On Suffix Tree Breadth

Golnaz Badkobeh[1,⋆], Juha Kärkkäinen[2], Simon J. Puglisi[2,⋆⋆], and Bella Zhukova[2,⋆⋆]

[1] Department of Computer Science
University of Warwick
Conventry, United Kingdom
g.badkobeh@warwick.ac.uk

[2] Helsinki Institute for Information Technology
Department of Computer Science
University of Helsinki
Helsinki, Finland
{juha.karkkainen,puglisi,bzhukova}@cs.helsinki.fi

Abstract. The suffix tree — the compacted trie of all the suffixes of a string — is the most important and widely-used data structure in string processing. We consider a natural combinatorial question about suffix trees: for a string S of length n, how many nodes $\nu_S(d)$ can there be at (string) depth d in its suffix tree? We prove $\nu(n,d) = \max_{S\in\Sigma^n} \nu_S(d)$ is $O((n/d)\log n)$, and show that this bound is almost tight, describing strings for which $\nu_S(d)$ is $\Omega((n/d)\log(n/d))$.

1 Introduction

The *suffix tree*, T_S, of a string S of n symbols is a compacted trie containing all the suffixes of S. Since its discovery by Weiner 44 years ago [6] — as an optimal solution to the longest common substring problem — the suffix tree has emerged as perhaps the most important abstraction in string processing [1], and now has dozens of applications, most notably in bioinformatics [5].

Consequently, combinatorial properties of suffix trees are of great interest, and have been exploited in various ways to obtain faster construction algorithms, succinct representations, and efficient pattern matching and discovery algorithms.

Our focus in this article is a natural combinatorial question about suffix trees: how many nodes $\nu_S(d)$ can there be at (string) depth d in the suffix tree of a string S? We prove that $\nu(n,d) = \max_{S\in\Sigma^n} \nu_S(d)$ is $O((n/d)\log n)$, and show that this bound is almost tight, describing strings for which $\nu_S(d)$ is $\Omega((n/d)\log(n/d))$.

In the following section we lay down notation and formally define basic concepts. Section 3 and Section 4 deal with the upper bound and lower bound in turn, and we close with a discussion of the results.

⋆ Supported by a Leverhulme Early Career Fellowship.
⋆⋆ Supported by the Academy of Finland via grant 294143.

2 Preliminaries

Throughout we consider a string $S = S[1..n] = S[1]S[2]\ldots S[n]$ of n symbols drawn from an ordered alphabet Σ of size σ. For $i = 1, \ldots, n$ we write $S[i..n]$ to denote the *suffix* of S of length $n - i + 1$, that is $S[i..n] = S[i]S[i+1]\cdots S[n]$. For convenience we will frequently refer to suffix $S[i..n]$ simply as "suffix i".

The suffix tree of S is a compact trie representing all the suffixes of S. Every suffix tree node either represents a suffix or is a branching node. Each branching node represents a string that occurs at least twice in S and has at least two distinct symbols following those occurrences. The string depth — or simply depth — of a node is the length of the string it represents. Figs. 1 and 2 show examples of suffix trees.

The *suffix array* of S, denoted SA, is an array $\mathrm{SA}[1..n]$ which contains a permutation of the integers $1..n$ such that $S[\mathrm{SA}[1]..n] < S[\mathrm{SA}[2]..n] < \cdots < S[\mathrm{SA}[n]..n]$. In other words, $\mathrm{SA}[j] = i$ iff $S[i..n]$ is the j^{th} suffix of S in ascending lexicographical order. We use SA^{-1} to denote the inverse permutation. For convenience, we also define $\mathrm{SA}[0] = n + 1$ to represent the empty suffix.

The *lcp array* $\mathrm{LCP} = \mathrm{LCP}[1..n]$ is an array defined by S and SA. Let $\mathrm{lcp}(i, j)$ denote the length of the longest common prefix of suffixes i and j. For every $j \in 1..n$,

$$\mathrm{LCP}[j] = \mathrm{lcp}(\mathrm{SA}[j-1], \mathrm{SA}[j]),$$

that is, LCP contains the length of the longest common prefix for each pair of lexicographically adjacent suffixes.

The *permuted lcp array* — $\mathrm{PLCP}[1..n]$ — has the same contents as LCP but in a different order. Specifically, for every $j \in 1..n$,

$$\mathrm{PLCP}[\mathrm{SA}[j]] = \mathrm{LCP}[j]. \quad (1)$$

Then $\mathrm{PLCP}[i] = \mathrm{lcp}(i, \phi(i))$ when we define $\phi(i) = \mathrm{SA}[\mathrm{SA}^{-1}[i] - 1]$.

A binary de Bruijn sequence of order k, denoted by β_k, is a binary word of length $2^k + k - 1$ where each of the 2^k words of length k over the binary alphabet appears as a factor exactly once. As an example, $\beta_4 = aaaabaabbababbbbaaa$ is a de Bruijn sequence of order 4, see Fig. 1.

3 Upper Bound

We are interested in the quantity $\nu(n, d)$, which is the maximum number of branching nodes at depth d over any string of length n.

A trivial upper bound on $\nu(n, d)$ — relevant for shallow levels — is $\nu(n, d) \leq \sigma^d$ for strings over an alphabet of size σ. Another easy upper bound is $\nu(n, d) \leq (n - d)/2$, since there are $n - d$ suffixes longer than d and each branching node at depth d must represent a prefix of at least two such suffixes. In particular, $\nu(2^k + k - 1, k - 1) = 2^{k-1}$ since the upper bound is matched by a binary de Bruijn sequence of order k, as shown in Fig. 1.

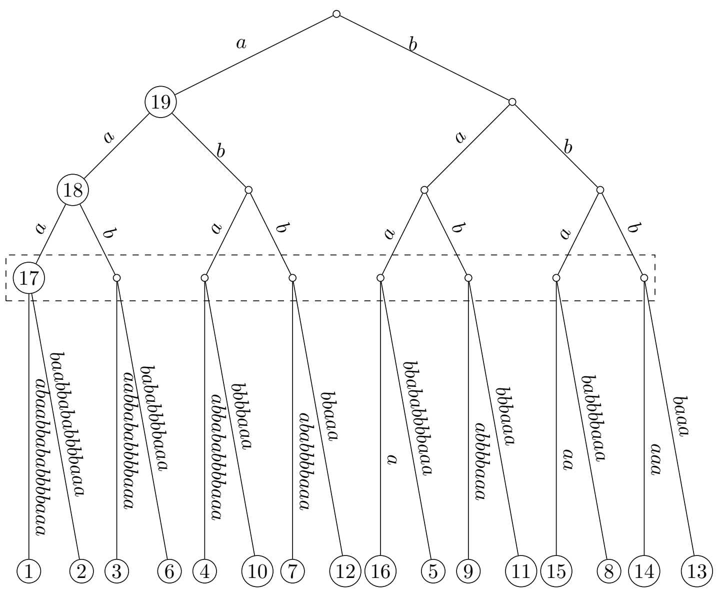

Fig. 1. The suffix tree of string $\beta_4 = aaaabaabbababbbbaaa$, the binary de Bruijn sequence of order 4. The dashed rectangle contains internal nodes at depth 3.

Based on the above, $\nu(n,d)$ increases with d up to level $d \approx \log_\sigma n$ and then starts to go down. The main result of this section is a much tighter upper bound for larger d showing a quick decrease after level $\log n$.

Our upperbound proof makes use of the concept of *irreducible* lcp values, first defined in [3]. We say that $\text{PLCP}[i] = \text{lcp}(i, \phi(i))$ is *reducible* if $S[i–1] = S[\phi(i)–1]$ and *irreducible* otherwise. In particular, it is irreducible if $i = 1$ or $\phi(i) = 1$. Reducible values are easy to compute via the next lemma.

Lemma 1 ([3]). *If* $\text{PLCP}[i]$ *is reducible, then* $\text{PLCP}[i] = \text{PLCP}[i-1] - 1$.

We also need an upper bound on the sum of irreducible lcp values.

Lemma 2 ([3,2]). *The sum of all irreducible lcp values is* $\leq n \log n$.

Theorem 3. *The number of branching nodes at depth* d *in the suffix tree for a string of length* n *is* $\leq (n/d) \log n$.

Proof. Let S be a string with $\nu(n,d)$ branching nodes at depth d in the suffix tree of S. Every such branching node corresponds to one or more values d in the lcp array, each of which in turn corresponds to a position in the PLCP array with value d. In other words, the number of d's in the PLCP array of S is an upper bound on $\nu_S(d)$. Let $i_1, \ldots, i_r$ be the positions of irreducible values in the PCLP array in ascending order, and let $i_{r+1} = n + 1$. Since $i_1 = 1$, the intervals $\text{PLCP}[i_j..i_{j+1} - 1]$, $j = 1..n$, form a partitioning of the PLCP array. Due to

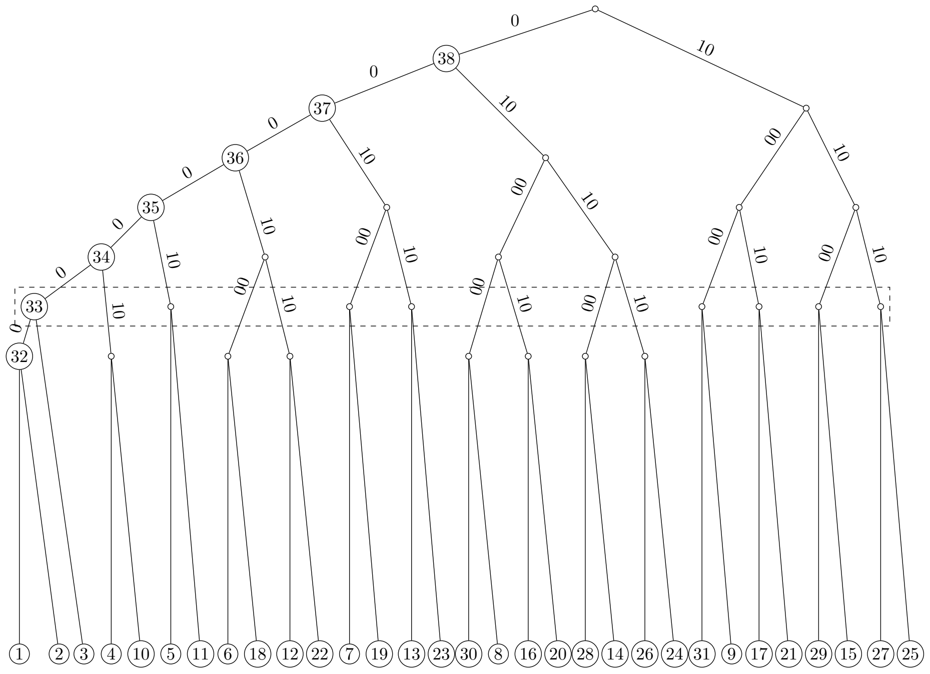

Fig. 2. The suffix tree of string $W_{2,4}$ = 00000000100000101000100010101010000000. The dashed rectangle contains internal nodes at depth 6.

Lemma 1, for every $j = 1..n$, $\mathrm{PLCP}[i_j..i_{j+1}-1]$ contains at most one d and only if $\mathrm{PLCP}[i_j] \geq d$. Therefore, each occurrence of d can be mapped to a unique irreducible lcp value $\geq d$. Using Lemma 2, $d\nu(n,d) \leq n \log n$ so the upper bound follows. □

4 Lower Bound

This section is devoted to proving the following result.

Theorem 4. *For any positive integers $j \geq 1$ and $k \geq 3$, there exists a string of length $n = j(2^k + k - 1)$ such that its suffix tree has $\geq \frac{1}{2}\left(\frac{n}{d} - 1\right)\log\left(\frac{n}{d} - 1\right)$ branching nodes at depth $d = j(k-1)$.*

Proof. Our proof is based on a construction of the following string, $W_{j,k}$. Let β_k be a binary de Bruijn sequence of order k. Clearly, the suffix tree of β_k is full up to depth $k-1$, and has 2^{k-1} nodes at depth $k-1$. Now, let $W_{j,k} = w_j(\beta_k)$ where morphism w_j is the following

$$\begin{cases} w_j(a) = 0^j \\ w_j(b) = 10^{j-1} \end{cases}$$

It is clear that $|W_{j,k}| = n = j(2^k + k - 1)$. Let $m = \nu_{W_{j,k}}(j(k-1))$ denote the number of branching nodes of the suffix tree of string $W_{j,k}$ at depth $d = j(k-1)$. We claim that $m \geq 2^{k-1}$. If both ya and yb occur in β_k for some string y, then both $x0$ and $x1$ occur in $W_{i,j}$ for $x = w_j(y)$. Thus every branching node representing y in the suffix tree of β_k is uniquely mapped to a branching node representing $x = w_j(y)$ in the suffix tree of $W_{i,j}$. Since the suffix tree of β_k has 2^{k-1} branching nodes at depth $k-1$, the claim $m \geq 2^{k-1}$ follows.

What remains is to show the steps for the calculation of the lower bound. Since $m \geq 2^{k-1}$ and $d = j(k-1) = \frac{n(k-1)}{(2^k+k-1)}$, we have $\frac{n}{d} = \frac{2^k}{k-1} + 1 \leq \frac{2m}{k-1} + 1$, which implies

$$m \geq \frac{1}{2}\left(\frac{n}{d} - 1\right)(k-1) \geq \frac{1}{2}\left(\frac{n}{d} - 1\right)\log\left(\frac{2^k}{k-1}\right) = \frac{1}{2}\left(\frac{n}{d} - 1\right)\log\left(\frac{n}{d} - 1\right) .$$

□

5 Discussion

Notice that the lowerbound construction implies $d = \Omega(\log n)$; thus it does not contradict the upper bounds for small d discussed in Section 3.

The upper and lower bounds of Theorems 3 and 4 are asymptotically equal when $d = O(n^{1-\epsilon})$ for any constant $\epsilon > 0$. We conjecture that the lower bound is asymptotically tight even for larger d, but proving a matching upper bound remains an open problem.

Essentially the same bounds hold for all variants and generalizations. We have counted only branching nodes but including leaves (and unary nodes representing suffixes) too would not change much as there can be only one leaf (or unary node) at each level. Similarly, adding a unique terminator symbol to the end of the string adds at most one node per level. Considering a suffix tree of multiple strings (containing all suffixes of all strings) could add more leafs to a level but no more than n/d leafs at a level d; thus the asymptotic results do not change. Another variant considers the string to be cyclic — replacing suffixes with rotations — and even suffix trees for collections of cyclic strings have been considered [2,4]. All the results hold in this case too: the key result for the upper bound, Lemma 2, was proved for collections of cyclic strings [2], and de Bruijn sequences are naturally defined as cyclic strings. Finally, notice that Theorems 3 and 4 hold for any alphabet size.

References

1. Apostolico, A., Crochemore, M., Farach-Colton, M., Galil, Z., Muthukrishnan, S.: 40 years of suffix trees. Commun. ACM 59(4), 66–73 (2016)
2. Kärkkäinen, J., Kempa, D., Piatkowski, M.: Tighter bounds for the sum of irreducible LCP values. Theor. Comput. Sci. 656, 265–278 (2016), https://doi.org/10.1016/j.tcs.2015.12.009

3. Kärkkäinen, J., Manzini, G., Puglisi, S.J.: Permuted longest-common-prefix array. In: Combinatorial Pattern Matching, 20th Annual Symposium, CPM 2009. Lecture Notes in Computer Science, vol. 5577, pp. 181–192. Springer (2009), https://doi.org/10.1007/978-3-642-02441-2_17
4. Kärkkäinen, J., Piatkowski, M., Puglisi, S.J.: String inference from longest-common-prefix array. In: 44th International Colloquium on Automata, Languages, and Programming, ICALP 2017. pp. 62:1–62:14 (2017), https://doi.org/10.4230/LIPIcs.ICALP.2017.62
5. Mäkinen, V., Belazzougui, D., Cunial, F., Tomescu, A.I.: Genome-Scale Algorithm Design: Biological Sequence Analysis in the Era of High-Throughput Sequencing. Cambridge University Press (2015), http://www.genome-scale.info/
6. Weiner, P.: Linear pattern matching algorithms. In: 14th Annual Symposium on Switching and Automata Theory. pp. 1–11. IEEE Computer Society (1973)