Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Tighter bounds for the sum of irreducible LCP values ☆

Juha Kärkkäinen [a,*], Dominik Kempa [a], Marcin Piątkowski [b]

[a] Helsinki Institute of Information Technology (HIIT) and Department of Computer Science, University of Helsinki, Finland
[b] Faculty of Mathematics and Computer Science, Nicolaus Copernicus University, Toruń, Poland

ARTICLE INFO

Article history:
Received 16 October 2015
Accepted 3 December 2015
Available online xxxx

Keywords:
LCP array
Irreducible LCP values
Suffix array
Burrows–Wheeler transform

ABSTRACT

The suffix array is frequently augmented with the longest-common-prefix (LCP) array that stores the lengths of the longest common prefixes between lexicographically adjacent suffixes of a text. While the sum of the values in the LCP array can be $\Omega(n^2)$ for a text of length n, the sum of so-called irreducible LCP values was shown to be $\mathcal{O}(n \lg n)$ a few years ago. In this paper, we improve the bound to $\mathcal{O}(n \lg r)$, where $r \leq n$ is the number of runs in the Burrows–Wheeler transform of the text. We also show that our bound is tight up to lower order terms (unlike the previous bound). Our results and the techniques used in proving them provide new insights into the combinatorics of text indexing and compression, and have immediate applications to LCP array construction algorithms.

1. Introduction

The suffix array [11,5], a lexicographically sorted array of the suffixes of a text, is the most important data structure in modern string processing. Modern text books spend dozens of pages in describing applications of suffix arrays, see e.g. [15]. In many of those applications, the suffix array needs to be augmented with the longest-common-prefix (LCP) array, which stores the lengths of the longest common prefixes between lexicographically adjacent suffixes (see for instance [1,15]).

A closely related array is the Burrows–Wheeler transform (BWT) [2], which stores the characters preceding each suffix in the lexicographical order of the suffixes. The BWT was designed for text compression and is at the heart of many compressed text indexes [14]. If a text is highly repetitive (and thus highly compressible), its BWT tends to contain long runs of the same character. For example, for any string x and positive integer k, x and x^k have the same number of BWT runs [10]. Thus the number of BWT runs is a rough measure of the (in)compressibility of the text.

An entry LCP[i] in the LCP array is called reducible if BWT[i] = BWT[$i-1$], and *irreducible* otherwise [8]. This distinction between two types of LCP entries is important because of three key properties. First, given all the irreducible LCP values, the reducible LCP values are easy to compute in linear time. This property has been utilized in several LCP array construction algorithms [13,8,19,4,16,7,18]. Second, the number of irreducible values is one less than the number of BWT runs and thus small for repetitive texts. This means that many of the above mentioned algorithms run faster and/or use less space for repetitive texts. There is also a compressed representation of the LCP array based on this property [19].

☆ A preliminary version of this paper appeared in Proceedings of the 26th Annual Symposium on Combinatorial Pattern Matching (CPM), 2015.

* Corresponding author.
E-mail addresses: juha.karkkainen@cs.helsinki.fi (J. Kärkkäinen), dominik.kempa@cs.helsinki.fi (D. Kempa), marcin.piatkowski@mat.umk.pl (M. Piątkowski).

http://dx.doi.org/10.1016/j.tcs.2015.12.009
0304-3975/

The third key property, an upper bound on the sum of the irreducible LCP values, is the topic of this paper. Kärkkäinen et al. [8] showed that the sum is $\mathcal{O}(n \lg n)$ for[1] all texts of length n. This is in contrast to the sum of *all* LCP values, which is $\mathcal{O}(n \lg n)$ for random (non-repetitive) texts but rises to $\Theta(n^2)$ for repetitive texts. For example, if x is a string without long repetitions, the LCP array of x^2 contains about $n/4$ values larger or equal to $n/4$ but only one of those values is irreducible. There may still be $\Theta(n)$ irreducible values but their sum is only $\mathcal{O}(n \lg n)$. Because of this property, computing the irreducible LCP values using brute force comparisons needs just $\mathcal{O}(n \lg n)$ time, which is utilized in many LCP array construction algorithms [8,19,7,18].

In this paper, we improve the upper bound to $n \lg r + \mathcal{O}(n)$, where r is the number of BWT runs. This increases the contrast between the sum of irreducible LCP values and the sum of all LCP values as the former actually decreases as the text gets more repetitive. The new bound immediately improves the time or work complexity bounds for the algorithms in [8,19,18] by a factor $\Theta(\log_r n)$.

Furthermore, we show the tightness of the bound by constructing an infinite family of strings with the irreducible LCP sum of $n \lg r - \mathcal{O}(n)$. More precisely, we show that for any positive integers j and k there exists a text of length $n = j \cdot 2^k$ with $r = 2^k - o(2^k)$ runs such that the sum of irreducible values is $n \lg r - \mathcal{O}(n)$. These bounds are an improvement even in the case of $r \approx n$ since the gap between the bounds is reduced to a lower order term while the previous bounds in [8], $2n \lg n$ and $\frac{1}{2} n \lg n$, were separated by a factor of four.

Our proofs are derived in a more general setting where the text can be a multiset of strings instead of a single string, closely related to the extended BWT introduced in [12]. Under this setting, the inverse Burrows–Wheeler transform is a total function which leads to cleaner combinatorics. In particular, we obtain upper and lower bounds for the irreducible LCP sum that match exactly; the small gap between the bounds mentioned above arises only when restricting the text to be a single string.

The paper is organized as follows. After introducing basic concepts and notation is Section 2, we define the generalized forms of the main data structures, the suffix array, the LCP array and the BWT in Section 3. In Section 4, we show how the irreducible sum can be computed using the reverse suffix tree, and based on this, we derive the upper bounds in Sections 5 and 6. Finally, in Sections 7 and 8, we construct the string family that proves the lower bound. For both upper and lower bounds, we first obtain a bound as a function of n (Sections 5 and 7) and then refine the analysis to obtain a bound as a function of both n and r (Sections 6 and 8).

2. Preliminaries

By $\mathcal{A}$ we denote a finite ordered set, called the *alphabet*. Elements of the alphabet are called *letters*. A finite *word* over $\mathcal{A}$ is a finite sequence of letters $w = a_0 a_1 \dots a_{n-1}$. The length of a word w is defined as the number of its letters and denoted by $|w|$. An empty sequence of letters, called the *empty word*, is denoted by ε. The set of all finite words over $\mathcal{A}$ is denoted by $\mathcal{A}^*$ and the set of all non-empty words over $\mathcal{A}$ by $\mathcal{A}^+ = \mathcal{A}^* \setminus \{\varepsilon\}$.

For two words $x = a_0 a_1 \dots a_{m-1}$ and $y = b_0 b_1 \dots b_{n-1}$, their *concatenation* is $xy = x \cdot y = a_0 a_1 \dots a_{m-1} b_0 b_1 \dots b_{n-1}$. For a word w and an integer $k \geq 1$, we use w^k to denote the concatenation of k copies of w, also called a *power* of w. A word w is *primitive* if w is not a power of some other word. The *root* of a word w is defined as the shortest word $u = \mathrm{root}(w)$ such that $w = u^k$ for some $k \geq 1$.

A word u is a *factor* of a word w if there exist words x and y such that $w = xuy$. Moreover, u is a *prefix* (resp. a *suffix*) of w if $x = \varepsilon$ (resp. $y = \varepsilon$). By $\mathrm{lcp}(u, v)$ we denote the length of the *longest common prefix* of u and v. For a word $w = a_0 \dots a_{n-1}$ and $i, j \in [0..n)$ by $w[i..j]$ we denote its factor of the form $a_i a_{i+1} \dots a_j$. A factor/prefix/suffix u of w is *proper* if $u \neq w$. A (multi)set of words W is *prefix-free* if no word in W is a proper prefix of another word in W.

The order on letters of $\mathcal{A}$ can be extended in a natural way into the *lexicographical order* of words. For any two words x and y we have $x < y$ if x is a proper prefix of y or we have $x = uav_1$ and $y = ubv_2$, where $a, b \in \mathcal{A}$ and $a < b$.

Let $a \in \mathcal{A}$ and $x \in \mathcal{A}^*$. We define a *rotation* operator $\sigma : \mathcal{A}^+ \to \mathcal{A}^+$ as $\sigma(a \cdot x) \mapsto x \cdot a$, a *first-letter* operator $\tau : \mathcal{A}^+ \to \mathcal{A}$ as $\tau(a \cdot x) \mapsto a$ and a *reverse* operator $\overline{\ } : \mathcal{A}^* \to \mathcal{A}^*$ as $\overline{\varepsilon} \mapsto \varepsilon$ and $\overline{a \cdot x} \mapsto \overline{x} \cdot a$. We say that a word w_1 is a *conjugate* of a word w_2 if $w_1 = \sigma^k(w_2)$ for some k.

The set of *infinite periodic words* is defined as $(\mathcal{A}^+)^\omega = \{w^\omega : w \in \mathcal{A}^+\}$, where $w^\omega = w \cdot w \cdot \dots$ is the infinite power of w. We extend several of the above operators to infinite periodic words: $\mathrm{root}(w^\omega) = \mathrm{root}(w)$, $\sigma(w^\omega) = \big(\sigma(w)\big)^\omega$, $\tau\big((a \cdot w)^\omega\big) = a$ and $\overline{(w^\omega)} = \big(\overline{w}\big)^\omega$. Some key properties are given below:

- The operators are well defined: if $u^\omega = v^\omega$ for two words u and v, then $\mathrm{root}(u) = \mathrm{root}(v)$, $\sigma(u)^\omega = \sigma(v)^\omega$, $\tau(u) = \tau(v)$, and $(\overline{u})^\omega = (\overline{v})^\omega$.
- The rotation operator σ is in fact a suffix operator for infinite periodic words: $w^\omega = \tau(w^\omega)\sigma(w^\omega)$ for all $w \in \mathcal{A}^+$. However, unlike a suffix operator for finite words, σ has a well defined inverse σ^{-1}.
- The lexicographical ordering of infinite periodic words is not necessarily the same as their roots. For example, with alphabet $\{a < b\}$, $ab < aba$ but $(ab)^\omega > (aba)^\omega$.

[1] Throughout the paper we use lg as a shorthand for $\log_2$.

Please cite this article in press as: J. Kärkkäinen et al., Tighter bounds for the sum of irreducible LCP values, Theoret. Comput. Sci. (2016), http://dx.doi.org/10.1016/j.tcs.2015.12.009