Information Processing Letters 125 (2017) 26–29

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Tight lower bounds for the longest common extension problem

Dmitry Kosolobov

University of Helsinki, Helsinki, Finland

ARTICLE INFO

Article history:
Received 6 January 2017
Received in revised form 11 May 2017
Accepted 11 May 2017
Available online 17 May 2017
Communicated by Marcin Pilipczuk

Keywords:
Longest common extension
Data structures
Trade-off
Lower bounds
Cell-probe model

ABSTRACT

The longest common extension problem is to preprocess a given string of length n into a data structure that uses $S(n)$ bits on top of the input and answers in $T(n)$ time the queries $LCE(i, j)$ computing the length of the longest string that occurs at both positions i and j in the input. We prove that the trade-off $S(n)T(n) = \Omega(n \log n)$ holds in the non-uniform cell-probe model provided that the input string is read-only, each letter occupies a separate memory cell, $S(n) = \Omega(n)$, and the size of the input alphabet is at least $2^{8\lceil S(n)/n \rceil}$. It is known that this trade-off is tight.

1. Introduction

Data structures for solving the so-called *longest common extension (LCE)* problem (sometimes referred to as the longest common prefix problem) play the central role in the wide range of string algorithms. In this problem we must preprocess an input string of length n so that one can answer the queries $LCE(i, j)$ computing the length of the longest string that occurs at both positions i and j in the input. Since the existing solutions to this problem often, in practice particularly, constitute a bottleneck either in space or in time of the algorithms relying in their core on the LCE queries, many efforts have been made in the past decades to develop better LCE data structures.

In this paper we prove that the trade-off[1] $S(n)T(n) = \Omega(n \log n)$ holds for any data structure that solves the LCE problem using $S(n)$ bits of space (called *additional space*) on top of the input and $T(n)$ time for the LCE queries, assuming that the input string is read-only, each letter occupies a separate memory cell, and $S(n) = \Omega(n)$ (such space is used in most applications of the LCE problem). For $S(n) = \Omega(n)$, this new trade-off improves by $\log n$ factor the trade-off $S(n)T(n) = \Omega(n)$ established by Bille et al. [2], who used a simple reduction to a lower bound obtained by Brodal et al. [3] for the so-called range minimum queries problem.

Our result is proved in the *cell-probe model* [12], in which the computation is free and time is counted as the number of cells accessed (probed) by the query algorithm. The algorithm is also allowed to be non-uniform, i.e., we can have different algorithms for different sizes n of the input. We assume that each letter of the input string is an integer located in a separate memory cell and each cell can store any integer from the set $\{0, 1, \ldots, n-1\}$. Hence, the maximal size of the input alphabet is n; this is a common assumption justified in, e.g., [4]. However, our main theorem poses a more specific restriction: the size of the input alphabet must be at least $2^{8\lceil S(n)/n \rceil}$. For instance, our trade-off is applicable for constant alphabets if $S(n) = \Theta(n)$, but to apply the trade-off in the case $S(n) = \Theta(n\sqrt{\log n})$, we have to have an alphabet of at least $2^{\Omega(\sqrt{\log n})}$ size.

E-mail address: dkosolobov@mail.ru.

[1] For brevity, log denotes the logarithm with base 2.

http://dx.doi.org/10.1016/j.ipl.2017.05.003
0020-0190/

Overview of LCE data structures. The classical solutions for the LCE problem use $\Theta(n\log n)$ bits of space and $O(1)$ time for queries (e.g., see [5,9]). In [1] Bille et al. presented a RAM data structure that solves the LCE problem using $O(\tau)$ time for queries and $O(\frac{n\log n}{\tau})$ bits of additional space, where τ is a parameter such that $1 \le \tau \le n$. This result shows that our trade-off is tight and cannot be improved. The construction time of this data structure (in $O(\frac{n\log n}{\tau})$ bits of space) is $O(n^{2+\varepsilon})$, which is unacceptably slow. In [15] Tanimura et al. proposed a data structure with significantly better $O(n\tau)$ construction time within the same $O(\frac{n\log n}{\tau})$ bits of additional space but with slightly suboptimal query time $O(\tau \min\{\log\tau, \log\frac{n}{\tau}\})$.

Denote by σ the size of the input alphabet. Recently, Tanimura et al. [16] described a data structure that, for $\sigma \le 2^{o(\log n)}$, uses $o(n\log n)$ bits of additional space and $O(1)$ time for LCE queries thus "surpassing" our trade-off and showing the importance of the condition $\sigma \ge 2^{8\lceil S(n)/n\rceil}$. We believe also that our trade-off does not hold if the algorithm can read $\Omega(\log_\sigma n)$ consecutive letters of the input string in $O(1)$ time packing them in one $\Omega(\log n)$-bit machine word; this model reflects the situation that one can often observe in practice.

All mentioned results consider applications in which the input string is treated as read-only. In practice, however, we usually need a data structure that provides fast access to the string and allows us to answer the LCE queries, but the space occupied by the string itself can be reorganized. The data structure of [6] using this model occupies $O(\frac{n\log n}{\tau})$ bits of additional space and answers LCE queries in $O(\log^* n(\log\frac{n}{\tau} + \tau^{\log 3}/\log_\sigma n))$ time, where τ is a parameter such that $1 \le \tau \le n$ (however, this result still does not break our trade-off). The construction time for this structure (in $O(\frac{n\log n}{\tau})$ bits) is $O(n(\log^* n + \frac{\log n}{\tau} + \frac{\log\tau}{\log_\sigma n}))$. In [14] Prezza described an "in-place" data structure[2] that replaces $n\lceil\log\sigma\rceil$ bits occupied by the input with a data structure that allows to retrieve any substring of length m of the input in optimal $O(\frac{m}{\log_\sigma n})$ time and answers the LCE queries in $O(\log\ell)$ time, where ℓ is the result of the query.[3] For his data structure, Prezza presents only a randomized construction algorithm working in $O(n\log n)$ expected time and $O(n\log n)$ bits of space.

In certain applications the exact accuracy of the LCE queries is less important than construction time, query time, and space. For such applications, several Monte Carlo data structures were developed: their construction algorithm builds with high probability (i.e., with probability $1 - \frac{1}{n^c}$ for any specified constant $c > 0$) a valid data structure answering any LCE query correctly but sometimes can produce a faulty data structure. Prezza [14] described a Monte Carlo version of his "in-place" data structure that answers the LCE queries in $O(\log\ell)$ time and has a construction algorithm working in $O(\frac{n}{\log_\sigma n})$ expected time using the same memory, i.e., also "in-place". Bille et al. [1] presented a Monte Carlo version of their data structure for read-only inputs that answers the LCE queries in $O(\tau)$ time using $O(\frac{n\log n}{\tau})$ bits of additional space and has $O(n\log\frac{n}{\tau})$ construction time (within the same space), where $1 \le \tau \le n$. Gawrychowski and Kociumaka [7, Th. 3.3] described a modification of this Monte Carlo solution for read-only inputs that has the same optimal space and query time bounds but can be constructed in optimal $O(n)$ time.

Recently, several LCE data structures for compressed strings were developed. For a more detailed discussion on this topic, we refer the reader to [10,16] and references therein.

2. Main result

Preliminaries. A *string* s *of length* n over an alphabet Σ is a map $\{0, 1, \dots, n-1\} \mapsto \Sigma$, where n is referred to as the *length of* s, denoted by $|s|$. We write $s[i]$ for the ith letter of s. A string $s[0]s[1]\cdots s[j]$ is a *prefix* of s. For any i and j, the set $\{k \in \mathbb{Z} \colon i \le k \le j\}$ (possibly empty) is denoted by $[i..j]$.

Theorem. *In the non-uniform cell-probe model the trade-off* $S(n)T(n) = \Omega(n\log n)$ *holds for any data structure that solves the LCE problem for a read-only string of length* n *using* $S(n)$ *bits of space and* $T(n)$ *time for queries assuming that each input letter occupies a separate cell, the size of the input alphabet is at least* $2^{8\lceil S(n)/n\rceil}$, *and* $S(n) = \Omega(n)$.

Proof. Without loss of generality, assume that $T(n) \ge 1$. Suppose, for the sake of contradiction, that $S(n)T(n) \notin \Omega(n\log n)$. Then, there is an infinite set N of positive integers such that $\lim_{n\in N} \frac{S(n)T(n)}{n\log n} = 0$. Hence, we obtain $\lim_{n\in N} \frac{S(n)}{n\log n} = 0$. Therefore, there is a positive function $\varepsilon(n)$ such that $S(n) = \varepsilon(n)n\log n$ for $n \in N$ and $\varepsilon(n)$ tends to 0 as $n \to +\infty$.

Let us first construct a family $\mathcal{F}$ of inputs for the subsequent analysis. Define $\sigma = 2^{8\lceil\varepsilon(n)\log n\rceil}$. The input alphabet is $[1..\sigma]$. Note that $\sigma \ge 2^8$ for $n > 1$ and $\sigma = 2^{8\lceil S(n)/n\rceil}$ for $n \in N$. Since $\log\sigma = o(\log n)$ and, consequently, $\sigma < n$ for sufficiently large n, each letter of the alphabet fits in one memory cell. Observe that, since $S(n) = \varepsilon(n)n\log n \le \frac{1}{8}n\log\sigma$ for $n \in N$, we are not able to encode the whole string in $S(n)$ bits and answer the LCE queries without any access to the string.

Denote $k = \lfloor\frac{1}{2}\log_\sigma n\rfloor$. Since $\log\sigma = o(\log n)$, we have $k = \Theta(\frac{\log n}{\log\sigma}) = \omega(1)$. Therefore, $k \ge 1$ for sufficiently large n. Let $s_1, s_2, \dots, s_{\sigma^k}$ denote all strings of length k over the alphabet $[1..\sigma]$. The family $\mathcal{F}$ consists of all strings of the form $s_1 s_2 \cdots s_{\sigma^k} t$, where t is a string of length $n - k\sigma^k$ over the alphabet $[1..\sigma]$. Since $\sigma^k k \le \sqrt{n}\log n$, it is easy to verify that $\frac{1}{2}n \le n - k\sigma^k$ for $n \ge 2^8$. Hence, we obtain $|\mathcal{F}| \ge \sigma^{\frac{1}{2}n} = 2^{\frac{1}{2}n\log\sigma}$ for $n \ge 2^8$. For convenience, we assume hereafter that $\min N \ge 2^8$.

By the pigeonhole principle, there is a subfamily $\mathcal{I} \subseteq \mathcal{F}$ such that $|\mathcal{I}| \ge |\mathcal{F}|/2^{S(n)}$ and, for any strings $s, s' \in \mathcal{I}$,

[2] The data structure uses only negligible $O(\log^2 n)$ bits of space on top of the input.

[3] A similar result in [13] seems to be very practical, but its correctness in the RAM model, where n tends to infinity, relies on a questionable assumption that the natural density of the logarithms of the Mersenne primes is non-zero (this is required to process these primes in constant time with $\Theta(\log n)$-bit machine words).