Computing All Squares in Compressed Texts

Lesha Khvorost*
Ural Federal University
jaamal@mail.ru

Abstract

We consider the problem of computing all squares in a string represented by a straight-line program (SLP). An instance of the problem is an SLP $\mathbb{S}$ that derives some string S and we seek a solution in the form of a table that contains information about all squares in S in a compressed form. We present an algorithm that solves the problem in $O(|\mathbb{S}|^4 \cdot \log^2 |S|)$ time and requires $O(|\mathbb{S}| \cdot \max\{|\mathbb{S}|, \log |S|\})$ space, where $|\mathbb{S}|$ (respectively $|S|$) is the size of the SLP $\mathbb{S}$ (respectively the length of the string S).

1 Introduction

Various compressed representations of strings are known: straight-line programs (SLPs), collage-systems, string representations using antidictionaries, etc. Nowadays text compression based on context-free grammars such as SLPs attracts much attention. The reason for this is not only that grammars provide well-structured compression but also that the SLP-based compression is in a sense polynomially equivalent to the compression achieved by the Lempel-Ziv algorithm that is widely used in practice. It means that, given a string S, there is a polynomial relation between the size of an SLP that derives S and the size of the dictionary stored by the Lempel-Ziv algorithm [?].

While compressed representations save storage space, there is a price to pay: some classical problems on strings become computationally hard when one deals with compressed data and measures algorithms' speed in terms of the size of compressed representations. As examples we mention here the problems **Hamming distance** [?] and **Literal shuffle** [?]. On the other hand, there exist problems that admit algorithms working rather well on compressed representations: **Pattern matching** [?], **Longest common substring** [?], **Computing all palindromes** [?]. This dichotomy gives rise to the following research direction: to classify important string problems by their behavior with respect to compressed data.

The **Computing All Squares** (**CAS**) problem is a well-known problem on strings. It is of importance, for example, in molecular biology. Up to recently it is was not known whether or not **CAS** admits an algorithm polynomial in

*The author acknowledges support from the Russian Foundation for Basic Research, grant 10-01-00793.

the size of a compressed representation of a given string.[1] In general, a string can have exponentially many squares with respect to the size of its compressed representation. For example, the string a^n has $\Theta(n^2)$ squares, while it is easy to build an SLP of size $O(\log n)$ that derives a^n. So we must store information about squares in a compressed form. Also this implies that we cannot search for squares consecutively by moving from one square to the "next" one. Squares should be somehow grouped in relatively large families that are to be discovered at once.

We formulate the **CAS** problem in terms of SLPs as follows:
PROBLEM: **CAS**
INPUT: an SLP $\mathbb{S}$ that derives some text S;
OUTPUT: a data structure (a S-table) that contains information about all squares in S in a compressed form.

Theorem 1. *There is an algorithm that solves the* ***CAS*** *problem using* $O(|\mathbb{S}|^4 \cdot \log^2 |S|)$ *time and* $O(|\mathbb{S}| \cdot \max(|\mathbb{S}|, \log |S|))$ *space.*

We would like to emphasize main features of the algorithm:

- This algorithm is divided into independent steps in contrast to classical algorithms in this area which consecutively accumulate information about required objects. As a result it can be parallelized.
- This algorithm presents a new technique for SLPs processing..
- The algorithm is quite difficult for practical implementation. It is not excluded that constants hidden in the "O" notation are actually very big.
- The present upper bound for the time complexity is rather high and is not matched by any lower bound. The question whether the upper bound can be lowered to say, cubic in $|\mathbb{S}|$ remains open.

[1]A polynomial algorithm that solves **CAS** for strings represented by Lempel-Ziv encodings was announced in [**?**]. This representation is slightly more general than that by SLPs. However no details of the algorithm have ever been published.