ARTICLE IN PRESS

Discrete Applied Mathematics ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Closed factorization[☆]

Golnaz Badkobeh[a], Hideo Bannai[b], Keisuke Goto[b], Tomohiro I[c], Costas S. Iliopoulos[d], Shunsuke Inenaga[b], Simon J. Puglisi[e,*], Shiho Sugimoto[b]

[a] *Department of Computer Science, University of Sheffield, United Kingdom*
[b] *Department of Informatics, Kyushu University, Japan*
[c] *Department of Computer Science, TU Dortmund, Germany*
[d] *Department of Informatics, King's College London, United Kingdom*
[e] *Helsinki Institute for Information Technology, Department of Computer Science, University of Helsinki, Finland*

ARTICLE INFO

Article history:
Received 1 October 2014
Received in revised form 1 March 2016
Accepted 16 April 2016
Available online xxxx

Keywords:
Closed strings
Borders
Suffix trees
Range successor queries
Manhattan skyline problem

ABSTRACT

A *closed string* is a string with a proper substring that occurs in the string as a prefix *and* a suffix, but not elsewhere. Closed strings were introduced by Fici (2011) as objects of combinatorial interest in the study of Trapezoidal and Sturmian words. In this paper we present algorithms for computing closed factors (substrings) in strings. First, we consider the problem of greedily factorizing a string into a sequence of longest closed factors. We describe an algorithm for this problem that uses linear time and space. We then consider the related problem of computing, for every position in the string, the longest closed factor starting at that position. We describe a simple algorithm for the problem that runs in $O(n \log n / \log \log n)$ time, where n is the length of the string. This also leads to an algorithm to compute the maximal closed factor containing (i.e. covering) each position in the string in $O(n \log n / \log \log n)$ time. We also present linear time algorithms to factorize a string into a sequence of shortest closed factors of length at least two, to compute the shortest closed factor of length at least two starting at each position of the string, and to compute a minimal closed factor of length at least two containing each position of the string.

1. Introduction

A *closed string* is a string with a proper substring that occurs as a prefix and a suffix but not elsewhere in the string. Closed strings were introduced by Fici [5] as objects of combinatorial interest in the study of Trapezoidal and Sturmian words. Since then, Badkobeh, Fici, and Lipták [2] have proved a tight lowerbound for the number of closed factors (substrings) in strings of given length and alphabet size.

In this paper we initiate the study of algorithms for computing closed factors. In particular we consider the following algorithmic problems.

☆ This research is partially supported by the Academy of Finland through grants 258308 and by the Japan Society for the Promotion of Science.

* Corresponding author.

E-mail addresses: g.badkobeh@sheffield.ac.uk (G. Badkobeh), bannai@inf.kyushu-u.ac.jp (H. Bannai), keisuke.gotou@inf.kyushu-u.ac.jp (K. Goto), tomohiro.i@cs.tu-dortmund.de (T. I), c.iliopoulos@kcl.ac.uk (C.S. Iliopoulos), inenaga@inf.kyushu-u.ac.jp (S. Inenaga), puglisi@cs.helsinki.fi (S.J. Puglisi), shiho.sugimoto@inf.kyushu-u.ac.jp (S. Sugimoto).

http://dx.doi.org/10.1016/j.dam.2016.04.009
0166-218X/© 2016 Elsevier B.V. All rights reserved.

Longest closed factorization problem This problem is to factorize a given string into a sequence of longest closed factors (we give a formal definition of the problem below, in Section 2). We describe an algorithm for this problem that uses $O(n)$ time and space, where n is the length of the given string.

Longest closed factor array problem This problem requires us to compute the length of the longest closed factor starting at each position in the input string. We show that this problem can be solved in $O(n\frac{\log n}{\log\log n})$ time with $O(n)$ space, using techniques from computational geometry.

Maximal closed factor array problem This problem asks us to compute the length of a longest closed factor containing (i.e. covering) each position in the input string. We show that this problem can also be solved in $O(n\frac{\log n}{\log\log n})$ time with $O(n)$ space.

Shortest closed factorization problem This is the problem of factorizing a given string into a sequence of shortest closed factors of length at least two. We show this can be solved in $O(n)$ time and $O(\sigma)$ space, where σ is the alphabet size.

Shortest closed factor array problem This problem requires us to compute the length of the shortest closed factor of length at least two starting at each position in the input string. We solve this problem in $O(n)$ time and space.

Minimal closed factor array problem This problem asks us to compute, for each position in the input string, the length of the shortest closed factor of length at least two containing the position. We present an $O(n)$-time and space algorithm for this problem.

This paper proceeds as follows. In the Section 2 we set notation, define the problems more formally, and outline basic data structures and concepts. Section 3 describes an efficient solution to the longest closed factorization problem and Section 4 then considers the longest closed factor array and the maximal closed factor array. In Section 5, we present efficient algorithms for computing the shortest closed factorization, the shortest closed factor array, and the minimal closed factor array. Reflections and outlook are offered in Section 6.

Some of these results appeared in a preliminary version of this paper [1].

2. Preliminaries

2.1. Strings and closed factorization

Let Σ denote a fixed alphabet of $|\Sigma|$ distinct letters (or characters). An element of Σ^* is called a string. For any strings $\mathsf{W}, \mathsf{X}, \mathsf{Y}, \mathsf{Z}$ such that $\mathsf{W} = \mathsf{XYZ}$, the strings $\mathsf{X}, \mathsf{Y}, \mathsf{Z}$ are respectively called a prefix, substring, and suffix of W. The length of a string X will be denoted by $|\mathsf{X}|$. Let ε denote the empty string of length 0, i.e., $|\varepsilon| = 0$. For any non-negative integer n, $\mathsf{X}[1, n]$ denotes a string X of length n. A prefix X of a string W with $|\mathsf{X}| < |\mathsf{W}|$ is called a proper prefix of W. Similarly, a suffix Z of W with $|\mathsf{Z}| < |\mathsf{W}|$ is called a proper suffix of W. For any string X and integer $1 \le i \le |\mathsf{X}|$, let $\mathsf{X}[i]$ denote the ith character of X, and for any integers $1 \le i \le j \le |\mathsf{X}|$, let $\mathsf{X}[i..j]$ denote the substring of X that starts at position i and ends at position j. For convenience, let $\mathsf{X}[i..j]$ be the empty string if $j < i$. For any strings X and Y, if $\mathsf{Y} = \mathsf{X}[i..j]$, then we say that i is an occurrence of Y in X.

If a non-empty string X is both a proper prefix and suffix of string W, then, X is called a *border* of W. A string W is said to be *closed*, if there exists a border X of W that occurs exactly twice in W, i.e., $\mathsf{X} = \mathsf{W}[1..|\mathsf{X}|] = \mathsf{W}[|\mathsf{W}| - |\mathsf{X}| + 1..|\mathsf{W}|]$ and $\mathsf{X} \ne \mathsf{W}[i..i + |\mathsf{X}| - 1]$ for any $2 \le i \le |\mathsf{W}| - |\mathsf{X}|$. We define a single character $\mathsf{c} \in \Sigma$ to be closed, assuming that the empty string ε occurs exactly twice in c. A string X is a *closed factor* of W, if X is closed and is a substring of W. Throughout we consider a string $\mathsf{X}[1..n]$ on Σ. We define the *longest closed factorization* of string $\mathsf{X}[1..n]$ as follows.

Definition 1 (*Longest Closed Factorization*)**.** The longest closed factorization of string $\mathsf{X}[1..n]$, denoted $\mathsf{LCF}(\mathsf{X})$, is a sequence $(\mathsf{G}_0, \mathsf{G}_1, \ldots, \mathsf{G}_k)$ of strings such that $\mathsf{G}_0 = \varepsilon$, $\mathsf{X}[1, n] = \mathsf{G}_0\mathsf{G}_1\cdots\mathsf{G}_k$ and, for each $1 \le j \le k$, G_j is the longest prefix of $\mathsf{X}[|\mathsf{G}_0\cdots\mathsf{G}_{j-1}| + 1..n]$ that is closed.

Example 1. For string $\mathsf{X} = \mathtt{ababaacbbbcbcc}$, $\mathsf{LCF}(\mathsf{X}) = (\varepsilon, \mathtt{ababa}, \mathtt{a}, \mathtt{cbbbcb}, \mathtt{cc})$.

We remark that a closed factor G_j in an LCF is a single character if and only if $|\mathsf{G}_1\cdots\mathsf{G}_{j-1}| + 1$ is the rightmost (last) occurrence of character $\mathsf{X}[|\mathsf{G}_1\cdots\mathsf{G}_{j-1}| + 1]$ in X.

We also define the *longest closed factor array* of string $\mathsf{X}[1..n]$.

Definition 2 (*Longest Closed Factor Array*)**.** The longest closed factor array of string $\mathsf{X}[1..n]$ is an array $\mathsf{LNG}[1..n]$ of integers such that for any $1 \le i \le n$, $\mathsf{LNG}[i] = \ell$ if and only if ℓ is the length of the longest prefix of $\mathsf{X}[i..n]$ that is closed.

Example 2. For string $\mathsf{X} = \mathtt{ababaacbbbcbcc}$, $\mathsf{LNG} = [5, 4, 3, 5, 2, 1, 6, 3, 2, 4, 3, 1, 2, 1]$.

Clearly, given the longest closed factor array $\mathsf{LNG}[1..n]$ of string X, $\mathsf{LCF}(\mathsf{X})$ can be computed in $O(n)$ time. However, the algorithm we describe in Section 4 to compute $\mathsf{LNG}[1..n]$ requires $O(n\frac{\log n}{\log\log n})$ time, and so using it to compute $\mathsf{LCF}(\mathsf{X})$ would also take $O(n\frac{\log n}{\log\log n})$ time overall. In Section 3 we present an optimal $O(n)$-time algorithm to compute $\mathsf{LCF}(\mathsf{X})$ that does not require $\mathsf{LNG}[1..n]$.