring associated with a p-group is the concatenation, in reverse order, of the $p-1$ substrings associated with the tandem domains belonging to the p-group.*

Our consideration of groups culminates in the next two results.

▶ **Corollary 17.** *The substring associated with a p-group contains at least $p-1$ different LZ phrase boundaries.*

We say that a p-group $\mathrm{dom}_{d+p-1}(F_i)$, …, $\mathrm{dom}_d(F_{i+p-1})$ is *disjoint* from a p'-group $\mathrm{dom}_{d'+p'-1}(F_k)$, …, $\mathrm{dom}_{d'}(F_{k+p'-1})$ if $i + p - 1 < k$ or $k + p' - 1 < i$. By combining Lemma 12 and Lemma 16 we obtain the following fact.

▶ **Lemma 18.** *Substrings associated with disjoint groups do not overlap.*

3.4 Subdomains

The concept of p-group does not easily extend to $p = 1$. If we simply define the 1-group as a single domain and extend the notion of groups to include 1-groups then Lemma 18 no longer holds (e.g. in Fig. 1 the substring associated with tandem domain $\mathrm{dom}_3(s[18..22])$, $\mathrm{dom}_2(s[23..24])$ is $s[10..14]$ and the substring associated with domain $\mathrm{dom}_1(s[7..17])$ is $s[7..17]$). Instead, we introduce a weaker lemma (Lemma 20) that also includes single domains.

▶ **Definition 19.** We say that a domain $\mathrm{dom}_e(F_k)$ is a *subdomain* of a domain $\mathrm{dom}_d(F_i) = F_j \cdots F_{i-1}$, $j \leq i$ if $k = i$ and $e = d$ (i.e., the domain is its own subdomain), or $j \leq k < i$ and $\mathrm{extdom}_e(F_k)$ is a substring of $\mathrm{extdom}_d(F_i)$ (or equivalently, if $k + e \leq i + d$). In other words, F_k has to be one of the Lyndon runs among $F_j, \ldots, F_{i-1}$ and the extended domain of F_k cannot extend (to the right) beyond the extended domain of F_i.

▶ **Lemma 20.** *Consider a tandem domain* $\mathrm{dom}_{e+1}(F_k)$, $\mathrm{dom}_e(F_{k+1})$ *such that* $\mathrm{dom}_{e+1}(F_k)$ *and* $\mathrm{dom}_e(F_{k+1})$ *are subdomains of* $\mathrm{dom}_d(F_i)$*. Then, the substring associated with the tandem domain* $\mathrm{dom}_{e+1}(F_k)$, $\mathrm{dom}_e(F_{k+1})$ *does not overlap the substring associated with* $\mathrm{dom}_d(F_i)$*.*

Proof. First, observe that in order for a tandem domain consisting of two subdomains to exist, $\mathrm{dom}_d(F_i)$ has to be non-empty. Thus, let $\mathrm{dom}_d(F_i) = F_j \cdots F_{i-1}$ for some $j < i$. This implies (Lemma 4) that the substring associated with $\mathrm{dom}_d(F_i)$ is a prefix of F_j.

Assume first that $\mathrm{dom}_{e+1}(F_k) = F_{j'} \cdots F_{k-1}$, $j < j' \leq k$. The substring associated with the tandem domain is a substring of $\mathrm{extdom}_{e+1}(F_k)$ thus it trivially does not overlap F_j.

Assume then that $\mathrm{dom}_{e+1}(F_k) = F_j \cdots F_{k-1}$. If $k+1 < i$ then by Lemma 4, $F_i \cdots F_{i+d-1}$ is a prefix of F_{k+1}. By Definition 10 the leftmost occurrence of $F_k \cdots F_{k+e}$ in s can be written as $F_{k+1} \cdots F_{k+e} x F_{k+1} \cdots F_{k+e}$. Thus clearly the leftmost occurrence of $F_i \cdots F_{i+d-1}$ (associated with $\mathrm{dom}_k(F_i)$) occurs in a prefix F_{k+1} not overlapped by $xF_{k+1} \cdots F_{k+e}$ (which is a substring associated with the tandem domain).

The remaining case is when $k + 1 = i$. Then by Definition 19 we must have $e = d$ and again the claim holds easily from Definition 10. ◀

For any domain $\mathrm{dom}_d(F_i) = F_j \cdots F_{i-1}, j < i$ we define the set of *canonical subdomains* as follows. Consider the following procedure. Initialize the set of canonical subdomains to contain $\mathrm{dom}_d(F_i)$. Then initialize $\delta = d$ and start scanning the Lyndon runs $F_j, \ldots, F_{i-1}$ right-to-left. When scanning F_t we check if $\mathrm{dom}_{\delta+1}(F_t) = F_j \cdots F_{t-1}$.

- If yes, we include $\mathrm{dom}_{\delta+1}(F_t)$ into the set, increment δ and continue scanning from F_{t-1}.
- Otherwise, i.e., if $\mathrm{dom}_{\delta+1}(F_t) = F_{j'} \cdots F_{t-1}$ for some $j' > j$, we include the domain $\mathrm{dom}_{\delta+1}(F_t)$ into the set. Then we set $\delta = 0$ and continue scanning from $F_{j'-1}$. All domains that were included into the set of canonical subdomains in this case are called *loose* subdomains.

See Fig. 3 for an example. The above procedure simply greedily constructs groups of domains, and whenever the candidate for the next domain in the current group does not have a domain that starts with F_j, we terminate the current group, add the loose subdomain into the set and continue building groups starting with the next Lyndon run outside the (just included) loose subdomain.

Note that the current group can be terminated when containing just one domain, so it is not a group in this case. Hence we call the resulting sequences of non-loose domains *clusters*, i.e., a cluster is either a single domain, or a p-group, $p \geq 2$. Note also that during the construction we may encounter more than one loose subdomain in a row, so clusters and loose subdomains do not necessarily alternate, but no two clusters occur consecutively.

Finally, observe that the sequence of clusters and loose subdomains always ends with a cluster (possibly of size one) containing $\mathrm{dom}_{d'}(F_j)$ for some d' ($d' = 3$ for the example in Fig. 3), since $\mathrm{dom}_{d'}(F_j) = \varepsilon$ for all d'.

3.5 Proof of the Main Theorem

We are now ready to prove the key Lemma of the proof. Recall that the size of $\mathrm{dom}_d(F_i) = F_j \cdots F_{i-1}$, $j \leq i$ is defined as $i - j$.

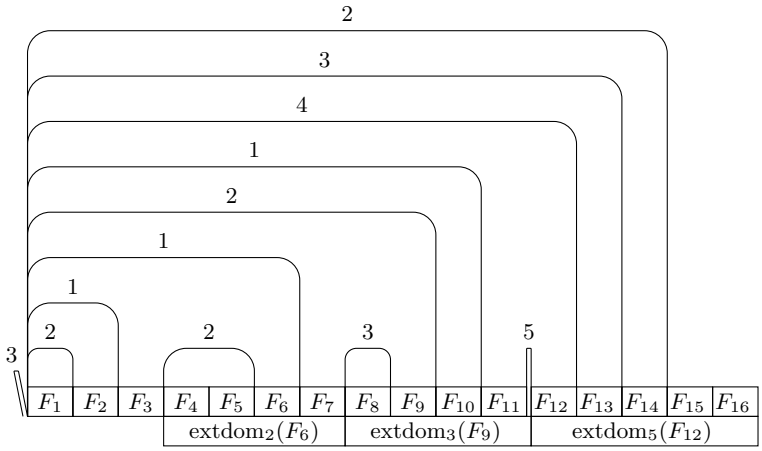

Figure 3 An example showing the set of canonical subdomains of $\mathrm{dom}_2(F_{15})$. Using notation from Lemma 21, the set has $p = 4$ clusters of size (left-to-right): $\ell_1 = 3, \ell_2 = 1, \ell_3 = 2, \ell_4 = 3$, and $t = 3$ loose subdomains: $\mathrm{dom}_2(F_6) = F_4F_5$, $\mathrm{dom}_3(F_9) = F_8$, $\mathrm{dom}_5(F_{12}) = \varepsilon$ of size $k_1 = 2$, $k_2 = 1$, $k_3 = 0$. Note how the extended domains of loose subdomains do not overlap each other. Furthermore, note that $\mathrm{extdom}_2(F_{15}) = F_1 \cdots F_{16}$ can be factorized as $F_1 \cdots F_{\ell_1}$ concatenated with the extended domains. By Corollary 17 and Lemmas 18 and 20, $F_1 \cdots F_{\ell_1}$ contains $1 + \sum_{h=1}^{p}(\ell_h - 1) = 6$ LZ phrase boundaries, while the extended domains $\mathrm{extdom}_2(F_6)$, $\mathrm{extdom}_3(F_9)$, $\mathrm{extdom}_5(F_{12})$ contain $\sum_{h=1}^{t}(\lceil k_h/2 \rceil + 1) = 5$ LZ phrase boundaries by Lemma 21.

▶ **Lemma 21.** *Let* $\mathrm{dom}_d(F_i)$ *be a domain of size* $k \geq 0$. *Then* $\mathrm{extdom}_d(F_i)$ *contains at least* $\lceil k/2 \rceil + 1$ *different LZ phrase boundaries.*

Proof. Let $\mathrm{dom}_d(F_i) = F_j \cdots F_{i-1}$, $j \leq i$ and $k = i - j$. The proof is by induction on k. For $k = 0$, $\mathrm{extdom}_d(F_i)$ is the substring of s associated with $\mathrm{dom}_d(F_i)$ (see Definition 7) and thus by Lemma 8, $\mathrm{extdom}_d(F_i)$ contains at least one LZ phrase boundary.

Let $k > 0$ and assume now that the claim holds for all smaller k. Consider the set $\mathcal{C}_{i,d}$ of canonical subdomains of $\mathrm{dom}_d(F_i)$. If $\mathcal{C}_{i,d}$ contains no loose subdomain, it consists of a single cluster which is a $(k+1)$-group. By Corollary 17, the substring associated with this group contains k phrase boundaries; by Lemma 20, one more boundary is provided by the domain $\mathrm{dom}_d(F_i)$ itself. We have $1 + k \geq 1 + \lceil k/2 \rceil$, which concludes the proof of this case.

For the rest of the proof assume that $\mathcal{C}_{i,d}$ contains $t \geq 1$ loose subdomains, denoted, left to right, by $\mathrm{dom}_{d_1}(F_{i_1}), \ldots, \mathrm{dom}_{d_t}(F_{i_t})$. Note that $d_t > d$. Let k_h be the size of $\mathrm{dom}_{d_h}(F_{i_h})$, $h = 1, \ldots, t$. Further, let $\ell \geq 1$ be the size of the leftmost cluster (the one that contains some domain of F_j). By the construction of the canonical set we have

$$\mathrm{extdom}_d(F_i) = F_j \cdots F_{j+\ell-1}\mathrm{extdom}_{d_1}(F_{i_1})\mathrm{extdom}_{d_2}(F_{i_2}) \cdots \mathrm{extdom}_{d_t}(F_{i_t}). \tag{1}$$

Both clusters and loose subdomains contribute some number of LZ phrase boundaries into their total. The boundaries contributed by clusters are all different by Lemma 18; let S be their number. These boundaries are also different from the boundary inside the substring associated with $\mathrm{dom}_d(F_i)$ by Lemma 20. Furthermore, it is easy to see from the proof of Lemma 20 that all these phrase boundaries are located inside $F_j \cdots F_{j+\ell-1}$. The number of phrase boundaries inside the extended domains of loose subdomains can be estimated by the inductive assumption (by Eq. 1, these external domains do not overlap each other or

$F_j \cdots F_{j+\ell-1}$). So we obtain that $\mathrm{extdom}_d(F_i)$ contains at least

$$1 + \sum_{h=1}^{t} \left(\left\lceil \frac{k_h}{2} \right\rceil + 1 \right) + S \tag{2}$$

different LZ phrase boundaries. Let us evaluate $\sum_{h=1}^{t} k_h$. By the construction, a loose d_h-subdomain is followed by exactly d_h Lyndon runs which are outside loose subdomains; then another loose subdomain follows (cf. Fig. 3). The only exception is the rightmost loose subdomain, which is followed by $d_t - d$ Lyndon runs outside loose subdomains (note that we only count Lyndon runs inside $\mathrm{dom}_d(F_i)$). Then

$$\sum_{h=1}^{t} k_h = k - \ell - \sum_{h=1}^{t} d_h + d. \tag{3}$$

Next we evaluate S. By Corollary 17, a cluster of size r contributes $r-1$ phrase boundaries. Then the leftmost (resp., rightmost) cluster contributes $\ell - 1$ (resp., $d_t - d - 1$) boundaries. Each of the remaining clusters is preceded by a loose d_h-subdomain, where $d_h > 1$, and contributes $d_h - 2$ boundaries. Using Knuth's notation [predicate] for the numerical value (0 or 1) of the predicate in brackets, we can write

$$S = \ell - 1 + \sum_{h=1}^{t} d_h - t - d - \sum_{h=1}^{t-1} [d_h > 1]. \tag{4}$$

Finally, we estimate the number in Eq. 2 using Eq. 3 and Eq. 4:

$$1+\sum_{h=1}^{t} \left(\left\lceil \frac{k_h}{2} \right\rceil + 1 \right) + S \geq 1+t+\frac{k-\ell-\sum_{h=1}^{t} d_h + d}{2} + \ell - 1 + \sum_{h=1}^{t} d_h - t - d - \sum_{h=1}^{t-1} [d_h > 1]$$
$$= \frac{k}{2} + \frac{\ell}{2} + \sum_{h=1}^{t} \frac{d_h}{2} - \frac{d}{2} - \sum_{h=1}^{t-1} [d_h > 1] = \frac{\ell + d_t - d}{2} + \frac{k}{2} + \sum_{h=1}^{t-1} \left(\frac{d_h}{2} - [d_h > 1] \right) \geq 1 + \frac{k}{2}.$$

The obtained lower bound for an integer can be rounded up to $1 + \lceil k/2 \rceil$, as required. ◀

Using the above Lemma we can finally prove the main Theorem.

Proof of Theorem 1. Partition the string s into extended domains as follows: take the string s' such that $s = s' \cdot \mathrm{extdom}_1(F_m)$ and partition s' recursively to get

$$s = \mathrm{extdom}_1(F_{i_1}) \cdots \mathrm{extdom}_1(F_{i_t}), \text{ where } i_t = m.$$

By Lemma 21, each extended domain $\mathrm{extdom}_1(F_{i_h})$ contains at least $\lceil k_h/2 \rceil + 1$ phrase boundaries, where k_h is the size of the domain $\mathrm{dom}_1(F_{i_h})$. Clearly, $\sum_{h=1}^{t} k_h = m - t$; hence the total number z of the boundaries satisfies

$$z \geq \sum_{h=1}^{t} \left(\left\lceil \frac{k_h}{2} \right\rceil + 1 \right) \geq \left\lceil \frac{m-t}{2} \right\rceil + t = \left\lceil \frac{m+t}{2} \right\rceil > \frac{m}{2},$$

as required. ◀

4 Lower Bound

The upper bound on the number of factors in the Lyndon factorization of a string, given in of Theorem 1, is supported by the following lower bound. Consider a string $s_k = B_0 \cdots B_k a$, $k \geq 0$, where:

$$B_0 = b,$$
$$B_1 = ab,$$
$$B_2 = a^2baba^2b,$$
$$\cdots$$
$$B_k = (a^k ba^1 b) \cdots (a^k ba^{k-1} b) a^k b.$$

For example, $s_3 = (b)(ab)(a^2baba^2b)(a^3baba^3ba^2ba^3b)(a)$.

▶ **Theorem 22.** *Let $f_1 \cdots f_{m_k}$ and $p_1 \cdots p_{z_k}$ be the Lyndon factorization and the non-overlapping LZ factorization of the string s_k, $k \geq 2$. Then $m_k = k^2/2 + k/2 + 2$, $z_k = k^2/2 - k/2 + 4$, and thus $m_k = z_k + \Theta(\sqrt{z_k})$.*

Proof. First we count Lyndon factors. All factors will be different, so their number coincides with the number of Lyndon runs. By the definition of Lyndon factorization, the block B_i $(0 < i \leq k)$ is factorized into i Lyndon factors:

$$B_i = a^i ba^1 b \cdot a^i ba^2 b \cdots a^i ba^{i-1} b \cdot a^i b. \tag{5}$$

For any suffix u of $B_0 \cdots B_{i-1}$ and any prefix v of B_i, $u \succ v$ holds since a^i is a prefix of B_i and this is the leftmost occurrence of a^i. Thus there is no Lyndon word that begins in $B_0 \cdots B_{i-1}$ and ends in B_i. This implies that the factorization of s_k is the concatenation of the first b, then k factorizations Eq. 5, and the final a, $k^2/2 + k/2 + 2$ factors in total.

Let $LZ(s_k)$ denote the LZ factorization of s_k. The size of $LZ(s_2) = b \cdot a \cdot ba \cdot aba \cdot baaba$ is 5. For $k \geq 3$, we prove by induction that

$$LZ(s_k) = LZ(s_{k-1}) \cdot a^{k-1}baba^{k-1} \cdot aba^2ba^{k-1} \cdots aba^{k-2}ba^{k-1} \cdot aba^{k-1}ba^k ba. \tag{6}$$

For $k = 3$ we have $LZ(s_3) = LZ(s_2) \cdot aababaa \cdot abaabaaaba$ and thus the claim holds. If $k > 3$, by the inductive hypothesis the last phrase in $LZ(s_{k-1})$ is $p = aba^{k-2}ba^{k-1}ba$. The factor p has only one previous occurrence: it occurs at the boundary between B_{k-2} and B_{k-1}, followed by b. So, p remains a phrase in $LZ(s_k)$. Each of subsequent $k-2$ phrases of Eq. 6 also has a single previous occurrence (inside B_{k-1}), and this occurrence is followed by b because B_{k-1} has no factor a^k. Thus, Eq. 6 correctly represents $LZ(s_k)$. Direct computation now gives $z_k = k^2/2 - k/2 + 4$. ◀

References

1 Golnaz Badkobeh, Hideo Bannai, Keisuke Goto, Tomohiro I, Costas S. Iliopoulos, Shunsuke Inenaga, Simon J. Puglisi, and Shiho Sugimoto. Closed factorization. *Discrete Appl. Math.*, 212:23–29, 2016.

2 Hideo Bannai, Travis Gagie, Shunsuke Inenaga, Juha Kärkkäinen, Dominik Kempa, Marcin Piątkowski, Simon J. Puglisi, and Shiho Sugimoto. Diverse palindromic factorization is NP-complete. In *Proceedings of the 19th International Conference on Developments in Language Theory (DLT)*, pages 85–96. Springer, 2015.

3 Hideo Bannai, Tomohiro I, Shunsuke Inenaga, Yuto Nakashima, Masayuki Takeda, and Kazuya Tsuruta. The runs theorem. *arXiv*, abs/1406.0263, 2014.

4 Hideo Bannai, Tomohiro I, Shunsuke Inenaga, Yuto Nakashima, Masayuki Takeda, and Kazuya Tsuruta. A new characterization of maximal repetitions by Lyndon trees. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 562–571. SIAM, 2015.

5 Djamal Belazzougui, Travis Gagie, Pawel Gawrychowski, Juha Kärkkäinen, Alberto Ordóñez Pereira, Simon J. Puglisi, and Yasuo Tabei. Queries on LZ-bounded encodings. In *Proceedings of the 2015 Data Compression Conference (DCC)*, pages 83–92. IEEE, 2015.

6 Srecko Brlek, Jacques-Olivier Lachaud, Xavier Provençal, and Christophe Reutenauer. Lyndon + Christoffel = digitally convex. *Pattern Recogn.*, 42(10):2239–2246, 2009.

7 Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Michael Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A distributed storage system for structured data. *ACM Trans. Comput. Syst.*, 26(2), 2008.

8 Moses Charikar, Eric Lehman, Ding Liu, Rina Panigrahy, Manoj Prabhakaran, Amit Sahai, and Abhi Shelat. The smallest grammar problem. *IEEE Trans. Information Theory*, 51(7):2554–2576, 2005.

9 Marc Chemillier. Periodic musical sequences and Lyndon words. *Soft Comput.*, 8(9):611–616, 2004.

10 Kuo-Tsai Chen, Ralph H. Fox, and Roger C. Lyndon. Free differential calculus, IV. The quotient groups of the lower central series. *Ann. Math.*, 68:81–95, 1958.

11 Yoann Dieudonné and Franck Petit. Circle formation of weak robots and Lyndon words. *Inf. Process. Lett.*, 101(4):156–162, 2007.

12 Gabriele Fici, Travis Gagie, Juha Kärkkäinen, and Dominik Kempa. A subquadratic algorithm for minimum palindromic factorization. *J. Discrete Algorithms*, 28:41–48, 2014.

13 Travis Gagie, Pawel Gawrychowski, Juha Kärkkäinen, Yakov Nekrich, and Simon J. Puglisi. LZ77-based self-indexing with faster pattern matching. In *Proceedings of the 11th Latin American Theoretical Informatics Symposium (LATIN)*, pages 731–742. Springer, 2014.

14 Joseph Yossi Gil and David Allen Scott. A bijective string sorting transform. *arXiv*, abs/1201.3077, 2012.

15 David Hill, George Melvin, and Damien Mondragon. Representations of quiver Hecke algebras via Lyndon bases. *J. Pure Appl. Algebr.*, 216:1052–1079, 2012.

16 Christopher Hoobin, Simon J. Puglisi, and Justin Zobel. Relative Lempel-Ziv factorization for efficient storage and retrieval of web collections. *PVLDB*, 5(3):265–273, 2011.

17 Roman M. Kolpakov and Gregory Kucherov. Finding maximal repetitions in a word in linear time. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 596–604. IEEE, 1999.

18 Manfred Kufleitner. On bijective variants of the Burrows-Wheeler transform. In *Proceedings of the 2009 Prague Stringology Conference (PSC)*, pages 65–79. Czech Technical University in Prague, 2009.

19 Pierre Lalonde and Arun Ram. Standard Lyndon bases of Lie algebras and enveloping algebras. *Transactions of the American Mathematical Society*, 347:1821–1830, 1995.

20 M. Lothaire. *Combinatorics on Words.* Cambridge University Press, 1997.

21 Sabrina Mantaci, Antonio Restivo, Giovanna Rosone, and Marinella Sciortino. Suffix array and Lyndon factorization of a text. *J. Discrete Algorithms*, 28:2–8, 2014.

22 Yoshiaki Matsuoka, Shunsuke Inenaga, Hideo Bannai, Masayuki Takeda, and Florin Manea. Factorizing a string into squares in linear time. In *Proceedings of the 27th Annual Symposium on Combinatorial Pattern Matching (CPM)*, pages 27:1–27:12. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016.

23 Marcin Mucha. Lyndon words and short superstrings. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 958–972. SIAM, 2013.

24 Wojciech Rytter. Application of Lempel-Ziv factorization to the approximation of grammar-based compression. *Theor. Comput. Sci.*, 302(1-3):211–222, 2003.

25 I Tomohiro, Yuto Nakashima, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Faster Lyndon factorization algorithms for SLP and LZ78 compressed text. *Theor. Comput. Sci.*, 656:215–224, 2016.

26 Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Trans. Inf. Theory*, 23(3):337–343, 1977.