Journal of Computer and System Sciences 80 (2014) 1245–1253

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Range LCP

Amihood Amir [a,b,1], Alberto Apostolico [c,d,2], Gad M. Landau [e,f,3], Avivit Levy [g,*,4], Moshe Lewenstein [a], Ely Porat [a,5]

[a] *Department of Computer Science, Bar-Ilan University, Ramat-Gan 52900, Israel*
[b] *Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218, United States*
[c] *College of Computing, Georgia Institute of Technology, 801 Atlantic Drive, Atlanta, GA 30318, United States*
[d] *Dipartimento di Ingegneria dell' Informazione, Università di Padova, Via Gradenigo 6/A, 35131 Padova, Italy*
[e] *Department of Computer Science, University of Haifa, Mount Carmel, Haifa 31905, Israel*
[f] *Department of Computer Science and Engineering, Polytechnic Institute of New York University, 6 Metrotech Center, Brooklyn, NY 11201, United States*
[g] *Department of Software Engineering, Shenkar College, 12 Anna Frank, Ramat-Gan, Israel*

ARTICLE INFO

Article history:
Received 29 January 2013
Received in revised form 21 January 2014
Accepted 5 February 2014
Available online 14 February 2014

Keywords:
Data structures
LCP
Pattern matching

ABSTRACT

In this paper, we define the *Range LCP* problem as follows. Preprocess a string S, of length n, to enable efficient solutions of the following query: Given $[i, j]$, $0 < i \leqslant j \leqslant n$, compute $\max_{\ell,k\in[i..j]} LCP(S_\ell, S_k)$, where $LCP(S_\ell, S_k)$ is the length of the longest common prefix of the suffixes of S starting at locations ℓ and k. This is a natural generalization of the classical LCP problem. We provide algorithms with the following complexities:

1. Preprocessing Time: $O(|S|)$, Space: $O(|S|)$, Query Time: $O(|j-i|\log\log n)$.
2. Preprocessing Time: none, Space: $O(|j-i|\log|j-i|)$, Query Time: $O(|j-i|\log|j-i|)$. However, the query just gives the pairs with the longest LCP, *not* the LCP itself.
3. Preprocessing Time: $O(|S|\log^2|S|)$, Space: $O(|S|\log^{1+\varepsilon}|S|)$ for arbitrary small constant ε, Query Time: $O(\log\log|S|)$.

* Corresponding author.
E-mail addresses: amir@cs.biu.ac.il (A. Amir), axa@cc.gatech.edu (A. Apostolico), landau@cs.haifa.ac.il (G.M. Landau), avivitlevy@shenkar.ac.il (A. Levy), moshe.lewenstein@gmail.com (M. Lewenstein), porately@cs.biu.ac.il (E. Porat).

[1] Partly supported by NSF grant CCR-09-04581, ISF grant 347/09, and BSF grant 2008217.
[2] Partly supported by BSF grant 2008217.
[3] Partly supported by the National Science Foundation Award 0904246, Israel Science Foundation grant 347/09, Yahoo, grant No. 2008217 from the United States–Israel Binational Science Foundation (BSF) and DFG.
[4] Partly supported by the Israel Science Foundation grant 347/09.
[5] Partly supported by BSF grant 2006334, ISF grant 1484/08, and Google Award.

http://dx.doi.org/10.1016/j.jcss.2014.02.010
0022-0000/© 2014 Elsevier Inc. All rights reserved.

1. Introduction

The Longest Common Prefix (LCP) has been historically an important tool in Combinatorial Pattern Matching:

1. The connection between Edit Distance and Longest Common Prefix (LCP) calculation has been shown and exploited in the classic Landau–Vishkin paper in 1989 [14]. It was shown in that paper that computing mismatches and LCPs is sufficient for computing the Edit Distance.
2. The LCP is the main tool in various Bioinformatics algorithms for finding maximal repeats in a genomic sequence.
3. The LCP plays an important role in compression. Its computation is required in order to compute the Ziv–Lempel compression, for example [15].

Therefore, the LCP has been amply studied and generalized versions of the problem are of interest. Generalizations of classical problems shed light on the original problem and lead to a better understanding of it, but also enable to develop new, usually stronger, tools for the solution of the problem. Generalization is, therefore, a classical approach in the study of algorithms.

A first natural generalization of the LCP problem is a "range" version. Indeed, "range" versions of the LCP problem were considered in the literature. Cormode and Muthukrishnan [5] consider a version they call the *Interval Longest Common Prefix* (*ILCP*) *Problem*. In that version, the maximum LCP between a given suffix and all suffixes in a given interval, is sought. They provide an algorithm whose preprocessing time is $O(|S|\log^2|S|\log\log|S|)$, and whose query time is $O(\log|S|\log\log|S|)$. This result was then improved by [12] to $O(|S|\log|S|)$ preprocessing time and $O(\log|S|)$ query time. Ilie and Tinta [9] consider a different "range" version, where a set of pairs of suffixes is given as input and the pair that has the maximum LCP is sought.

This paper provides efficient algorithms for a more general version of the Range LCP problem.

Problem definition. The formal definitions of LCP and the range LCP problem are given below.

Definition 1. Let $A = A[1]\cdots A[n]$ and $B = B[1]\cdots B[n]$ be strings over alphabet Σ. The *Longest Common Prefix* (*LCP*) of A and B is the empty string if $A[1] \neq B[1]$. Otherwise it is the string $a_1 \cdots a_k$, $a_i \in \Sigma$, $i = 1, \ldots, k$, $k \leqslant n$, where $A[i] = B[i] = a_i$, $i = 1, \ldots, k$, and $A[k+1] \neq B[k+1]$ or $k = n$.

We abuse notations and sometimes refer to the *length of the LCP*, k, as the LCP. We, thus, denote the length of the LCP of A and B by $LCP(A, B)$.

Given a constant c and string $S = S[1]\cdots S[n]$ over alphabet $\Sigma = \{1, \ldots, n^c\}$, one can preprocess S in linear time in a manner that allows subsequent LCP queries in constant time. I.e., any query of the form $LCP(S_i, S_j)$, where S_i and S_j are the suffixes of S starting at locations i, j of S, respectively, $1 \leqslant i, j, n$, can be computed in constant time. For general alphabets, the preprocessing takes time $O(n\log n)$.

This can be done either via suffix tree construction [19,16,17,6] and Lowest Common Ancestor (LCA) queries [7,3], or suffix array construction [10] and LCP queries [11].

Our problem is formally defined as follows.

Definition 2. The *Range LCP* problem is the following:

INPUT: String $S = S[1]\cdots S[n]$ over alphabet Σ.
Preprocess S in a manner allowing efficient solutions to queries of the form:
QUERY: i, j, $1 \leqslant i \leqslant j \leqslant n$.
Compute $RangeLCP(i, j) = \max_{\ell,k\in[i..j]} LCP(S_\ell, S_k)$, where S_ℓ (resp. S_k) is the suffix of S starting at location ℓ (resp. k), i.e. $S_\ell = S[\ell]S[\ell+1]\cdots S[n]$.

Simple baseline solutions. For the sake of completeness, we describe two straight-forward algorithms, which we henceforth call Algorithm Base1 and Algorithm Base2, to solve the Range LCP problem. They are the baseline to improve. Algorithm Base1 has a linear-time preprocessing but a quadratic Range LCP query time. The algorithm preprocesses the input string S for LCP queries. Then, given an interval $[i, j]$, it computes $LCP(k, \ell)$ for every pair k, ℓ, $i \leqslant k \leqslant \ell \leqslant j$, and chooses the pair with the maximum LCP. As seen in Section 1.1 below, the preprocessing can be accomplished in linear time for alphabets that are fixed polynomials of n and in time $O(n\log n)$ for general alphabets. The query processing has $|j-i|^2$ LCP calls, each one taking a constant time for a total time $O(|j-i|^2)$. Algorithm Base2 uses two-dimensional range-maximum queries, defined as follows.