Quasi-Linear-Time Algorithm for Longest Common Circular Factor

Mai Alzamel
Department of Informatics, King's College London, UK
Department of Computer Science, King Saud University, Riyadh, Saudi Arabia
https://nms.kcl.ac.uk/mai.alzamel/
mai.alzamel@kcl.ac.uk

Maxime Crochemore
Department of Informatics, King's College London, UK
Laboratoire d'Informatique Gaspard-Monge, Université Paris-Est, Marne-la-Vallée, France
http://www-igm.univ-mlv.fr/~mac/
maxime.crochemore@kcl.ac.uk

Costas S. Iliopoulos
Department of Informatics, King's College London, UK
https://nms.kcl.ac.uk/costas.iliopoulos/
costas.iliopoulos@kcl.ac.uk

Tomasz Kociumaka
Institute of Informatics, University of Warsaw, Poland
Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel
https://www.mimuw.edu.pl/~kociumaka/
kociumaka@mimuw.edu.pl

Jakub Radoszewski
Institute of Informatics, University of Warsaw, Poland
https://www.mimuw.edu.pl/~jrad
jrad@mimuw.edu.pl

Wojciech Rytter
Institute of Informatics, University of Warsaw, Poland
https://www.mimuw.edu.pl/~rytter
rytter@mimuw.edu.pl

Juliusz Straszyński
Institute of Informatics, University of Warsaw, Poland
j.straszynski@mimuw.edu.pl

Tomasz Waleń
Institute of Informatics, University of Warsaw, Poland
https://www.mimuw.edu.pl/~walen
walen@mimuw.edu.pl

Wiktor Zuba
Institute of Informatics, University of Warsaw, Poland
w.zuba@mimuw.edu.pl

Abstract

We introduce the Longest Common Circular Factor (LCCF) problem in which, given strings S and T of length at most n, we are to compute the longest factor of S whose cyclic shift occurs as a factor of T. It is a new similarity measure, an extension of the classic Longest Common Factor. We show how to solve the LCCF problem in $\mathcal{O}(n \log^4 n)$ time using $\mathcal{O}(n \log^2 n)$ space.

2012 ACM Subject Classification Theory of computation → Pattern matching

Keywords and phrases longest common factor, circular pattern matching, internal pattern matching, intersection of hyperrectangles

30th Annual Symposium on Combinatorial Pattern Matching (CPM 2019).
Editors: Nadia Pisanti and Solon P. Pissis; Article No. 25; pp. 25:1–25:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Digital Object Identifier 10.4230/LIPIcs.CPM.2019.25

Related Version `https://arxiv.org/abs/1901.11305`

Funding *Tomasz Kociumaka*: Supported by ISF grants no. 824/17 and 1278/16 and by an ERC grant MPM under the EU's Horizon 2020 Research and Innovation Programme (grant no. 683064).
Jakub Radoszewski: Supported by the "Algorithms for text processing with errors and uncertainties" project carried out within the HOMING programme of the Foundation for Polish Science co-financed by the European Union under the European Regional Development Fund.
Juliusz Straszyński: Supported by the "Algorithms for text processing with errors and uncertainties" project carried out within the HOMING programme of the Foundation for Polish Science co-financed by the European Union under the European Regional Development Fund.

1 Introduction

We introduce a new variant of the Longest Common Factor (LCF) Problem, called the Longest Common Circular Factor (LCCF) Problem. In the LCCF problem, given two strings S and T, both of length at most n, we seek for the longest factor of S whose cyclic shift occurs as a factor of T. The length of the LCCF is a new string similarity measure that is 2-approximated by the length of the LCF. We show that the exact value of LCCF can be computed efficiently.

A linear-time solution to the LCF problem is one of the best-known applications of the suffix tree [2]. Just as the LCF problem was an extension of the classical pattern matching, the LCCF can problem be seen as an extension of the circular pattern matching. The latter can still be solved in linear time using the suffix tree and admits a number of efficient solutions based on practical approaches [4, 10, 17, 21, 28, 32], also in the approximate variant [6, 7, 18, 20], as well as an indexing variant [3, 21, 22], and the problem of detecting various circular patterns [29]. The LCCF problem is further related to the notion of unbalanced translocations [9, 11, 31, 33, 34].

One can formally state the problem in scope as follows.

Longest Common Circular Factor (LCCF)
Input: Two strings S and T of length at most n each.
Output: A pair of longest factors, F of S and F' of T, for which there exist strings U and V such that $F = UV$ and $F' = VU$; we denote $\mathsf{LCCF}(S,T) = (F, F')$.

This problem can be solved in a straightforward way in $\mathcal{O}(n^2 \log n)$ time and $\mathcal{O}(n)$ space using *period queries* [26, 27, 24]; see Section 2.1. Our main result is the following.

▶ **Theorem 1** (Main Result). *The Longest Common Circular Factor problem on two strings of length at most n can be solved in $\mathcal{O}(n \log^4 n)$ time and $\mathcal{O}(n \log^2 n)$ space.*

Henceforth, we assume for simplicity that $|S| = |T| = n$; otherwise, the shorter string can be padded with a special character # that does not occur in either of the strings.

Our approach. We apply local consistency techniques from the area of internal pattern matching (in case U and V are not highly periodic; Section 3) and Lyndon roots (otherwise; Section 4). The LCCF problem is reduced to finding configurations satisfying conjunction of four conditions of type $i \in \mathsf{Occ}(X)$, where $\mathsf{Occ}(X)$ is the set of occurrences of a factor X.

Each configuration can be decomposed into two subconfigurations (pairs of consecutive fragments), one in S and one in T. We guarantee that the number of subconfigurations is nearly linear so that we can compute them all for both S and T. Then, the task reduces to finding two subconfigurations which agree (produce a full configuration) and constitute an optimal solution. This is done using geometric techniques in Section 6. Each condition $i \in \mathsf{Occ}(X)$ can be seen as membership of a point in a range since $\mathsf{Occ}(X)$ forms an interval in the suffix array. This gives a reduction of the LCCF problem to an intersection problem for 4D-rectangles. The latter task is solved efficiently using a sweep line algorithm.

2 Preliminaries

We consider strings over an integer alphabet Σ. If W is a string, then by $|W|$ we denote its length and by $W[1], \ldots, W[|W|]$ its characters. By $x = W[i \mathinner{.\,.} j]$ we denote a *fragment* of W between the ith and jth character, inclusively. We also denote this fragment x by $W[i \mathinner{.\,.} j+1)$, and we define $\mathsf{first}(x) = i$ as well as $\mathsf{last}(x) = j$. If $\mathsf{first}(x) = 1$, then x is a prefix, and if $\mathsf{last}(x) = |W|$, it is a suffix of W. Fragments x and y are *consecutive* if $\mathsf{last}(x) + 1 = \mathsf{first}(y)$; we then also say that y *follows* x.

The string $W[i] \cdots W[j]$ that corresponds to the fragment x is a *factor* of W. We say that two fragments *match* if the corresponding factors are the same. Let us note that a fragment can be represented by its endpoints in $\mathcal{O}(1)$ space; this representation can also be used to specify the corresponding factor.

By W^R we denote the reversal of a string W. We say that a positive integer p is a *period* of a string W if $W[i] = W[i+p]$ for all $i = 1, \ldots, |W| - p$. By $\mathsf{per}(W)$ we denote the shortest period of W. A string W is called *(weakly) periodic* if its shortest period satisfies $2\mathsf{per}(W) \le |W|$. Fine and Wilf's Periodicity Lemma [16] asserts that if a string W has periods p and q such that $p + q \le |W|$, then $\gcd(p, q)$ is also a period of W.

2.1 $\mathcal{O}(n^2 \log n)$-Time and $\mathcal{O}(n)$-Space Algorithm

A *period query* [26] is an internal query that is defined on a text W as follows: Given a fragment x of the text, report all periods of x (represented as several arithmetic progressions). In particular, the answer gives the shortest period $p = \mathsf{per}(x)$ and the longest *border* $U = x[1 \mathinner{.\,.} |x| - p] = x[1 + p \mathinner{.\,.} |x|]$. Period queries can be answered in $\mathcal{O}(\log n)$ time using a data structure of size $\mathcal{O}(n)$. A randomized $\mathcal{O}(n)$-time construction of this data structure was presented in [27], whereas a deterministic variant appeared in [24, Theorem 1.1.12].

▶ **Proposition 2.** *The Longest Common Circular Factor problem on two strings of length n can be solved in $\mathcal{O}(n^2 \log n)$ time and $\mathcal{O}(n)$ space.*

Proof. Let us set $W = S\#T\#S$, where $\#$ is a special character that occurs neither in S nor in T. For every pair of positions $i, j \in [1 \mathinner{.\,.} n]$, we ask a period query for $x = W[i \mathinner{.\,.} n+j] = S[i \mathinner{.\,.} n]\#T[1 \mathinner{.\,.} j)$ and $y = W[n+j+1 \mathinner{.\,.} 2n+1+i] = T[j \mathinner{.\,.} n]\#S[1 \mathinner{.\,.} i)$. This lets us recover the longest borders U of x and V of y so that (UV, VU) is a common circular factor of S and T. The longest of these factors over all pairs of positions (i, j) corresponds to the LCCF. ◀

2.2 Synchronizing Sets

In this section, we present the notion of *synchronizing sets* recently introduced by Kociumaka and Kempa [23] for BWT construction and answering LCE queries. Intuitively, a τ-synchronizing set P of a string W is a subset of position of W such that:

- the choice whether $i \in \mathsf{P}$ is made based on a context of 2τ subsequent characters,
- P contains at least one in every τ positions of each region of W whose period exceeds $\frac{1}{3}\tau$.

The underlying idea of making a locally consistent selection based on fixed-length contexts originates from internal pattern matching [27]. Later, the construction has been used for LCE queries [8] and derandomized [24]. We formalize our results using synchronizing sets as they provide a cleaner interface compared to that of the synchronizing functions of [24].

▶ **Definition 3** (Kempa and Kociumaka [23, Definition 3.1]). *Let W be a string of length n and let $\tau \le \frac{1}{2}n$ be a positive integer. We say that a set $\mathsf{P} \subseteq [1\mathinner{.\,.} n-2\tau+1]$ is a τ-*synchronizing set *of W if it satisfies the following conditions (see Figure 1):*

- *if $W[i\mathinner{.\,.} i+2\tau) = W[i'\mathinner{.\,.} i'+2\tau)$, then $i \in \mathsf{P}$ if and only if $i' \in \mathsf{P}$ (for $i, i' \in [1\mathinner{.\,.} n-2\tau+1]$),*
- *$\mathsf{P} \cap [i\mathinner{.\,.} i+\tau) = \emptyset$ if and only if $\mathsf{per}(W[i\mathinner{.\,.} i+3\tau-2]) \le \frac{1}{3}\tau$ (for $i \in [1\mathinner{.\,.} n-3\tau+2]$).*

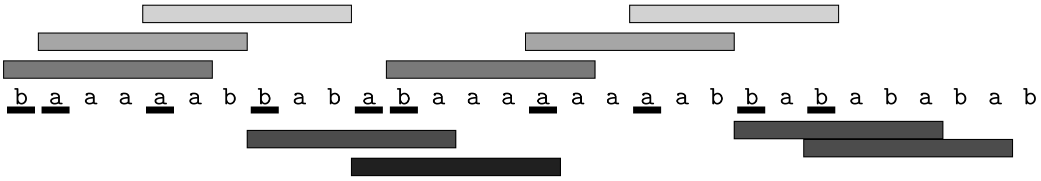

Figure 1 A 3-synchronizing set $\mathsf{P} = \{1, 2, 5, 8, 11, 12, 16, 19, 22, 24\}$ of a string W of length 30. The 10 positions in P are the starting positions of the occurrences of length-$(2 \cdot 3)$ factors `aabbab`, `aaaaab`, `baaaaa`, `bababa`, and `abaaaa` (marked as rectangles, listed top-down). Out of each three consecutive positions in $[1\mathinner{.\,.} 25]$, at least one belongs to P. The only exception is $[13\mathinner{.\,.} 16)$ due to the fact that $\mathsf{per}(W[13\mathinner{.\,.} 20]) = 1 \le \frac{1}{3} \cdot 3$.

A key technical result is a deterministic linear-time construction of a synchronizing set of optimal size $\mathcal{O}(\frac{n}{\tau})$. The linear running time is improved in [24, 23] to $\mathcal{O}(\frac{n}{\log_\sigma n})$ for small alphabet size σ and parameter $\tau = \Theta(\log_\sigma n)$, but that is not relevant for this paper.

▶ **Lemma 4** ([23, Proposition 8.10], [24, Lemma 4.4.9]). *Given a string W of length n and a positive integer $\tau \le \frac{1}{2}n$, in $\mathcal{O}(n)$ time one can construct a τ-synchronizing set of size $\mathcal{O}(\frac{n}{\tau})$.*

The central property of a synchronizing set P is that if a factor X is sufficiently long and not highly periodic, then most of the positions of P contained in the occurrences of X are located consistently. In our applications, we actually use the leftmost of these positions only. Hence, for an integer i and a set P, we define $\mathsf{succ}_P(i) = \min\{p \in P : p \ge i\}$ to be the *successor* of i in P. We assume that $\min \emptyset = \infty$ so that $\mathsf{succ}_P(i) = \infty$ if $i > \max P$.

▶ **Lemma 5.** *Let P be a τ-synchronizing set in a string W. If $W[i\mathinner{.\,.} j] = X = W[i'\mathinner{.\,.} j']$, where $|X| \ge 3\tau - 1$ and $\mathsf{per}(X) > \frac{1}{3}\tau$, then $\mathsf{succ}_\mathsf{P}(i) - i = \mathsf{succ}_\mathsf{P}(i') - i' \le |X| - 2\tau$.*

Proof. First, suppose for a proof by contradiction that $\mathsf{succ}_\mathsf{P}(i) - i > |X| - 2\tau = j - i + 1 - 2\tau$. Consequently, $\mathsf{P} \cap [i\mathinner{.\,.} j-2\tau+2) = \emptyset$, which yields $\mathsf{P} \cap [p\mathinner{.\,.} p+\tau) = \emptyset$ for $p \in [i\mathinner{.\,.} j-3\tau+2]$. Hence, Definition 3 implies $\mathsf{per}(W[p\mathinner{.\,.} p+3\tau-2]) \le \frac{1}{3}\tau$ for $p \in [i\mathinner{.\,.} j-3\tau+2]$. Due to $j - i + 1 = |X| \ge 3\tau - 1$, this range is non-empty, so the Periodicity Lemma yields $\frac{1}{3}\tau < \mathsf{per}(X) = \mathsf{per}(W[i\mathinner{.\,.} j]) \le \frac{1}{3}\tau$, contradicting our assumption that $\mathsf{succ}_\mathsf{P}(i) - i > |X| - 2\tau$.

Due to $\mathsf{succ}_\mathsf{P}(i) - i \le |X| - 2\tau$, we now conclude that $W[\mathsf{succ}_\mathsf{P}(i)\mathinner{.\,.} \mathsf{succ}_\mathsf{P}(i) + 2\tau) = W[i' - i + \mathsf{succ}_\mathsf{P}(i)\mathinner{.\,.} i' - i + \mathsf{succ}_\mathsf{P}(i) + 2\tau)$. Therefore, $\mathsf{succ}_\mathsf{P}(i) \in \mathsf{P}$ implies $i' - i + \mathsf{succ}_\mathsf{P}(i) \in \mathsf{P}$ in the light of Definition 3. Consequently, $\mathsf{succ}_\mathsf{P}(i') - i' \le \mathsf{succ}_\mathsf{P}(i) - i$. A symmetric argument shows that $\mathsf{succ}_\mathsf{P}(i) - i \le \mathsf{succ}_\mathsf{P}(i') - i'$, which completes the proof. ◀

2.3 Types of Factors

We define the *type* of a (non-empty) string W as $\mathsf{type}(W) = \lfloor \log(|W|+1)-1 \rfloor$. We denote by $\mathsf{LCCF}_{a,b}(S,T)$ the longest common circular factor (UV, VU) of S and T such that $\mathsf{type}(U) = a$ and $\mathsf{type}(V) = b$. We also say that it is the type-(a,b) LCCF. Moreover, we denote by $\mathsf{LCF}(S,T)$ the (ordinary) longest common factor of S and T (corresponding to $U = \varepsilon$ or $V = \varepsilon$). Our basic strategy is to compute $\mathsf{LCCF}_{a,b}(S,T)$ independently for every pair (a,b) and report the longest alternative among the obtained common circular factors, including (F,F) for $F = \mathsf{LCF}(S,T)$. However, we observe that if $\mathsf{LCCF}(S,T) = \mathsf{LCCF}_{a,b}(S,T) = (UV,VU)$, then $\frac{1}{2}|F| \le \frac{1}{2}|UV| \le \max(|U|,|V|) \le |F|$, and therefore $\mathsf{type}(F) - 1 \le \max(a,b) \le \mathsf{type}(F)$. Consequently, it suffices to iterate over $\mathcal{O}(\log|F|)$ pairs (a,b) satisfying the latter condition.

For each type a, we introduce a synchronizing set P_a of the concatenation ST. For $a = 0$, we set $\mathsf{P}_0 = [1\,.\,.\,|ST|]$, while for $1 \le a \le \mathsf{type}(ST)$, let us define P_a as a 2^{a-1}-synchronizing set of W. Using Lemma 4, we make sure that $|\mathsf{P}_a| = \mathcal{O}(\frac{n}{2^a})$ and the set P_a can be constructed in $\mathcal{O}(n)$ time, which sums up to $\mathcal{O}(n \log n)$ across all types a.

Moreover, let us define

$$\mathsf{P}_a(S) = \{p \in \mathsf{P}_a \,:\, p \le |S|\} \qquad \text{and} \qquad \mathsf{P}_a(T) = \{p - |S| \,:\, p \in \mathsf{P}_a,\, p > |S|\}.$$

Intuitively, $\mathsf{P}_a(S)$ and $\mathsf{P}_a(T)$ represent the subsets of P_a corresponding to S and T, respectively. The following result relates these notions to the common factors of S and T.

▶ **Corollary 6.** *If $S[i\,.\,.\,j] = F = T[i'\,.\,.\,j']$, where F is a type-a string satisfying $\mathsf{per}(F) > \frac{1}{6}2^a$, then $\mathsf{succ}_{\mathsf{P}_a(S)}(i) - i = \mathsf{succ}_{\mathsf{P}_a(T)}(i') - i' < |F|$.*

Proof. The claim is trivial for $a = 0$ due to $\mathsf{succ}_{\mathsf{P}_0(S)}(i) - i = \mathsf{succ}_{\mathsf{P}_0(T)}(i') - i' = 0$. Otherwise, we have $|F| \ge 2^{a+1} - 1 > 3 \cdot 2^{a-1} - 1$ and $\mathsf{per}(F) \ge \frac{1}{6}2^a = \frac{1}{3}2^{a-1}$, so Lemma 5 yields $\mathsf{succ}_{\mathsf{P}_a(S)}(i) - i = \mathsf{succ}_{\mathsf{P}_a(T)}(i') - i' \le |F| - 2 \cdot 2^{a-1} < |F|$. ◀

3 Nonperiodic Case

We say that a string U of type a is *highly periodic* if $\mathsf{per}(U) \le \frac{1}{6}2^a$. We consider now $\mathsf{LCCF}_{a,b}(S,T) = (F,F')$ such that $F = UV$, $F' = VU$, U is of type a, V is of type b, and neither U nor V is highly periodic. We call it the *nonperiodic* case.

For a pair of fragments (u,v), by $\Gamma_{u,v}$ we denote a condition which states that u is followed by a fragment that matches v and by $\Delta_{u,v}$ we denote a condition which states that v follows a fragment that matches u. We say that two pairs of consecutive fragments, (x,y) in S and (z,t) in T, *agree* if and only if

$$\Gamma_{y,z} \text{ and } \Delta_{y,z} \text{ and } \Gamma_{t,x} \text{ and } \Delta_{t,x}.$$

We reduce the LCCF problem in this case to the following abstract problem; see Figure 2.

Fragment-Families-Problem
Input: Two collections $\mathcal{F}_1$ and $\mathcal{F}_2$ of pairs of consecutive fragments of a string W of length n, with $m = |\mathcal{F}_1| + |\mathcal{F}_2|$
Output: $(x,y) \in \mathcal{F}_1$ and $(z,t) \in \mathcal{F}_2$ that agree and maximize $|x| + |y| + |z| + |t|$

For a string $W \in \{S,T\}$ and a type a, we introduce the following set of *synchronizers*:

$$\begin{aligned} \mathsf{LeftSync}_a(W,i) &= \mathsf{P}_a(W) \cap [i - 2^{a+2} + 2\,.\,.\,i-1],\\ \mathsf{RightSync}_a(W,i) &= \mathsf{P}_a(W) \cap [i\,.\,.\,i + 2^{a+2} - 3]. \end{aligned}$$

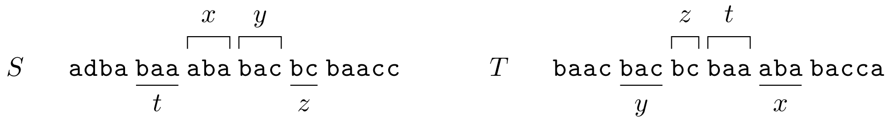

Figure 2 Pairs (x, y) and (z, t) agree; $txyz$ and $yztx$ form a common circular factor of S and T.

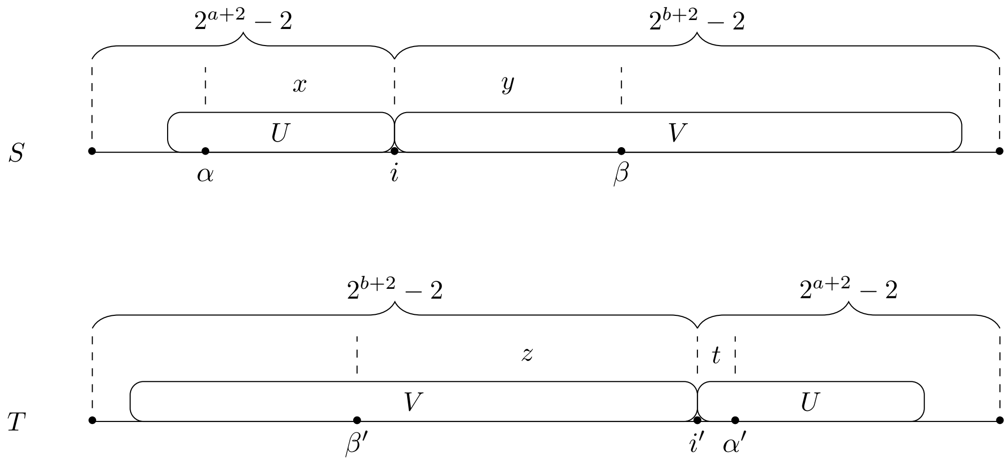

Figure 3 Assume that $\alpha \in \mathsf{LeftSync}_a(S, i)$, $\beta \in \mathsf{RightSync}'_b(S, i)$, $\beta' \in \mathsf{LeftSync}_a(T, i')$, $\beta' \in \mathsf{RightSync}'_b(T, i')$. Then $\Psi_S(\alpha, i, \beta) = (x, y)$ agrees with $\Psi_T(\beta', i', \alpha') = (z, t)$ if and only if there is a common circular factor of S and T: $F = UV$, $F' = VU$, where $U = tx$ and $V = yz$.

By $\mathsf{RightSync}'_a(W, i)$ we denote the singleton of the leftmost position in $\mathsf{RightSync}_a(W, i)$ or an empty set if there is no such position. For positions $\alpha \le i \le \beta$ in W, by

$$\Psi_W(\alpha, i, \beta) = (W[\alpha \mathinner{.\,.} i), W[i \mathinner{.\,.} \beta))$$

we denote a pair of consecutive fragments of W that are delimited by these positions. We then define the set of *candidates* (see Figure 3):

$$\mathsf{CAND}_{a,b}(W) = \{\Psi_W(\alpha, i, \beta) \;:\; \alpha \in \mathsf{LeftSync}_a(W, i),\ \beta \in \mathsf{RightSync}'_b(W, i),\ i \in [1 \mathinner{.\,.} |W|]\}.$$

Using this terminology, an informal scheme of a general algorithm is as follows:

```
Algorithm 1: Compute-LCCF$_{a,b}(S, T)$.
Compute the sets $\mathsf{CAND}_{a,b}(S)$, $\mathsf{CAND}_{b,a}(T)$
Find two pairs $(x, y) \in \mathsf{CAND}_{a,b}(S)$, $(z, t) \in \mathsf{CAND}_{b,a}(T)$ which agree and have
  maximum $|t| + |x| + |y| + |z|$
return $txyz$
```

▶ **Lemma 7** (Correctness for Nonperiodic Case). *The* $\mathsf{LCCF}_{a,b}$ *problem in the nonperiodic case can be reduced to the* Fragment-Families-Problem $(\mathcal{F}_S, \mathcal{F}_T)$ *for* $\mathcal{F}_S = \mathsf{CAND}_{a,b}(S)$ *and* $\mathcal{F}_T = \mathsf{CAND}_{b,a}(T)$.

Proof. Take a pair of fragments f_S of S and f_T of T such that f_S is an occurrence of a factor $F = UV$ and f_T is an occurrence of a factor $F' = VU$ such that U is of type a, V is of type b, and none of them is highly periodic. Denote by u_S and v_S the consecutive fragments of f_S corresponding to U and V, and similarly by v_T and u_T the consecutive fragments of f_T corresponding to V and U, and let $i = \mathsf{first}(v_S)$ and $j = \mathsf{first}(u_T)$. Moreover, consider $\alpha = \mathsf{succ}_{\mathsf{P}_a(S)}(\mathsf{first}(u_S))$ and $\alpha' = \mathsf{succ}_{\mathsf{P}_a(T)}(\mathsf{first}(u_T))$. By Corollary 6, $\alpha - \mathsf{first}(u_S) = \alpha' - \mathsf{first}(u_T) \le |U|$. Consequently, $\alpha \in \mathsf{LeftSync}_a(S, i)$ and $\alpha' \in \mathsf{RightSync}'_a(T, j)$. Moreover, the relative position of α within u_S coincides with the relative position of α' within u_T. Symmetrically, $\beta = \mathsf{succ}_{\mathsf{P}_b(S)}(\mathsf{first}(v_S)) \in \mathsf{RightSync}'_b(S, i)$ and $\beta' = \mathsf{succ}_{\mathsf{P}_b(T)}(\mathsf{first}(v_T)) \in \mathsf{LeftSync}_b(T, j)$. Moreover, the relative position of β within v_S coincides with the relative position of β' within v_T. This means that there exists a pair $(x, y) \in \mathsf{CAND}_{a,b}(S)$ such that $x = S[\alpha \,..\, i) = T[\alpha' \,..\, i' + |U|)$ and $y = S[i \,..\, \beta) = T[i' - |V| \,..\, \beta')$, and a pair $(z, t) \in \mathsf{CAND}_{b,a}(T)$ such that $z = T[\beta' \,..\, i') = S[\beta \,..\, i + |V|)$ and $t = T[i' \,..\, \alpha') = S[i - |U| \,..\, \alpha)$. The equalities listed above imply that the two pairs agree.

Conversely, for every two pairs $(x, y) \in \mathsf{CAND}_{a,b}(S), (z, t) \in \mathsf{CAND}_{b,a}(T)$ that agree, there exists a factor F in string S matching $txyz$ and a factor F' matching $yztx$ in T. Thus, there is a one-to-one correspondence between pairs that agree and fragments of strings of right type that are cyclic shifts. Hence, by finding two pairs that agree and maximize $|x| + |y| + |z| + |t|$, we construct a solution to the $\mathsf{LCCF}_{a,b}$ problem. ◀

▶ **Lemma 8** (Complexity for Nonperiodic Case). *In the nonperiodic case, the LCCF problem can be reduced in $\mathcal{O}(n \log n)$ time to $\mathcal{O}(\log n)$ instances of the* FRAGMENT-FAMILIES-PROBLEM *with $m = \mathcal{O}(n)$.*

Proof. For each type $a \in [0 \,..\, \mathsf{type}(\mathsf{LCF}(S, T))]$, we compute the synchronizing sets $\mathsf{P}_a(S)$ and $\mathsf{P}_a(T)$ in $\mathcal{O}(n)$ time using Lemma 4 as described in Section 2.3. Observe that each position $p \in \mathsf{P}_a(W)$ may belong to just $\mathcal{O}(2^a)$ sets $\mathsf{LeftSync}_a(W, i)$. Consequently, the total size of the sets $\mathsf{LeftSync}_a(W, i)$ (for a fixed type a) is $\mathcal{O}(n)$, and we can compute them in $\mathcal{O}(n)$ time using a sliding window. The running time across all types a is $\mathcal{O}(n \log n)$.

The family $\mathsf{CAND}_{a,b}(W)$ is constructed straight from the definition based on the sets $\mathsf{LeftSync}_a(W, i)$ and the synchronizing set $\mathsf{P}_b(W)$. As $|\mathsf{CAND}_{a,b}(W)| \le \sum_i |\mathsf{LeftSync}_a(W, i)|$, the size of this family is $\mathcal{O}(n)$, and the construction time is also linear. Across all $\mathcal{O}(\log n)$ pairs $a, b \ge 0$ with $\mathsf{type}(\mathsf{LCF}(S, T)) - 1 \le \max(a, b) \le \mathsf{type}(\mathsf{LCF}(S, T))$, the overall time complexity is $\mathcal{O}(n \log n)$. ◀

4 Periodic Case

We consider now $\mathsf{LCCF}_{a,b}(S, T) = (F, F')$ such that $F = UV$, $F' = VU$, U is of type a, V is of type b, and both U and V are highly periodic.

Recall that a *Lyndon string* is a string that is lexicographically smaller than all its non-trivial cyclic shifts. If W is a weakly periodic string with the shortest period p, then its *Lyndon root* λ is the Lyndon string that is a cyclic shift of $W[1 \,..\, p]$. A Lyndon representation of W is then (c, e, d) such that $W = \lambda' \lambda^e \lambda''$ where $|\lambda'| = c < |\lambda|$ and $|\lambda''| = d < |\lambda|$; see [13]. Lyndon strings have the following synchronization property that follows from the periodicity lemma: if λ is a Lyndon string, then it has exactly two occurrences in λ^2; see [12].

For a string W, by $\mathsf{HPerPref}_a(W)$ and $\mathsf{HPerSuf}_a(W)$ we denote the longest highly periodic type-a prefix and type-a suffix of W, respectively (or the empty string if there is no appropriate prefix or suffix). Let us start with the following simple observation; see Figure 4.

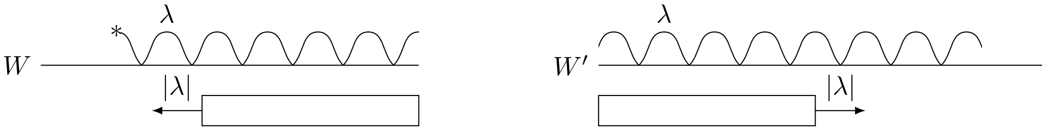

Figure 4 Illustration of Observation 9. A highly periodic suffix of W that is also a prefix of W' of length at most $\min(|X|,|Y|)-|\lambda|$ can be extended by $|\lambda|$ characters.

▶ **Observation 9.** *Let W and W' be two strings for which the strings $X = \mathsf{HPerSuf}_a(W)$ and $Y = \mathsf{HPerPref}_a(W')$ have the same Lyndon root λ. Then the longest suffix of W that is also a prefix of W' has length greater than $\min(|X|,|Y|)-|\lambda|$.*

For a position i in a string W and a type a, we denote by $\mathsf{LeftLyn}_a(W,i)$ the set of positions where the first, second, and last occurrence of the Lyndon root start in $\mathsf{HPerSuf}_a(W[1\mathinner{.\,.} i])$. If the latter string is empty, we assume that $\mathsf{LeftLyn}_a(W,i)$ is also empty. Similarly, we define $\mathsf{RightLyn}_a(W,i)$ as the set of positions where the first, second to last, and last occurrence of the Lyndon root start in $\mathsf{HPerPref}_a(W[i\mathinner{.\,.}|W|])$. We can redefine the set of candidates as follows (see Figure 5)

$$\mathsf{CAND}_{a,b}(W) = \{\Psi_W(\alpha,i,\beta) \;:\; \alpha\in\mathsf{LeftLyn}_a(W,i),\ \beta\in\mathsf{RightLyn}_b(W,i),\ i\in[1\mathinner{.\,.}|W|]\}.$$

The following lemma implies the correctness of our algorithm in this case.

▶ **Lemma 10** (Correctness for Periodic Case)**.** *The $\mathsf{LCCF}_{a,b}$ problem in the periodic case can be reduced to the* Fragment-Families-Problem *$(\mathcal{F}_S,\mathcal{F}_T)$ for the redefined sets $\mathcal{F}_S = \mathsf{CAND}_{a,b}(S)$ and $\mathcal{F}_T = \mathsf{CAND}_{b,a}(T)$.*

Proof. Take a pair of fragments f_S of S and f_T of T such that f_S is an occurrence of a factor $F = UV$ and f_T is an occurrence of a factor $F' = VU$ ($(F,F') = \mathsf{LCCF}_{a,b}(S,T)$) such that U is of type a, V is of type b, and both U and V are highly periodic. Denote by u_S and v_S the consecutive fragments of f_S corresponding to U and V, and similarly by v_T and u_T the consecutive fragments of f_T corresponding to V and U, and let $i = \mathsf{first}(v_S)$ and $j = \mathsf{first}(u_T)$.

Let $X = \mathsf{HPerSuf}_a(S[1\mathinner{.\,.} i])$ and $Y = \mathsf{HPerPref}_a(T[j\mathinner{.\,.} n])$. Note that u_S is a highly periodic suffix of X and that X has the same period as U (a different period would contradict the periodicity lemma). Symmetrically, u_T is a highly periodic prefix of Y and Y has the same period as U. Let λ be the Lyndon root of U, and observe that λ is also the Lyndon root of X and Y. By Observation 9, we have

$$|X|-|U|<|\lambda| \text{ or } |Y|-|U|<|\lambda|,$$

as otherwise we would be able to find a Common Circular Factor of type (a,b) that is longer by $|\lambda|$, thus contradicting our choice of f_S ad f_T.

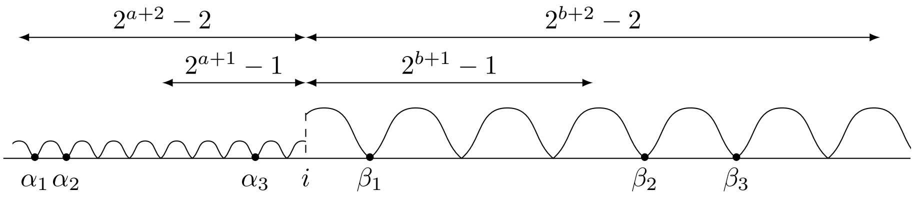

Figure 5 In this case, $\mathsf{CAND}_{a,b}(W)$ contains $\Psi_W(\alpha_p,i,\beta_q)$ for $p,q\in\{1,2,3\}$.

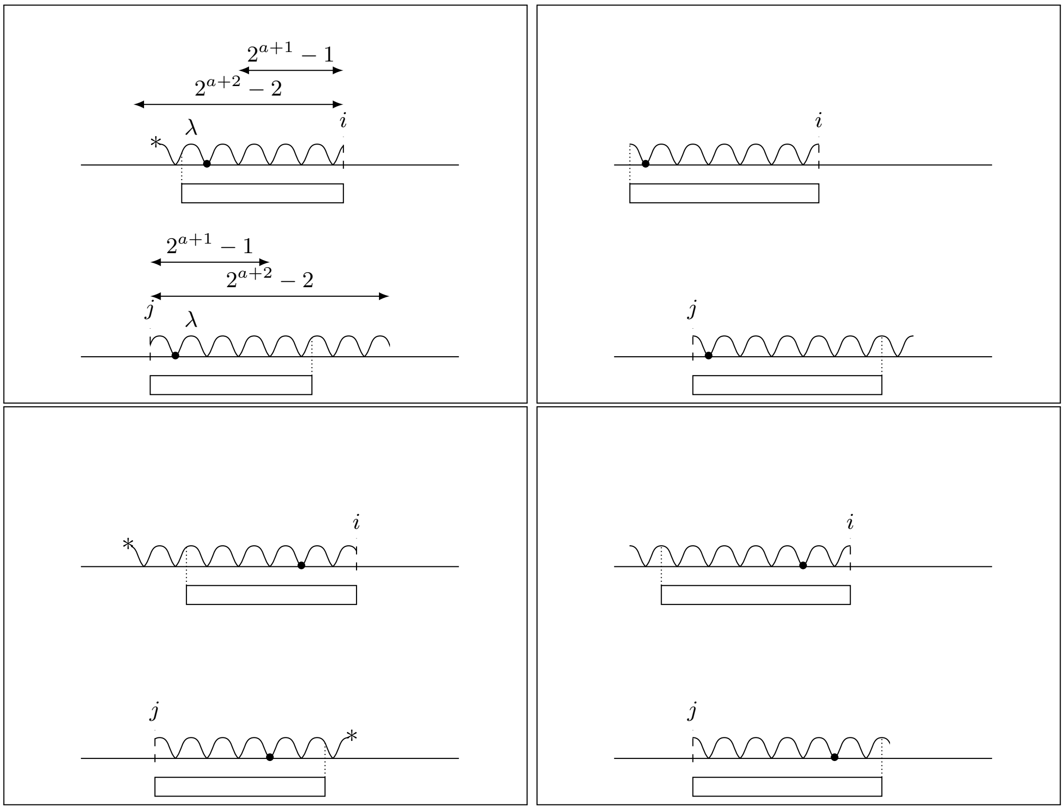

Figure 6 Four cases from the proof of Lemma 10.

If $|X|-|U|<|\lambda|$, then the first occurrence of λ in u_S is also the first or second occurrence of Lyndon root in X. This is due to the synchronization property of Lyndon strings. Moreover, the first λ in u_T is also the first occurrence of λ in Y. On the other hand, if $|Y|-|U|<|\lambda|$, then the last occurrence of λ in u_T is the last or the second to last occurrence of λ in u_T, whereas the last occurrence of λ in u_S is also the last occurrence of λ in X. In either case, u_S and u_T contain a pair of corresponding occurrences of λ whose starting positions belong to $\mathsf{LeftLyn}_a(S,i)$ and $\mathsf{RightLyn}_b(T,j)$, respectively; see Figure 6.

As the same reasoning can be applied to v_S and v_T, there exist pairs $(x,y)\in\mathsf{CAND}_{a,b}(S)$ and $(z,t)\in\mathsf{CAND}_{b,a}(T)$ which correspond to our choice of occurrences of the Lyndon roots. These pairs agree and $|x|+|y|+|z|+|t|=|F|$; thus, Fragment-Families-Problem $(\mathcal{F}_S,\mathcal{F}_T)$ will find a solution at least that good.

The converse direction is identical to the one from the proof of Lemma 7. ◀

We proceed with an efficient implementation. A *run* in string W is a maximal weakly periodic fragment $W[i\,..\,j]$ with a given period p. We use *2-period queries* which, given a weakly periodic fragment u of a string, compute its shortest period and the run of the same period it belongs to. Such queries can be answered in $\mathcal{O}(1)$ time after $\mathcal{O}(n)$-time preprocessing [27, 24] (for a simplified solution, see [5]). Let us also recall that the Lyndon representation of a run can be computed in constant time after linear-time preprocessing [13].

▶ **Lemma 11** (Complexity for Periodic Case). *In the periodic case, the LCCF problem can be reduced in $\mathcal{O}(n\log n)$ time to $\mathcal{O}(\log n)$ instances of the* Fragment-Families-Problem *with $m=\mathcal{O}(n)$.*

Proof. First, we spend $\mathcal{O}(n \log n)$ time in total to construct the sets $\mathsf{LeftLyn}_a(W, i)$ and $\mathsf{RightLyn}_a(W, i)$ for each $W \in \{S, T\}$ and $a \le \mathsf{type}(W)$. For this, we use the following result:

▷ Claim 12. After $\mathcal{O}(n)$-time preprocessing of a string W, each set $\mathsf{LeftLyn}_a(W, i)$ and $\mathsf{RightLyn}_a(W, i)$ can be constructed in $\mathcal{O}(1)$ time.

Proof. To compute $\mathsf{HPerPref}_a(u)$ for a fragment u of W, it suffices to ask a 2-period query for $u[1\..2^{a+1}-1]$ (see [27, 5]), determine the Lyndon representation of the resulting run (see [13, 5]), and then trim its Lyndon representation to u. A symmetric solution works for $\mathsf{HPerSuf}_a(u)$. This allows us to construct the sets $\mathsf{LeftLyn}_a$ and $\mathsf{RightLyn}_a$. ◁

Now, the family $\mathsf{CAND}_{a,b}(W)$, which is of size at most $9n$, can be computed in $\mathcal{O}(n)$ based on the sets $\mathsf{LeftLyn}_a(W, i)$ and $\mathsf{RightLyn}_b(W, i)$. Our reduction to the Fragment-Families-Problem problem relies on $\mathcal{O}(\log n)$ such families constructed for pairs $a, b \ge 0$ such that $\mathsf{type}(\mathsf{LCF}(S, T)) - 1 \le \max(a, b) \le \mathsf{type}(\mathsf{LCF}(S, T))$; see Section 2.3 and Lemma 8. ◀

5 General Case

Finally, we consider the general problem of computing $\mathsf{LCCF}_{a,b}(S, T) = (F, F')$. It can be reduced to several instances of the Fragment-Families-Problem directly by combining the techniques of the previous two sections.

▶ **Lemma 13** (Correctness for General Case). *The* $\mathsf{LCCF}_{a,b}$ *problem can be reduced to the* Fragment-Families-Problem $(\mathcal{F}_S, \mathcal{F}_T)$.

Proof. In the proofs of Lemmas 7 and 10 the U and V parts of the factors were considered separately. Hence, it is enough to define $\mathsf{CAND}_{a,b}(W)$ as

$$\begin{aligned}\mathsf{CAND}_{a,b}(W) = \{\Psi_W(\alpha, i, \beta) \ :\ & \alpha \in \mathsf{LeftSync}_a(W, i) \cup \mathsf{LeftLyn}_a(W, i),\\ & \beta \in \mathsf{RightSync}'_b(W, i) \cup \mathsf{RightLyn}_b(W, i),\ i \in [1\..|W|]\}.\end{aligned}$$

Depending on whether U or V is highly periodic or not, the existence of an agreeing pair $(x, y) \in \mathcal{F}_S$ and $(z, t) \in \mathcal{F}_T$ can be shown be repeating the arguments in the proofs of Lemma 7 or Lemma 10, respectively. ◀

▶ **Lemma 14** (Complexity for General Case). *The LCCF problem can be reduced in* $\mathcal{O}(n \log n)$ *time to* $\mathcal{O}(\log n)$ *instances of the* Fragment-Families-Problem *with* $m = \mathcal{O}(n)$.

Proof. The families can be computed combining the methods from Lemmas 8 and 11, obtaining the desired complexities and sizes. ◀

6 Solution to Fragment-Families-Problem

In this section we show how to solve the Fragment-Families-Problem for a string W of length n by a reduction to intersecting special 4-dimensional rectangles.

First, we give a geometric interpretation of two predicates:

- a factor U has an occurrence in W starting at position q (is a prefix of the suffix starting at position q), and
- U has an occurrence ending at position q (is a suffix of the prefix ending at position q)

relating them to the membership of q in a corresponding subinterval of $[1\..n]$.

Let us recall that the suffix array [30] of a string W, SA_W, is a permutation of $[1\..n]$ such that $W[\mathsf{SA}_W[i]\..n] < W[\mathsf{SA}_W[i+1]\..n]$ for every $i \in [1\..n-1]$. By $\mathsf{FirstPos}(U)$ let us denote the set of starting positions of occurrences of U in W. Our geometric interpretation is possible due to the following well known fact (see [12]).

▶ **Observation 15.** *The set* $\mathsf{FirstPos}(U)$ *consists of consecutive elements in* SA_W.

Let $\mathsf{LastPos}(U)$ be the set of ending positions of occurrences of U in W. We also use the FirstPos, LastPos notation for fragments which means operations on corresponding factors.

▶ **Observation 16.**

1. *A fragment* u *is a prefix/suffix of the suffix starting (prefix ending) at position* q *if and only if* $q \in \mathsf{FirstPos}(u)$, $q \in \mathsf{LastPos}(u)$, *respectively.*
2. $\Gamma_{u,v} \equiv ((\mathsf{last}(u)+1) \in \mathsf{FirstPos}(v))$ *and* $\Delta_{u,v} \equiv ((\mathsf{first}(v)-1) \in \mathsf{LastPos}(u))$.

We define a *d-rectangle* ($d \geq 2$) as a Cartesian product of d closed intervals, such that at least $d-2$ of them are singletons. E.g., $\{3\} \times [2\mathinner{.\,.}5] \times [1\mathinner{.\,.}7] \times \{0\}$ is a 4-rectangle. In other words, a d-rectangle is an isothetic hyperrectangle of dimension at most 2.

By $\mathcal{I}(U)$ and $\mathcal{J}(U)$ we denote the subintervals of $[1\mathinner{.\,.}n]$ that correspond to the intervals of $\mathsf{FirstPos}(U)$ in the suffix array SA_W and of $\mathsf{LastPos}(U)$ in the (analogously defined) prefix array of W, PA_W, respectively, as stated in Observation 15. (PA_W is a permutation of $[1\mathinner{.\,.}n]$ such that $W[1\mathinner{.\,.}\mathsf{PA}_W[i]]^R < W[1\mathinner{.\,.}\mathsf{PA}_W[i+1]]^R$ for every $i \in [1\mathinner{.\,.}n-1]$.) For pairs (x, y) and (z, t) of consecutive fragments, we denote:

$$\begin{aligned}\mathsf{RECT}(x,y) &= \mathcal{I}(x) \times \mathcal{J}(y) \times \{\mathsf{SA}_W^{-1}[\mathsf{last}(y)+1]\} \times \{\mathsf{PA}_W^{-1}[\mathsf{first}(x)-1]\},\\ \mathsf{RECT}'(z,t) &= \{\mathsf{SA}_W^{-1}[\mathsf{last}(t)+1]\} \times \{\mathsf{PA}_W^{-1}[\mathsf{first}(z)-1]\} \times \mathcal{I}(z) \times \mathcal{J}(t).\end{aligned}$$

Observation 16.2 now implies the following.

▶ **Observation 17.** *Two pairs of consecutive fragments* (x, y), (z, t) *agree if and only if* $\mathsf{RECT}(x, y) \cap \mathsf{RECT}'(z, t) \neq \emptyset$.

Two d-rectangles $[a_1\mathinner{.\,.}b_1]\times\cdots\times[a_d\mathinner{.\,.}b_d]$ and $[a'_1\mathinner{.\,.}b'_1]\times\cdots\times[a'_d\mathinner{.\,.}b'_d]$ are called *compatible* if, for each $i \in \{1,\ldots,d\}$, $[a_i\mathinner{.\,.}b_i]$ or $[a'_i\mathinner{.\,.}b'_i]$ is a singleton. Let us note that the 4-rectangles in the above observation are compatible.

6.1 Intersecting 4D Rectangles

We consider two families of 4-rectangles with weights and wish to find a pair of intersecting rectangles, one per family, with maximum total weight. The general problem of finding such an intersection of two families of m weighted hyperrectangles in d dimensions can be solved in $\mathcal{O}(m\log^{2d} m)$ time by an adaptation of a classic approach [14]. Below, we consider a special variant of the problem that has a much more efficient solution.

Max-Weight Intersection of Compatible Rectangles in 4D
Input: Two families $\mathcal{R}_1$ and $\mathcal{R}_2$ of 4-rectangles in $\mathbb{Z}^4$ with integer weights containing m rectangles in total, such that each $R_1 \in \mathcal{R}_1$ and $R_2 \in \mathcal{R}_2$ are compatible
Output: Check if there is an intersecting pair of 4-rectangles $R_1 \in \mathcal{R}_1$ and $R_2 \in \mathcal{R}_2$ and, if so, compute the maximum total weight of such a pair

A very similar problem was considered as Problem 3 in [19] for an arbitrary d. The sole difference is that the weight of an intersection of two d-rectangles $R_1 \in \mathcal{R}_1$ and $R_2 \in \mathcal{R}_2$ in that problem was the maximum ℓ_1-norm of a point in $R_1 \cap R_2$. A solution to Problem 3 for $d = 4$ in the case that the 4-rectangles are compatible working in $\mathcal{O}(m\log^3 m)$ time and $\mathcal{O}(m\log^2 m)$ space was given as [19, Lemma 5.8]. The algorithm presented in that lemma actually solves the Max-Weight Intersection of Compatible Rectangles in 4D problem and applies it for specific weight assignment of the 4-rectangles on the input. It uses hyperplane sweep and a variant of an interval stabbing problem. Henceforth, we will use the following result.

▶ **Fact 18** (see [19, Lemma 5.8]). Max-Weight Intersection of Compatible Rectangles in 4D *can be solved in* $\mathcal{O}(m \log^3 m)$ *time and* $\mathcal{O}(m \log^2 m)$ *space.*

6.2 Algorithm for Fragment-Families-Problem

Let us recall that the suffix tree [35] of a string W, ST_W, is a compacted trie of all the suffixes of W. It can be computed in $\mathcal{O}(n)$ time (see [15]) and reading the suffixes of W in its preorder traversal yields the suffix array of W. An efficient implementation of Observation 15 is known; see [1, 25].

▶ **Lemma 19.** *The sets* $\mathcal{I}(u)$ *and* $\mathcal{J}(u)$ *can be computed in* $\mathcal{O}(n+m)$ *total time for a batch of* m *fragments* u *of a length-*n *string* W*.*

Proof. Without loss of generality, it suffices to show how to compute $\mathcal{I}(u)$. For every explicit node of ST_W, we can compute the interval of elements of SA_W that are located in its subtree. This can be done in a bottom-up order in $\mathcal{O}(n)$ time.

A weighted ancestor query in ST_W, given a terminal node w and positive integer d, returns the ancestor of w located at depth d (being an explicit or implicit node). A batch of m such queries (for any tree of n nodes with positive integer weights of edges) can be answered in $\mathcal{O}(n+m)$ time; see [25, Section 7.1].

A weighted ancestor query can be used to compute, given a fragment u of W, the corresponding (explicit or implicit) node w of ST_W. The interval stored in the nearest explicit descendant of w equals $\mathcal{I}(u)$. ◀

We are now ready to show a solution to the Fragment-Families-Problem.

▶ **Lemma 20.** *The* Fragment-Families-Problem *can be solved in* $\mathcal{O}(n + m\log^3 m)$ *time and* $\mathcal{O}(n + m \log^2 m)$ *space.*

Proof. We construct families $\mathcal{R}_1$ and $\mathcal{R}_2$ of weighted 4-rectangles. For every $(x,y) \in \mathcal{F}_1$, we add $\mathsf{RECT}(x,y)$ to $\mathcal{R}_1$ with weight $|x|+|y|$. For every $(z,t) \in \mathcal{F}_2$, we add $\mathsf{RECT}'(z,t)$ to $\mathcal{R}_2$ with weight $|z|+|t|$. By Observation 17, the solution to Max-Weight Intersection of Compatible Rectangles in 4D for $\mathcal{R}_1$ and $\mathcal{R}_2$ is the solution to Fragment-Families-Problem$(\mathcal{F}_1, \mathcal{F}_2)$.

Note that we have $|\mathcal{R}_1| = |\mathcal{F}_1|$ and $|\mathcal{R}_2| = |\mathcal{F}_2|$. Using Lemma 19 and a linear-time algorithm for constructing SA_W and PA_W (and SA_W^{-1} and PA_W^{-1}) [15], computation of 4-rectangles RECT, RECT' can be done in $\mathcal{O}(n+m)$ time in total. Finally, Max-Weight Intersection of Compatible Rectangles in 4D can be solved in $\mathcal{O}(m\log^3 m)$ time and $\mathcal{O}(m \log^2 m)$ space. ◀

As a consequence of Lemmas 13 and 14 and the above lemma, we obtain the main result.

▶ **Theorem 1** (Main Result). *The Longest Common Circular Factor problem on two strings of length at most* n *can be solved in* $\mathcal{O}(n\log^4 n)$ *time and* $\mathcal{O}(n \log^2 n)$ *space.*

7 Conclusions

We have presented an $\mathcal{O}(n\log^4 n)$-time algorithm for computing the Longest Common Circular Factor (LCCF) of two strings of length n. Let us recall that the Longest Common Factor (LCF) of two strings can be computed in $\mathcal{O}(n)$ time. We leave an open question if the LCCF problem can also be solved in linear time.

References

1 Amihood Amir, Gad M. Landau, Moshe Lewenstein, and Dina Sokol. Dynamic text and static pattern matching. *ACM Transactions on Algorithms*, 3(2):19, 2007. doi:10.1145/1240233.1240242.

2 Alberto Apostolico, Maxime Crochemore, Martin Farach-Colton, Zvi Galil, and S. Muthukrishnan. 40 years of suffix trees. *Communications of the ACM*, 59(4):66–73, 2016. doi:10.1145/2810036.

3 Tanver Athar, Carl Barton, Widmer Bland, Jia Gao, Costas S. Iliopoulos, Chang Liu, and Solon P. Pissis. Fast circular dictionary-matching algorithm. *Mathematical Structures in Computer Science*, 27(2):143–156, 2017. doi:10.1017/S0960129515000134.

4 Md. Aashikur Rahman Azim, Costas S. Iliopoulos, Mohammad Sohel Rahman, and M. Samiruzzaman. A fast and lightweight filter-based algorithm for circular pattern matching. In Pierre Baldi and Wei Wang, editors, *5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics, BCB 2014*, pages 621–622. ACM, 2014. doi:10.1145/2649387.2660804.

5 Hideo Bannai, Tomohiro I, Shunsuke Inenaga, Yuto Nakashima, Masayuki Takeda, and Kazuya Tsuruta. The "Runs" Theorem. *SIAM Journal on Computing*, 46(5):1501–1514, 2017. doi:10.1137/15M1011032.

6 Carl Barton, Costas S. Iliopoulos, and Solon P. Pissis. Fast algorithms for approximate circular string matching. *Algorithms for Molecular Biology*, 9:9, 2014. doi:10.1186/1748-7188-9-9.

7 Carl Barton, Costas S. Iliopoulos, and Solon P. Pissis. Average-Case Optimal Approximate Circular String Matching. In Adrian-Horia Dediu, Enrico Formenti, Carlos Martín-Vide, and Bianca Truthe, editors, *Language and Automata Theory and Applications, LATA 2015*, volume 8977 of *LNCS*, pages 85–96. Springer, 2015. doi:10.1007/978-3-319-15579-1_6.

8 Or Birenzwige, Shay Golan, and Ely Porat. Locally Consistent Parsing for Text Indexing in Small Space. ArXiv preprint, 2018. arXiv:1812.00359.

9 Domenico Cantone, Simone Faro, and Arianna Pavone. Sequence Searching Allowing for Non-Overlapping Adjacent Unbalanced Translocations. ArXiv preprint, 2018. arXiv:1812.00421.

10 Kuei-Hao Chen, Guan-Shieng Huang, and Richard Chia-Tung Lee. Bit-Parallel Algorithms for Exact Circular String Matching. *The Computer Journal*, 57(5):731–743, 2014. doi:10.1093/comjnl/bxt023.

11 Da-Jung Cho, Yo-Sub Han, and Hwee Kim. Alignment with non-overlapping inversions and translocations on two strings. *Theoretical Computer Science*, 575:90–101, 2015. doi:10.1016/j.tcs.2014.10.036.

12 Maxime Crochemore, Christophe Hancart, and Thierry Lecroq. *Algorithms on strings*. Cambridge University Press, 2007. doi:10.1017/cbo9780511546853.

13 Maxime Crochemore, Costas S. Iliopoulos, Marcin Kubica, Jakub Radoszewski, Wojciech Rytter, and Tomasz Waleń. Extracting powers and periods in a word from its runs structure. *Theoretical Computer Science*, 521:29–41, 2014. doi:10.1016/j.tcs.2013.11.018.

14 Herbert Edelsbrunner. A new approach to rectangle intersections, Part I. *International Journal of Computer Mathematics*, 13(3–4):209–219, 1983. doi:10.1080/00207168308803364.

15 Martin Farach-Colton, Paolo Ferragina, and S. Muthukrishnan. On the sorting-complexity of suffix tree construction. *Journal of the ACM*, 47(6):987–1011, 2000. doi:10.1145/355541.355547.

16 Nathan J. Fine and Herbert S. Wilf. Uniqueness Theorems for Periodic Functions. *Proceedings of the American Mathematical Society*, 16:109–114, 1965. doi:10.1090/S0002-9939-1965-0174934-9.

17 Kimmo Fredriksson and Szymon Grabowski. Average-optimal string matching. *Journal of Discrete Algorithms*, 7(4):579–594, 2009. doi:10.1016/j.jda.2008.09.001.

18 Kimmo Fredriksson and Gonzalo Navarro. Average-optimal single and multiple approximate string matching. *ACM Journal of Experimental Algorithmics*, 9:1.4:1–1.4:47, 2004. doi:10.1145/1005813.1041513.

19 Szymon Grabowski, Tomasz Kociumaka, and Jakub Radoszewski. On Abelian Longest Common Factor with and without RLE. *Fundamenta Informaticae*, 163(3):225–244, 2018. doi:10.3233/FI-2018-1740.

20 Tommi Hirvola and Jorma Tarhio. Bit-Parallel Approximate Matching of Circular Strings with k Mismatches. *ACM Journal of Experimental Algorithmics*, 22:1.5:1–1.5:22, 2017. doi:10.1145/3129536.

21 Costas S. Iliopoulos, Solon P. Pissis, and M. Sohel Rahman. Searching and Indexing Circular Patterns. In Mourad Elloumi, editor, *Algorithms for Next-Generation Sequencing Data, Techniques, Approaches, and Applications*, pages 77–90. Springer, 2017. doi:10.1007/978-3-319-59826-0_3.

22 Costas S. Iliopoulos and M. Sohel Rahman. Indexing Circular Patterns. In Shin-Ichi Nakano and Md. Saidur Rahman, editors, *Algorithms and Computation, WALCOM 2008*, volume 4921 of *LNCS*, pages 46–57. Springer, 2008. doi:10.1007/978-3-540-77891-2_5.

23 Dominik Kempa and Tomasz Kociumaka. String synchronizing sets: Sublinear-time BWT construction and optimal LCE data structure. In Edith Cohen, editor, *51st Annual ACM Symposium on Theory of Computing, STOC 2019*. ACM, 2019. doi:10.1145/3313276.3316368.

24 Tomasz Kociumaka. *Efficient Data Structures for Internal Queries in Texts*. PhD thesis, University of Warsaw, October 2018. URL: https://www.mimuw.edu.pl/~kociumaka/files/phd.pdf.

25 Tomasz Kociumaka, Marcin Kubica, Jakub Radoszewski, Wojciech Rytter, and Tomasz Waleń. A Linear Time Algorithm for Seeds Computation. ArXiv preprint, 2019. arXiv:1107.2422v2.

26 Tomasz Kociumaka, Jakub Radoszewski, Wojciech Rytter, and Tomasz Waleń. Efficient Data Structures for the Factor Periodicity Problem. In Liliana Calderón-Benavides, Cristina N. González-Caro, Edgar Chávez, and Nivio Ziviani, editors, *String Processing and Information Retrieval, SPIRE 2012*, volume 7608 of *LNCS*, pages 284–294. Springer, 2012. doi:10.1007/978-3-642-34109-0_30.

27 Tomasz Kociumaka, Jakub Radoszewski, Wojciech Rytter, and Tomasz Waleń. Internal Pattern Matching Queries in a Text and Applications. In Piotr Indyk, editor, *26th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, pages 532–551. SIAM, 2015. doi:10.1137/1.9781611973730.36.

28 Jie Lin and Donald A. Adjeroh. All-Against-All Circular Pattern Matching. *The Computer Journal*, 55(7):897–906, 2012. doi:10.1093/comjnl/bxr126.

29 Jie Lin, Yue Jiang, and Don Adjeroh. Circular Pattern Discovery. *The Computer Journal*, 58(5):1061–1073, 2015. doi:10.1093/comjnl/bxu009.

30 Udi Manber and Eugene W. Myers. Suffix Arrays: A New Method for On-Line String Searches. *SIAM Journal on Computing*, 22(5):935–948, 1993. doi:10.1137/0222058.

31 Hideaki Ogiwara, Takashi Kohno, Hirofumi Nakanishi, Kazuhiro Nagayama, Masanori Sato, and Jun Yokota. Unbalanced translocation, a major chromosome alteration causing loss of heterozygosity in human lung cancer. *Oncogene*, 27(35):4788–4797, 2008. doi:10.1038/onc.2008.113.

32 Robert Susik, Szymon Grabowski, and Sebastian Deorowicz. Fast and Simple Circular Pattern Matching. In Aleksandra Gruca, Tadeusz Czachórski, and Stanisław Kozielski, editors, *Man-Machine Interactions, ICMMI 2013*, volume 242 of *Advances in Intelligent Systems and Computing*, pages 537–544. Springer, 2013. doi:10.1007/978-3-319-02309-0_59.

33 Dorothy Warburton. De novo balanced chromosome rearrangements and extra marker chromosomes identified at prenatal diagnosis: clinical significance and distribution of breakpoints. *The American Journal of Human Genetics*, 49(5):995–1013, 1991. PMID:1928105.

34 Brooke Weckselblatt, Karen E. Hermetz, and M. Katharine Rudd. Unbalanced translocations arise from diverse mutational mechanisms including chromothripsis. *Genome Research*, 25(7):937–947, 2015. doi:10.1101/gr.191247.115.

35 Peter Weiner. Linear Pattern Matching Algorithms. In *14th Annual Symposium on Switching and Automata Theory, SWAT 1973*, pages 1–11, Washington, DC, USA, 1973. IEEE Computer Society. doi:10.1109/SWAT.1973.13.