# Closed Factorization ${ }^{\star}$ 

Golnaz Badkobeh ${ }^{1}$, Hideo Bannai ${ }^{2}$, Keisuke Goto ${ }^{2}$, Tomohiro $\mathrm{I}^{2}$, Costas S. Iliopoulos ${ }^{3}$, Shunsuke Inenaga ${ }^{2}$, Simon J. Puglisi ${ }^{4}$, and Shiho Sugimoto ${ }^{2}$<br>${ }^{1}$ Department of Computer Science, University of Sheffield United Kingdom<br>g.badkobeh@sheffield.ac.uk<br>${ }^{2}$ Department of Informatics, Kyushu University<br>Japan<br>\{bannai,keisuke.gotou,tomohiro.i,inenaga, shiho.sugimoto\}@inf.kyushu-u.ac.jp<br>${ }^{3}$ Department of Informatics, King's College London United Kingdom<br>c.iliopoulos@kcl.ac.uk<br>${ }^{4}$ Department of Computer Science, University of Helsinki<br>Finland<br>puglisi@cs.helsinki.fi


#### 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 (Proc. WORDS, 2011) as objects of combinatorial interest in the study of Trapezoidal and Sturmian words. In this paper we consider algorithms for computing closed factors (substrings) in strings, and in particular for 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 $\mathrm{O}(n \log n / \log \log n)$ time.


## 1 Introduction

A closed string is a string with but does not have internal occ objects of combinatorial inter Since then, Badkobeh, Fici, ar number of closed factors (subs

In this paper we initiate th particular we consider two alg, factorization problem, is to gre
 occurs as a pr were introduc pezoidal and $S$ ed a tight lowe en length and or computing first, which tring into a s closed factors (we give a formal definition of the problem below, describe an algorithm for this problem that uses $\mathrm{O}(n)$ time and spa length of the given string.

The second problem we consider is the closed factor array probld us to compute the length of the longest closed factor starting at ea

[^0][^1]input string. We show that this problem can be solved in $\mathrm{O}\left(n \frac{1}{\log }\right.$ techniques from computational geometry.

This paper proceeds as follows. In the next section we set no problems more formally, and outline basic data structures and co describes an efficient solution to the closed factorization problem a considers the closed factor array. Reflections and outlook are offere

## 2 Preliminaries

### 2.1 Strings and Closed Factorization

Let $\Sigma$ denote a fixed integer alphabet. An element of $\Sigma^{*}$ is called a string. For any strings $\mathrm{W}, \mathrm{X}, \mathrm{Y}, \mathrm{Z}$ such that $\mathrm{W}=\mathrm{XYZ}$, the strings $\mathrm{X}, \mathrm{Y}, \mathrm{Z}$ are respectively called a prefix, substring, and suffix of $W$. The length of a string $X$ will be denoted by $|X|$. Let $\varepsilon$ denote the empty string of length 0 , i.e., $|\varepsilon|=0$. For any non-negative integer $n, \mathrm{X}[1, n]$ denotes a string X of length $n$. A prefix X of a string W with $|\mathrm{X}|<|\mathrm{W}|$ is called a proper prefix of $W$. Similarly, a suffix $X$ of $W$ with $|Z|<|W|$ is called a proper suffix of W . For any string X and integer $1 \leq i \leq|\mathrm{X}|$, let $\mathrm{X}[i]$ denote the $i$ th character of X , and for any integers $1 \leq i \leq j \leq|\mathrm{X}|$, let $\mathrm{X}[i . . j]$ denote the substring of X that starts at position $i$ and ends at position $j$. For convenience, let $\mathrm{X}[i . . j]$ be the empty string if $j<i$. For any strings X and Y , if $\mathrm{Y}=\mathrm{X}[i . . j]$, 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., $\mathrm{X}=\mathrm{W}[1 . .|\mathrm{X}|]=\mathrm{W}[|\mathrm{W}|-|\mathrm{X}|+1 . .|\mathrm{W}|]$ and $\mathrm{X} \neq \mathrm{W}[i . . i+|\mathrm{X}|-1]$ for any $2 \leq i \leq|\mathrm{W}|-|\mathrm{X}|$. We suppose that any single character $C \in \Sigma$ is closed, assuming that the empty string $\varepsilon$ 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 $\mathrm{X}[1, n]$ on $\Sigma$. We define the closed factorization of string $\mathrm{X}[1, n]$ as follows.
Definition 1 (Closed Factorization). The closed factorization of string $X[1, n]$, denoted $\mathrm{CF}(\mathrm{X})$, is a sequence $\left(\mathrm{G}_{0}, \mathrm{G}_{1}, \ldots, \mathrm{G}_{k}\right)$ of strings such that $\mathrm{G}_{0}=\varepsilon, \mathrm{X}[1, n]=$ $\mathrm{G}_{1} \cdots \mathrm{G}_{k}$ and, for each $1 \leq j \leq k, \mathrm{G}_{j}$ is the longest prefix of $\mathrm{X}\left[\left|\mathrm{G}_{1} \cdots \mathrm{G}_{j-1}\right|+1 . . n\right]$ that is closed.

Example 2. For string $\mathrm{X}=$ ababaacbbbcbcc\$, $\mathrm{CF}(\mathrm{X})=(\varepsilon$, ababa, a, cbbbcb, cc, \$).
We remark that a closed factor $\mathrm{G}_{j}$ is a single character if and only if $\left|\mathrm{G}_{1} \cdots \mathrm{G}_{j-1}\right|+1$ is the rightmost (last) occurrence of character $\mathrm{X}\left[\left|\mathrm{G}_{1} \cdots \mathrm{G}_{j-1}\right|+1\right]$ in X .

We also define the longest closed factor array of string $\mathrm{X}[1, n]$.
Definition 3 (Longest Closed Factor Array). The longest closed factor array of $\mathrm{X}[1, n]$ is an array $\mathrm{A}[1, n]$ of integers such that for any $1 \leq i \leq n, \mathrm{~A}[i]=\ell$ if and only if $\ell$ is the length of the longest prefix of $\mathrm{X}[i . . n]$ that is closed

Example 4. For string $X=$ ababaacbbbcbcc $\$ . A=[5,4,3,5$,
Clearly, given the lor computed in $\mathrm{O}(n)$ time. $\mathrm{A}[1, n]$ requires $\mathrm{O}\left(n \frac{\log n}{\log \log }\right.$
$\mathrm{O}\left(n \frac{\log n}{\log \log n}\right)$ time overall. compute $\mathrm{CF}(\mathrm{X})$ that does
y $\mathrm{A}[1, n]$ of we describe to comput an optimal


Figure 1. The suffix tree of string $X=a b a b a \operatorname{cbbbcbcc} \$$, where each leaf stores the beginning position of the corresponding suffix. All the branches from an internal node are sorted in ascending lexicographical order, assuming $\$$ is the lexicographically smallest. The suffix array $S A$ of $X$ is $[15,5,3,1,6,4,2,8,9,10,12,14,7,11,13]$, which corresponds to the sequence of leaves from left to right, al linear time by a depth first traversal on the suffix tree.

## 2.2

The su of a st such tl $\mathrm{X}[i . . n]$ Th of X. Suf time for For a letters or $u$ to node $v$ an every string $X$
op subscripts when they are clear from the context) ] which contains a permutation of the integers [1..n]
 $[n] . . n]$. In other words, $\mathrm{SA}[j]=i$ iff aphical order. pacted trie consisting of all suffixes ce, and can be constructed in linear the suffix tree for an example string. $(w)$ be the string spelt out by the o $w$. If there is a branch from node say $u=\operatorname{parent}(v)$. Assuming that ter $\$$ which occurs nowhere else in $X$, there is a one-to-one correspondence between the suffixes of $X$ and the leaves of the suffix tree of $\mathbf{X}$. We assume the branches from a node $u$ to each child $v$ of $u$ are stored in ascending lexicographical order of pathlabel $(v)$. When this is the case, SA is simply the leaves of the suffix tree when read during a depth-first traversal. At each internal node $v$ in the suffix tree we store two additional values $v . s$ and $v . e$ such that SA [v.s..v.e] contains the beginning positions of all the suffixes in the subtree rooted at $v$.

## 3 Greedy Longest Closed Factorization in Linear Time

In this section, we present how to compute the closed factorization $\operatorname{CF}(X)$ of a given string $\mathrm{X}[1, n]$. Our high level strategy is to build a data structure that helps us to efficiently compute, for a given position $i$ in X , the longest closed factor starting at $i$. The core of this data structure is the suffix tree for X , which we decorate in various ways.

Let $S$ be the set of the beginning positions of the longest closed factors in $\mathrm{CF}(\mathrm{X})$. For any $i \in S$, let $\mathrm{G}=\mathrm{X}[i . . i+|\mathrm{G}|-1]$ be the longest closed factor of X starting at position $i$ in X .

Let $\mathrm{G}^{\prime}$ be the unique border of the longest closed factor G starting at position $i$ of X , and $b_{i}$ be its length, i.e., $\mathrm{G}^{\prime}=\mathrm{G}\left[1 . . b_{i}\right]=\mathrm{X}\left[i . . i+b_{i}-1\right]$ (if G is a single character, then $\mathrm{G}^{\prime}=\varepsilon$ and $b_{i}=0$ ). The following lemma shows that we can efficiently compute $\operatorname{CF}(\mathrm{X})$ if we know $b_{i}$ for all $i \in S$.

Lemma 5. Given $b_{i}$ for all $i \in S$, we can compute $\mathrm{CF}(\mathrm{X})$ in a total of $\mathrm{O}(n)$ time and space independently of the alphabet size.
$G=X[i]$. Hence, in this case it clearly takes $O(1)$ time and
an compute $G$ in $\mathrm{O}(|\mathrm{G}|)$ time and $\mathrm{O}\left(b_{i}\right)$ space, as follows. We
$\mathrm{g}^{\prime}$ of G using the Knuth-Morris-Pratt (KMP) string matching
rocessing takes $\mathrm{O}\left(b_{i}\right)$ time and space. We then search for the
n $\mathrm{X}[i+1 . . n]$ (i.e. the next occurrence of the longest border
of the occurrence $\left.\mathrm{X}\left[i . i+b_{i}-1\right]\right)$. The location of the next
re the end of the closed factor is, and so it also tells us $\mathrm{G}=$
arch takes $\mathrm{O}(|\mathrm{G}|)$ time - i.e. time proportional to the length

What remains is to be able to efficiently compute $b_{i}$ for a given $i \in S$. The following lemma gives an efficient solution to this subproblem:

Lemma 6. We can preprocess the suffix tree of string $\mathrm{X}[1, n]$ in $\mathrm{O}(n)$ time and space, so that $b_{i}$ for each $i \in S$ can be computed in $\mathrm{O}(1)$ time.

Proof. In each leaf of the suffix tree, we store the beginning position of the suffix corresponding to the leaf. For any internal node $v$ of the suffix tree of $\mathbf{X}$, let $\max (v)$ denote the maximum leaf value in the subtree rooted at $v$, i.e.,

$$
\max (v)=\max \{i \mid \mathrm{X}[i . . i+\operatorname{pathlabel}(v)-1], 1 \leq i \leq n-\operatorname{pathlabel}(v)-1\} .
$$

We can compute $\max (v)$ for every $v$ in a total of $\mathrm{O}(n)$ time total via a depth first traversal. Next, let $\mathrm{P}[1, n]$ be an array of pointers to suffix tree nodes (to be computed next). Initially every $\mathrm{P}[i]$ is set to null. We traverse the suffix tree in pre-order, and for each node $v$ we encounter we set $\mathrm{P}[\max (v)]=v$ if $\mathrm{P}[\max (v)]$ is null. At the end of the traversal $\mathrm{P}[i]$ will contain a pointer to the highest node $w$ in the tree for which $i$ is the maximum leaf value (i.e., $i$ is the rightmost occurrence of pathlabel $(w)$ ).

We are now able to compute $b_{i}$, the length of the unique border of the longest closed factor starting at any given $i$, as follows. First we retrieve node $v=\mathrm{P}[i]$. Observe that, because of the definition of $\mathrm{P}[i]$, there are no occurrences of substring $\mathrm{X}[i . . i+|\operatorname{pathlabel}(v)|]$ to the right of $i$. Let $u=\operatorname{parent}(v)$. There are two cases to consider:


Figure 2. Illustration of string X . We retrí black circle represents the same set of occur pathlabel $(u)-1$ ], whe
the longest closed factor $G$ starting at position $i$ nplies $\max (v)=i$. Let $u$ be the parent of $v$. The which represents $\mathrm{X}[i . . i+$ pathlabel $(u)]$, which has ce $b_{i}=|\operatorname{pathlabel}(u)|$, and therefore $\mathrm{G}=\mathrm{X}[i . . j+$ nce of pathlabel $(u)$ with $j>i$.

- If $u$ is not the root, then observe that there always exists an occurrence of substring pathlabel $(u)$ to the right of position $i$ (otherwise $i$ would be the rightmost occurrence of pathlabel $(u)$ e case since $u$ is higher than $v$, and we defined $\mathrm{P}[i]$ to be ith $\max (w)=i)$. Let $j$ be the the leftmost occurrence of at of $i$. Then, the longest closed factor starting at position by the KMP algorithm as
- If $u$ is the root, then it tu $X[i]$ in $X$. Hence, the longe ८) $\mid-1$ ] (this position $j$ is found

The thing we have not sh since the set of occurrences of $X[i . . j+$ pathlabe sponding to the string) is equivalent to that of $i$ that is longer than $|\operatorname{pathlabel}(u)|$ does not occ be any longer. Hence $\mid$ pathlabel $(u) \mid=b_{i}$. (See a

Clearly $v=\mathrm{P}[i]$ can be retrieved in $\mathrm{O}(1)$ tim
htmost occurrence of character at position $i$ is $\mathrm{X}[i]$. can be obtained in $\mathrm{O}(1)$ time from $v$. This con

The main result of this section follows:
deed the case, subtree correing starting at thus $b_{i}$ cannot

$$
u=\operatorname{parent}(v)
$$

Theorem 7. Given a string $\mathrm{X}[1, n]$ over an integer alphabet, the closed factorization $\mathrm{CF}(\mathrm{X})=\left(\mathrm{G}_{1}, \ldots, \mathrm{G}_{k}\right)$ of X can be computed in $\mathrm{O}(n)$ time and space.

Proof. $\mathrm{G}_{0}=\varepsilon$ by definition other $\mathrm{G}_{j}$ in ascending order X, i.e., $s_{1}=1$ and $s_{i}=\mid \mathrm{G}_{1}$ time and space from $b_{s_{1}}$ usir the first $j-1$ factors $\mathrm{G}_{1}, \ldots$ $\mathrm{O}\left(\left|\mathrm{G}_{j}\right|\right)$ time and space from the proof completes.

The following is an example of how the

We compute the on of $\mathrm{G}_{i}$ in in $\mathrm{O}\left(\left|\mathrm{G}_{1}\right|\right)$ computed pute $\mathrm{G}_{j}$ in ${ }_{1}\left|\mathrm{G}_{j}\right|=n$,
ction com- putes $C F(X)$ for a given string $X$.
r the running example string $X=$ ababaacbbbcbcc $\$$, and see ws the suffix tree of X .
ode $\mathrm{P}[1]$ representing ababaacbbbcbcc $\$$, whose parent represents et $b_{1}=|\mathrm{aba}|=3$. We run the KMP algorithm with pattern aba factor $\mathrm{G}_{1}=$ ababa.
2. We then check node $\mathrm{P}[6]$ representing a . Since its parent is the root, we get $b_{2}=0$ and therefore the second factor is $\mathrm{G}_{2}=\mathrm{a}$.
3. We then check node $\mathrm{P}[7]$ representing cbbbcbcc $\$$, whose parent represents cb . Hence we get $b_{3}=|\mathrm{cb}|=2$. We run the KMP algorithm with pattern cb and find the third factor $G_{3}=\mathrm{cbbbcb}$.
4. We then check node $\mathrm{P}[13]$ representing cc\$, whose parent represents c. Hence we get $b_{4}=|c|=1$. We run the KMP algorithm with pattern c and find the fourth factor $G_{1}=c c$
k node $\mathrm{P}[15]$ representing $\$$. Since its parent is the root, we get efore the fifth factor is $G_{5}=\$$.
btain $C F(X)=($ ababa, $a, ~ c b b b c b, ~ c c, ~ \$)$, which coincides with Ex-

## losed Factor Array

A haturar extersion of the problem in the previous section is to compute the longest closed factor starting at every position in X in linear time - not just those involved in the factorization. Formally, we would like to compute the longest closed factor array of X , i.e., an array $\mathrm{A}[1, n]$ of integers such that $\mathrm{A}[i]=\ell$ if and only if $\ell$ is the length of the longest closed factor starting at position $i$ in $X$.

Our algorithm for closed factorization computes the longest closed factor starting at a given position in time proportional to the factor's length, and so does not immediately provide a linear time algorithm for computing $A$; indeed, applying the algorithm naïvely at each position would take $\mathrm{O}\left(n^{2}\right)$ time to compute A . In what follows, we present a more efficient solution:

Theorem 9. Given a string $\mathrm{X}[1, n]$ over an integer alphabet, the closed factor array of X can be computed in $\mathrm{O}\left(n \frac{\log n}{\log \log n}\right)$ time and $\mathrm{O}(n)$ space.
Proof. We extend the data structure of the last section to allow A to be computed in $\mathrm{O}\left(n \frac{\log n}{\log \log n}\right)$ time and $\mathrm{O}(n)$ ge is to replace the KMP algorithm scanning in the first algori the closed factor in time i

We first preprocess th data structure of Yu , Hon given a range $[s, e] \subseteq[1$, null if there is no value l range successor queries to $\mathrm{O}\left(n \frac{\log n}{\log \log n}\right)$ time to construcu.

Now, to compute the longest closed factor starting to compute $\mathrm{A}[i]$ ) we do the following. First we comput the longest closed factor starting at $i$, in $\mathrm{O}(1)$ time usi X (i.e. order of process of computing $b_{i}$ we determine the node $u$ havir


We have considerea dut two problems on closed factors here, and many others remain. For example, how efficiently can one compute all the closed factors in a string (or, say, the closed factors that occur at least $k$ times)? Relatedly, how many closed factors does a string contain in the worst case and on average?

One also wonders if the closed factor array can be computed in linear time, by somehow avoiding range successor queries.

## References

1. G. Badkobeh, G. Fici, and Zs. Lipták: A note on words with the smallest number of closed factors. http://arxiv.org/abs/1305.6395, 2013.
2. M. Farach: Optimal suffix tree construction with large alphabets, in Proceedings of the 38th Annual Symposium on Foundations of Computer Science, 1997, pp. 137-143.
3. G. Fici: A classification of trapezoidal words, in Proc. 8th International Conference Words 2011 (WORDS 2011), Electronic Proceedings in Theoretical Computer Science 63, 2011, pp. 129-137, See also http://arxiv.org/abs/1108.3629v1.
4. D. E. Knuth, J. H. Morris, and V. R. Pratt: Fast pattern matching in strings. SIAM J. Comput., 6(2) 1977, pp. 323-350.
5. U. Manber and G. W. Myers: Suffix arrays: a new method for on-line string searches. SIAM Journal on Computing, 22(5) 1993, pp. 935-948.
6. G. Navarro and Y. Neckrich: Personal Communication.
7. G. Navarro and Y. Neckrich: Sorted range reporting, in Proc. Scandinavian Workshop on Algorithm Theory, LNCS 7357, Springer, 2012, pp. 271-282.
8. P. Weiner: Linear pattern matching, in IEEE 14th Annual Symposium on Switching and Automata Theory, IEEE, 1973, pp. 1-11.
9. C.-C. Yu, W.-K. Hon, and B.-F. Wang: Improved data structures for the orthogonal range successor problem. Computational Geometry, 44(3) 2011, pp. 148-159.

[^0]:    * This research is partially supported by the Academy of Finland through grants 258308 and 250345 (CoECGR), and by the Japan Society for the Promotion of Science.

[^1]:    Golnaz Badkobeh, Hideo Bannai, Keisuke Goto, Tomohiro I, Costas S. Iliopoulos, Shunsuke Inenaga, Simon J. Puglisi, Shiho Sugimoto: Closed Factorization, pp. 162-168.
    Proceedings of PSC 2014, Jan Holub and Jan Ždárek (Eds.), ISBN 978-80-01-05547-2 © Czech Technical University in Prague, Czech Republic

