Range LCP

Amihood Amir[1,2,⋆], Alberto Apostolico[3,4,⋆⋆] Gad M. Landau[5,6,⋆⋆⋆], Avivit Levy[7,8], Moshe Lewenstein[1], and Ely Porat[1,†]

[1] Department of Computer Science, Bar-Ilan University, Ramat-Gan 52900, Israel
{amir,porately}@cs.biu.ac.il
[2] Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218
[3] College of Computing, Georgia Institute of Technology, 801 Atlantic Drive, Atlanta, GA 30318, USA
axa@cc.gatech.edu
[4] Dipartimento di Ingegneria dell' Informazione, Università diPadova, Via Gradenigo 6/A, 35131 Padova, Italy
[5] Department of Computer Science, University of Haifa, Mount Carmel, Haifa 31905, Israel
landau@cs.haifa.ac.il
[6] Department of Computer Science and Engineering, Polytechnic Institute of New York University, 6 Metrotech Center, Brooklyn, NY 11201
[7] Department of Software Engineering, Shenkar College, 12 Anna Frank, Ramat-Gan, Israel
avivitlevy@shenkar.ac.il
[8] CRI, Haifa University, Mount Carmel, Haifa 31905, Israel

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 $[i, j]$, $0 < i \leq j \leq n$, compute $\max_{\ell,k \in \{i,\ldots,j\}} LCP(S_\ell, S_k)$, 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.

Surprisingly, while it is known how to preprocess a string in linear time to enable LCP computation of two suffixes in constant time, this seems quite difficult in the *Range LCP* problem. It is trivial to answer such queries in time $O(|j - i|^2)$ after a linear-time preprocessing and easy to show an $O(1)$ query algorithm after an $O(|S|^2)$ time preprocessing. We provide algorithms that solve the problem with the following complexities:

⋆ Partly supported by NSF grant CCR-09-04581, ISF grant 347/09, and BSF grant 2008217.

⋆⋆ Partly supported by BSF grant 2008217.

⋆⋆⋆ 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.

† Partly supported by BSF grant 2006334, ISF grant 1484/08, and Google award.

1. Preprocessing Time: $O(|S|)$, Space: $O(|S|)$, Query Time: $O(|j-i| \log\log n)$.
2. Preprocessing Time: no preprocessing, 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: $O(\log\log |S|)$.

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 [11]. 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 [12].

Therefore, the LCP has been amply studied and generalized versions of the problem are of interest. A first natural generalization of the problem is a "range" version. Indeed, a "range" version of the LCP problem was considered by Cormode and Muthukrishnan [4]. They called it 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 [10] to $O(|S| \log |S|)$ preprocessing time and $O(\log |S|)$ query time.

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 $A = A[1], \ldots, A[n]$ and $B = B[1], \ldots, B[n]$ 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 $a_1, \ldots, a_k$, $a_i \in \Sigma$, $i = 1, \ldots, k$, $k \leq n$, 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 $S = S[1], \ldots, S[n]$ 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, $1 \leq i, j, n$, can