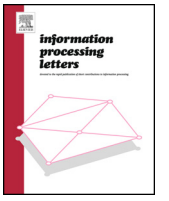
Information Processing Letters 182 (2023) 106398

Contents lists available at ScienceDirect

Information Processing Letters

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



Longest bordered and periodic subsequences

Hideo Bannai [a], Tomohiro I [b], Dominik Köppl [a,c,*]

[a] Tokyo Medical and Dental University, M&D Data Science Center, Tokyo, Japan
[b] Kyushu Institute of Technology, Department of Artificial Intelligence, Iizuka, Japan
[c] University of Muenster, Department of Computer Science, Muenster, Germany

ARTICLE INFO

Article history:
Received 1 March 2022
Received in revised form 3 April 2023
Accepted 5 April 2023
Available online 11 April 2023

Keywords:
Design of algorithms
Computational complexity
Dynamic programming
Subsequences
Periodicity

ABSTRACT

We present an algorithm computing the longest periodic subsequence of a string of length n in $\mathcal{O}(n^7)$ time with $\mathcal{O}(n^3)$ space. We obtain improvements when restricting the exponents or extending the search allowing the reported subsequence to be subperiodic down to $\mathcal{O}(n^2)$ time and $\mathcal{O}(n)$ space. By allowing subperiodic subsequences in the output, the task becomes finding the longest bordered subsequence, for which we devise a conditional lower bound.

1. Introduction

A natural extension of the analysis of regularities such as squares or palindromes perceived as substrings of a given text is the study of the same type of regularities when considering subsequences. In this line of research, given a text of length n, Kosowski [25] proposed an algorithm running in $\mathcal{O}(n^2)$ time using $\mathcal{O}(n)$ words of space to find a longest subsequence that is a square. Inoue et al. [21] generalized this setting to consider the longest such subsequence common of two texts T and S of length n, and gave an algorithm computing this sequence in $\mathcal{O}(n^6)$ time using $\mathcal{O}(n^4)$ space, also providing improvements in the case where the number of matching character pairs $T[i] = S[j]$ is rather small. Recently, Inoue et al. [22] provided similar improvements for the longest square subsequence of a single string. Here, we consider the problem for a single text, but allow the subsequence to have exponents other than a multiple of two. In detail, we want to find a longest subsequence that is periodic (exponent ≥ 2) or sub-periodic (exponent strictly between 1 and 2).

A list of related problems includes finding the longest palindromic subsequence [10,19], absent subsequences [24], longest increasing and decreasing subsequences [27,14,2], maximal common subsequences [26,11], the longest run subsequence [28], the longest Lyndon subsequence [4], longest rollercoasters [5,16,6,15], and computing the subsequence automaton [7,3]. Our techniques rely on finding longest common subsequences, which is conceived as a well-studied problem (see [17,18,23] and the references therein).

2. Preliminaries

Let $\mathbb{N}$ denote all natural numbers $1, 2, \ldots$, and $\mathbb{Q}^+$ the set of all rational numbers greater than or equal to 1. We distinguish integer intervals $\{1, 2, \ldots, n\} = [1..n] \subset \mathbb{N}$ and intervals of rational numbers $[1/2, 3/4] \subset \mathbb{Q}^+$.

Let Σ denote a totally ordered set of symbols called the *alphabet*. An element of Σ^* is called a *string*. Given a string $S \in \Sigma^*$, we denote its length with $|S|$ and its i-th symbol with $S[i]$ for $i \in [1..|S|]$. Further, we write $S[i..j] =$

* Corresponding author at: Tokyo Medical and Dental University, M&D Data Science Center, Tokyo, Japan.
E-mail addresses: hdbn@tmd.ac.jp (H. Bannai), tomohiro@ai.kyutech.ac.jp (T. I), dominik.koeppl@uni-muenster.de (D. Köppl).

https://doi.org/10.1016/j.ipl.2023.106398

$S[i]\cdots S[j]$ and $S[i..] = S[i]S[i+1]\cdots S[|S|]$. A *factorization* $T = F_1 \cdots F_z$ is a partitioning of T into substrings $F_1, \ldots, F_z$. A *subsequence* of a string S with length ℓ is a string $S[i_1]\cdots S[i_\ell]$ with $i_1 < \ldots < i_\ell$. For a string V, we denote $\mathrm{pos}_S(V) := \min\{i \in [0..|S|] : V \text{ is a subsequence of } S[1..i]\}$, and stipulate that $\mathrm{pos}_S(V) := \infty$ if V is not a subsequence of S.

Given a string S, we can write S in the form $S = U^\alpha U'$ with U' being either empty or a proper prefix of U. Then $u := |U|$ is called a *period* of S, and $\delta := \alpha + |U'|/u \in \mathbb{Q}^+$ is called the *exponent* of S with respect to the period u. For the largest possible such exponent δ, S is called *periodic* if $\delta \geq 2$, or *sub-periodic* if $\delta \in (1, 2)$. For instance, the unary string $T = \mathtt{a}\cdots\mathtt{a}$ has the minimum period 1 with exponent $|T|$, or more generally, period $p \in [1..|T|]$ with exponent $|T|/p$. A string S has a *border* P if P is both a non-empty proper prefix and a proper suffix of S. If S has no border, then it has only $|S|$ as a period (with exponent 1).

Further, for two strings Y and Z, let $\mathrm{LCS}_{Y,Z}(y, z)$ denote the longest common subsequence (LCS) of $Y[1..y]$ and $Z[1..z]$. We assume the reader to be familiar with the computation of the LCS via dynamic programming (DP), involving a DP matrix with $|Y||Z|$ entries (cf. [12, Chapter IV, Section 15.4] for a textbook reference). Instead of just two strings, we also make use of its generalization to determine the longest common subsequence of multiple strings: for a sequence $X_1, X_2, \cdots, X_k$ of strings, we write $\mathrm{LCS}(X_1, X_2, \ldots, X_k)$ to denote the longest common subsequence of $X_1, X_2, \ldots,$ and X_k. Assuming that all strings X_j have length $\mathcal{O}(n)$ for $j \in [1..k]$, it is known that we can lookup the length of the longest common subsequence $\mathrm{LCS}_{X_1,\ldots,X_k}(x_1, \ldots, x_k)$ of k strings $X_1[1..x_1], \ldots, X_k[1..x_k]$ in constant time if we have a table of size $\mathcal{O}(n^k)$ whose entries we can fill in $\mathcal{O}(kn^k)$ time via dynamic programming. Instead of a table, we can determine in $\mathcal{O}(n^{k-1})$ space the length of the LCS by keeping only two rows in one dimension of the k-dimensional DP matrix. To actually retrieve an LCS within $o(n^k)$ space, Schrödl [29] noticed that Hirschberg's algorithm [17] can reduce the space for the computation of the LCS of k strings of length n from $\mathcal{O}(n^k)$ to $\mathcal{O}(kn^{k-1})$.

Structure of the paper We first show an algorithm (Sect. 3) that computes the longest bordered subsequence. Bordered subsequences can be either periodic ($\delta \geq 2$) or sub-periodic ($\delta \in (1, 2) \subset \mathbb{Q}^+$). By reducing the problem of computing the (classic) LCS to the computation of the longest bordered subsequence, we obtain a conditional lower bound such that it seems hard to improve our algorithm for the longest bordered subsequence significantly. Subsequently, we modify this algorithm to omit the sub-periodic subsequences by allowing more time and space in Sect. 4. Table 1 gives an overview of our obtained results. Our algorithm that finds a longest bordered subsequence has better time and space complexities than those that find longest subsequences with more restricted exponents.

A key observation is that a longest (sub-)periodic subsequence S is *maximal*, meaning that no occurrence $T[i_1]T[i_2]\cdots T[i_{|S|}]$ of S in T can be extended with a character in $T[1..i_1 - 1]$ or $T[i_{|S|} + 1..]$ to form a longer subsequence without breaking the periodicity.

Table 1
Space and time complexities for finding subperiodic and/or periodic subsequences of specific exponents. ϵ is a rational number with $0 < \epsilon < 1$. The exponent column means that a subsequence is considered only if it has at least one exponent within the domain given in the column.

rational exponent δ	time	space	solution
$\delta > 1$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	Theorem 3.3
$\delta = 2\alpha, \alpha \in \mathbb{N}$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	[25]
$\delta \in (2, 3] \cup [4, \infty)$	$\mathcal{O}(n^5)$	$\mathcal{O}(n^2)$	Theorem 4.4
$\delta = (3+\epsilon)\alpha, \alpha \in \mathbb{N}$	$\mathcal{O}(n^7)$	$\mathcal{O}(n^3)$	Theorem 4.6

3. Longest bordered subsequence

We start with the search for a longest subsequence with a rational exponent $\delta > 1$. Thus, this sequence is periodic or subperiodic, i.e., it has a border. The idea is that we iterate over all possible factorizations of T into $T = Y \cdot Z$, and for each such factorization, we compute the longest subsequence $U'U'$ common in both Y and Z, and then find the longest extension of U' to $U'U''$ that is still a subsequence of Y, which gives us $U'U''U'$ as a possible candidate for the answer to our query. We then return the length of the longest such candidate with respect to all partitions.

Formally, we fix a factorization $T = Y \cdot Z$, and define an array $G_1[y] := 2 \cdot \mathrm{LCS}_{Y,Z}(y, |Z|) + |Y| - y$. See Fig. 1 for an example. The following lemma states that G_1 holds the answer to our problem if we have chosen the correct partition.

Lemma 3.1. *The maximum value of G_1 is the length of the longest subsequence $U'U''U'$ of YZ with $U'U''$ and U' being subsequences of Y and Z, respectively.*

Proof. Let y_s be the index at which the maximum value of G_1 is stored. Let S' be the longest common subsequence of $Y[1..y_s]$ and Z, and $S'' := Y[y_s + 1..]$. By definition, $|S'S''S'| = G_1[y_s]$. Assume that $|S'S''S'| < |U'U''U'|$. Obviously, S'' and U'' are suffixes of Y. Let y_u be the index such that $U'' = Y[y_u + 1..]$. Then U' is a common subsequence of $Y[1..y_u]$ and Z, with a length of at most $\mathrm{LCS}_{Y,Z}(y_u, |Z|)$. Hence, $|U'U''U'| \leq G_1[y_u] \leq G_1[y_s] = |S'S''S'|$, a contradiction. □

It is left to iterate over all partitions, and take the maximum. This leads us to the following theorem.

Theorem 3.2. *We can find a longest bordered subsequence in $\mathcal{O}(n^3)$ time using $\mathcal{O}(n)$ space.*

Proof. Fix one of the n different factorizations $T = Y \cdot Z$. While we compute the lengths in G_1, we compute simultaneously the necessary LCS values keeping only the y-th row of the DP matrix in memory such that we can compute $\mathrm{LCS}_{Y,Z}(y + 1, |Z|)$ from this row in $\mathcal{O}(|Z|)$ time and space. Neglecting the LCS computation, we can compute each entry of G_1 in constant time. Given the maximum

y	1	2	3	4	5
$Y[y]$	a	b	c	a	a
$G_1[y]$	abcaa · a	acaa · a	acaa · ac	acaa · aca	aca · aca

Fig. 1. G_1 defined in Sect. 3 for the text $T = \texttt{abcaaaca} = Y \cdot Z$ with the factors $Y = \texttt{abcaa}$ and $Z = \texttt{aca}$. For visualization, we show the subsequences found with G_1, although G_1 actually stores only their lengths. Each · separates the subsequences extracted from Y and Z. According to Lemma 3.1, the length of the longest bordered subsequence $U'U''U'$ of T with $U'U''$ and U' being subsequences of Y and Z, respectively, is the maximum value in G_1, which is $G_1[4] = 7$ in this example.

value $G_1[y']$ in G_1, we can compute an instance of a bordered subsequence that has the length $G_1[y']$ as follows. Since the y'-th entry stores the maximum length, the solution is $U' \cdot Y[y'+1..] \cdot U'$, and an instance of U' can be retrieved by Hirschberg's algorithm applied to the computation of LCS$(Y[1..y'], Z)$. In total, we need $\mathcal{O}(|Y||Z|)$ time for the LCS computation and Hirschberg's algorithm, but only $\mathcal{O}(n)$ space. So we need $\mathcal{O}(n^2)$ time per factorization $T = Y \cdot Z$, and therefore $\mathcal{O}(n^3)$ time for the entire computation. □

We subsequently show that we can improve the running time by some machinery due to Tiskin [31, Section 4.3], who showed that we can compute a longest repeated subsequence, i.e., a subsequence whose exponent is 2α for an integer $\alpha \geq 1$, with $\mathcal{O}(n)$ so-called *prefix-suffix LCS queries*. A prefix-suffix LCS query at positions $i, j \in [1..n-1]$ with $i < j$ asks to compute the length of an LCS of $T[1..i]$ and $T[j..]$. Tiskin [31, Theorem 4] proposed a data structure answering a prefix-suffix LCS query in $\mathcal{O}(\lg n / \lg\lg n)$ time; it can be built in $\mathcal{O}(n^2 \lg\lg n / \lg n)$ time. For finding a longest repeated subsequence, it is sufficient to consider only the $n-1$ queries of the form $j = i+1 \in [2..n]$.

Similarly, we can find a longest bordered sequence by computing the two indices $i, j \in [1..n]$ with $i < j$ that maximize $2 \cdot \text{LCS}(T[1..i], T[j..n]) + (j - i - 1)$ under the constraint that $\text{LCS}(T[1..i], T[j..n]) > 0$. These two indices can be found by considering all $\mathcal{O}(n^2)$ prefix-suffix LCS queries. We can then backtrack to find a longest common subsequence S of $T[1..i]$ and $T[j..n]$ (for instance by computing $\text{LCS}(T[1..i], T[j..n])$ in the textbook way using $\mathcal{O}(n^2)$ time), and report $S \cdot T[i+1..j-1] \cdot S$ as the solution.

In more detail, Tiskin [30, Algorithm 5.1] computes a so-called *semi-local seaweed matrix* in $\mathcal{O}(n^2)$ time in $\mathcal{O}(n)$ space. The semi-local seaweed matrix is a binary (permutation) matrix of size $2n \times 2n$, which stores at most one '1' per row, and thus can be represented space-economically in $\mathcal{O}(n)$ space [9, Definition 3]. By interpreting the $\mathcal{O}(n)$ '1's of this matrix as points in the plane, Tiskin [30, Thm. 4.10 and Equations after Ex. 4.9] reduces each prefix-suffix LCS to counting the number of points in a quarter-plane. Here, we can make use of the observation of Charalampopoulos et al. [9, Lemma 4], namely that these counting queries can be answered by the data structure of Chan and Patrascu [8], which uses $\mathcal{O}(n)$ space, can be built in $\mathcal{O}(n\sqrt{\lg n})$ time, and answers such a query in $\mathcal{O}(\lg n / \lg\lg n)$ time. In total, we obtain an algorithm computing the longest bordered subsequence in $\mathcal{O}(n^2 \lg n / \lg\lg n)$ time while using $\mathcal{O}(n)$ working space. For our setting, it is possible to shave the $\mathcal{O}(\lg n / \lg\lg n)$ time factor by resorting to simple scans of the matrix instead of building and using the data structure of Chan and Patrascu [8]. The idea is to process all possible corners row by row in a left-to-right manner, computing the entries of the matrix $H_{S,T}$ needed in [30, Thm. 4.10] and [9, Lemma 4], in $\mathcal{O}(n^2)$ total time. $H_{S,T}$ is a matrix whose entries correspond to precomputed LCS values of prefixes/suffixes of S and T; however $H_{S,T}$ is not computed explicitly. The space can be bounded by $\mathcal{O}(n)$ by maintaining the counter indices for processing the matrix and the length of the longest bordered subsequence computed so far. Overall, we obtain the following result.

Theorem 3.3. *We can find a longest bordered subsequence in $\mathcal{O}(n^2)$ time using $\mathcal{O}(n)$ space.*

To give evidence that a significant improvement over this algorithm is unlikely, we adapt the hardness result of Inoue et al. [20, Lemma 10] of longest square subsequences for longest bordered subsequences:

Theorem 3.4. *The computation of LCS(Y, Z) for two strings Y and Z, each of length n, can be reduced to the computation of a longest bordered subsequence of a string of length $\mathcal{O}(n)$ in linear time.*

Proof. Let $m := 2n + 1$ and define the string $T := \#^m Y \$^m \#^m Z \$^m$ with two special characters # and \$ that do neither appear in Y nor in Z. In what follows, we claim that we can deduce an LCS of Y and Z from a longest bordered subsequence L of T. Since $|YZ| = 2n < |\#^m\$^m\#^m\$^m| = 4m$, L has a length of at least $4m$ by consuming all occurrences of # and \$ appearing in T. Since L is a bordered subsequence, we can write it as $L = UU'$ with U' being a non-empty prefix of U.

In what follows we show that L always needs to contain all occurrences of # and \$ appearing in T, i.e., L is of the form $\#^m Y' \$^m \#^m Z' \$^m$, with Y' and Z' being subsequences of Y and Z, respectively.

- If U contains a #, then # must be a prefix of U and U'. If this were not the case then we would have selected characters from the second character run of #'s in T, i.e., for U we can select characters from the $(2n+3m)$-length substring $Y\$^m\#^m Z\$^m$ of T. However, $2n + 3m < 4m$, a contradiction. Hence, U and U' contain characters from the first and the second character run of # in T, and we maximize the length of L by taking all m #'s of each such run.
- By symmetry, if U' contains a \$, then $\$^m$ must be suffix of U and U'.

- Otherwise (U contains no # and U' contains no \$), we have selected a subsequence of the substring $Y\$^m\#^m Z$ of T, which has a length less than $4m$.

We conclude that U and U' have $\#^m$ and $\$^m$ as a prefix and a suffix, respectively.

Since # and \$ are unique characters, the longest border of L is $U' = \#^m Z' \$^m$, and therefore we know that Z' is the longest common subsequence of Y and Z. In particular, Y' cannot be longer than Z' since U and U' both start and end with the same characters, that appear nowhere else. We conclude that $U = U'$, in particular $Y' = Z'$. Finally, the LCS of Y and Z is Z' with $2|Z'| = |L| - 4m$. □

Abboud et al. [1] showed that the strong exponential time hypothesis (SETH) is false if there exists a constant $\epsilon > 0$ and an algorithm computing the LCS of $k \geq 2$ strings in $\mathcal{O}(n^{k-\epsilon})$ time for an alphabet of size $\mathcal{O}(k)$. Hence, solving the longest bordered subsequence in $\mathcal{O}(n^{2-\epsilon})$ time for any $\epsilon > 0$ would imply that SETH is false.

In the following we want to omit subsequences having exponents only in $(1, 2)$, i.e., we are interested in the longest subsequence having an exponent of at least two.

4. Longest periodic subsequence

Based on the approach of the previous section, our main idea is to generalize the factorization of T from 2 to k factors. By computing the LCS of these k factors, we can obtain all longest periodic subsequences having an exponent of the form $k\ell$ with $\ell \in \mathbb{N}$. For $k = 2$, we can find all square subsequences, i.e., the longest common subsequence with an exponent of 2α for $\alpha \in \mathbb{N}$, similar to Kosowski [25]. We generalize to exponent values with $\ell \in \mathbb{Q}^+$ by stopping matching characters in the last factor when computing a solution like in Sect. 3. Instead of considering all possible factorization sizes, the following lemma shows that it is sufficient to consider only the values $k = 3$ and $k = 4$.

Lemma 4.1. *Given a subsequence of T with a rational exponent larger than two, it has the form V^2V' or V^3V' with V' being a (not necessarily proper) prefix of V.*

Proof. Let $S = U^\delta$ be a subsequence of T with a rational exponent δ larger than two.

First, let us consider $\delta \geq 4$. Let $V := U^{\lceil \delta/3 \rceil}$ and $V' = U^\beta$ with $\beta := \delta - \lceil \delta/3 \rceil \cdot 2$. Notice that $0 \leq \beta \leq \lceil \delta/3 \rceil$ for $\delta \geq 4$, so we can write S in the form V^2V'. For $\delta \in (2, 3]$, set $V := U$ and $V' := U^\beta$ with $\beta \leq 1$. We can again write S in the form V^2V'.

Finally, we consider $\delta \in (3, 4)$, i.e., S has the form $S = (U'U'')^3U'$. Then it is in general not possible to write S in the form V^2V' since both occurrences of V as well as V' have to start with a prefix of U' having four occurrences. However, it is possible to write S as V^3V' by setting $V := U'U''$ and $V' := U'$. □

A conclusion is that we can find all subsequences with an exponent in $(2, 3] \cup [4, \infty)$ by a factorization of size three, while we need a factorization of size four for finding those with exponents within $(3, 4)$.

4.1. Three factors

We start with an algorithm factorizing T into three factors, and first consider a special case.

Lemma 4.2. *We can find a longest periodic subsequence of T with an exponent divisible by three in $\mathcal{O}(n^5)$ time using $\mathcal{O}(n^2)$ space.*

Proof. Let $T = XYZ$ be a factorization of T into three factors. We can compute the longest periodic subsequence $X'Y'Z'$ of T with an exponent divisible by three such that X', Y', and Z' are subsequences of X, Y, and Z, respectively, by a call of $\mathrm{LCS}_{X,Y,Z}(|X|, |Y|, |Z|)$, taking $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ space. However, there are $\binom{n}{2} = \mathcal{O}(n^2)$ such partitions of T into three factors, and therefore, we perform $\mathcal{O}(n^2)$ LCS computations, giving $\mathcal{O}(n^5)$ total time to determine the factorization that yields the maximal length. We then apply Hirschberg's algorithm on that factorization to obtain a subsequence of T with the desired properties. □

For the general case, we can make an observation similar to Sect. 3 for G_1 in that it suffices to determine the pair of positions (x, y) until which we consider the longest common subsequence U' of $X[1..x]$, $Y[1..y]$, and Z and then prolong the subsequences U' contained in $X[1..x]$ and $Y[1..y]$ with the longest common subsequence U'' of $X[x + 1..]$ and $Y[y + 1..]$ to form $U'U''$. More formally, let G_2 be a 2-dimensional matrix with $G_2[x, y] := 3 \cdot \mathrm{LCS}_{X,Y,Z}(x, y, |Z|) + 2 \cdot \mathrm{LCS}(X[x + 1..], Y[y + 1..])$. See Fig. 2 for an example. The following property holds.

Lemma 4.3. *The maximum value stored in G_2 is the length of the longest subsequence $(U'U'')^2U'$ of T such that (a) $U'U''$ is a common subsequence of $X[1..x]$, and $Y[1..y]$, and (b) U' is a common subsequence of $X[1..x]$, $Y[1..y]$, and Z.*

Proof. We basically follow the proof of Lemma 3.1. Let x_s and y_s be the indices in G_2 at which the maximum value $G_2[x_s, y_s]$ is stored. Let S' be the longest common subsequence of $X[1..x_s]$, $Y[1..y_s]$, and Z, and let $S'' := \mathrm{LCS}(X[x_s + 1..], Y[y_s + 1..])$. By definition, $|(S'S'')^2S'| = G_2[x_s, y_s]$. Assume that $|(S'S'')^2S'| < |(U'U'')^2U'|$. Let $x_u := \mathrm{pos}_X(U')$ and $y_u := \mathrm{pos}_Y(U')$ be the smallest indices such that U' is a common subsequence of $X[1..x_u], Y[1..y_u]$, and Z. On the one hand, we have $|U'| \leq \mathrm{LCS}_{X,Y,Z}(x_u, y_u, |Z|)$, and equality follows from the fact that we otherwise would find a maximum longest common subsequence V' of $X[1..x_u], Y[1..y_u]$, and Z such that $|V'| = \mathrm{LCS}_{X,Y,Z}(x_u, y_u, |Z|)$ and $(V'U'')^2V'$ is longer than $(U'U'')^2U'$. On the other hand, we have $|U''| \leq \mathrm{LCS}(X[x_u + 1..], Y[y_u + 1..])$, and equality follows from a similar argument. Consequently, $|(U'U'')^2U'| = G_2[x_u, y_u] \leq G_2[x_s, y_s]$. □

We again iterate over all possible factorizations and obtain the following theorem.

G_2

i	1	2	3	4
Y \ $X[i]$	a	a	b	b
$Y[1] =$ a	abb · abb · a	abb · abb · a	ab · ab · a	a · a · a
$Y[2] =$ b	ab · a · a	ab · ab · a	abb · abb · ab	ab · ab · ab
$Y[3] =$ b	ab · a · a	a · a · a	ab · ab · ab	ab · ab · ab

Fig. 2. G_2 defined in Sect. 4.1 for the text $T = \texttt{aabbabbaab} = XYZ$ with the factors $X = \texttt{aabb}$, $Y = \texttt{abb}$, and $Z = \texttt{aab}$. Given a longest subsequence $(U'U'')^2U''$ with $U'U''$ being a common subsequence of X and Y, and U'' being a common subsequence of X, Y, and Z, then, according to Lemma 4.3, the length of this subsequence is the maximum value stored in G_2, which is $G_2[3,2] = 8$ in this example.

Theorem 4.4. *We can find a longest periodic subsequence with an exponent in* $(2,3] \cup [4,\infty)$ *in* $\mathcal{O}(n^5)$ *time using* $\mathcal{O}(n^2)$ *space.*

Proof. Let us fix a factorization $T = X \cdot Y \cdot Z$. We allocate space for G_2 having $\mathcal{O}(n^2)$ entries, and fill G_2 as follows: While computing the length of the LCS of X, Y, Z with $\mathcal{O}(|Y||Z|)$ space keeping only the DP matrix values for $X[1..x-1]$ and $X[1..x]$ in memory, we can compute $G_2[x,y]$ in constant time. Hence, we can fill G_2 in $\mathcal{O}(n^3)$ time (the time for the LCS computation of X, Y, Z) while needing $\mathcal{O}(|Y||Z|) = \mathcal{O}(n^2)$ additional working space. If $G_2[x',y']$ stores the maximum value of G_2, we can proceed similarly to the proof of Theorem 3.2. Namely, we can make use of Hirschberg's algorithm to retrieve a common subsequence U' of X, Y, and Z of length $\mathrm{LCS}_{X,Y,Z}(x', y', |Z|)$. After computing the common subsequence S of $X[x'+1..]$ and $Y[y'+1..]$ in $\mathcal{O}(n^2)$ time and space, we obtain the subsequence $U'SU'$. In total, we can compute this subsequence in $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ space per factorization. Finally, we proceed as in the proof of Lemma 4.2 for taking the maximum value of the computed lengths over all factorizations and finding an instance of a subsequence with the desired properties. □

4.2. Four factors

Finally, we consider a factorization of size four to capture exponents in $(3,4)$. We have $\binom{n}{3} = \mathcal{O}(n^3)$ possibilities to factorize T into four factors W, X, Y, Z. Let us fix a factorization $T = WXYZ$.

We can compute similarly to G_2 the 3-dimensional matrix

$$G_3[w,x,y] := 4 \cdot \mathrm{LCS}_{W,X,Y,Z}(w,x,y,|Z|) + 3 \cdot \mathrm{LCS}(W[w+1..], X[x+1..], Y[y+1..]).$$

Compared to Sect. 4.1, we just added another dimension by the additional input string W, which is treated the same way as we have treated X and Y. Therefore, the consequences are straight-forward.

Lemma 4.5. *The maximum value stored in* G_3 *is the length of the longest subsequence* $(U'U'')^3U'$ *of* T *such that (a)* $U'U''$ *is a common subsequence of* $W[1..w]$, $X[1..x]$, *and* $Y[1..y]$, *and (b)* U' *is a common subsequence of* $W[1..w]$, $X[1..x]$, $Y[1..y]$, *and* Z.

Proof. Analogous to Lemma 4.3. □

By adding an additional dimension to the proof of Theorem 4.4 to compute G_3 instead of G_2, we obtain our final theorem.

Theorem 4.6. *We can compute the longest periodic subsequence with an exponent in* $\bigcup_{\alpha\in\mathbb{N}}(3\alpha, 4\alpha)$ *in* $\mathcal{O}(n^7)$ *time using* $\mathcal{O}(n^3)$ *space.*

Proof. We have $\mathcal{O}(n^3)$ different factorizations, and for each factorization we compute G_3 in $\mathcal{O}(n^4)$ time within $\mathcal{O}(n^3)$ additional space. □

5. Open problems

We are unaware of polynomial-time algorithms computing several other types of regularities when considering subsequences. For instance, we are not aware of an algorithm computing the longest *sub-periodic* subsequence, or an efficient algorithm computing the longest (common) subsequence *without a border*. Other problems are finding the longest (common) subsequence that is *primitive* (no exponent in $\mathbb{N} \setminus \{1\}$), or the longest (common) subsequence that is *non-primitive* (exponent in $\mathbb{N} \setminus \{1\}$).

Finally, we would like to study better lower bounds on the time complexities for computing our proposed problems. For $m = \Theta(n)$ large enough, we can generalize Theorem 3.4 to make use of Theorem 4.4 computing the longest periodic subsequence with an exponent in $(2,3) \cup [4,\infty)$ of the string $\#^m X\$^m\#^m Y\$^m\#^m Z\$^m$ to find $\mathrm{LCS}(X,Y,Z)$, and of Theorem 4.6 computing the longest periodic subsequence with an exponent in $\bigcup_{\alpha\in\mathbb{N}}(3\alpha,4\alpha)$ of the string $\#^m W\$^m\#^m X\$^m\#^m Y\$^m\#^m Z\$^m$ to find $\mathrm{LCS}(W,X,Y,Z)$, where W, X, Y, Z are four strings of equal length n. Hence, there is a multiplicative gap of $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$ in the lower and upper bounds of the time complexities of Theorem 4.4 and Theorem 4.6, respectively.

Since our time bounds compared to the longest square subsequence are rather large, we expect that it should be possible to reduce the time complexities by linear factors. However, in the case that we have tight bounds, as mentioned by Tiskin [32], it could be interesting to study accelerating techniques such as introduced in [13], dividing the time by a poly-logarithmic term.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Numbers JP20H04141 (HB), JP19K20213 (TI), JP22H03551, JP21K17701, and JP23H04378 (DK). We thank the anonymous reviewers for their outstanding work, leading to the definition of the tables G_i, improved time and space complexities, and to the analysis of a lower bound for the longest bordered subsequence.

References

[1] A. Abboud, A. Backurs, V.V. Williams, Tight hardness results for LCS and other sequence similarity measures, in: Proc. FOCS, 2015, pp. 59–78.
[2] A. Agrawal, P. Gawrychowski, A faster subquadratic algorithm for the longest common increasing subsequence problem, in: Proc. ISAAC, in: LIPIcs, vol. 181, 2020, pp. 4:1–4:12.
[3] R.A. Baeza-Yates, Searching subsequences, Theor. Comput. Sci. 78 (2) (1991) 363–376.
[4] H. Bannai, T. I, T. Kociumaka, D. Köppl, S.J. Puglisi, Computing longest (common) Lyndon subsequences, in: Proc. IWOCA, in: LNCS, vol. 13270, 2022, pp. 128–142.
[5] T.C. Biedl, A. Biniaz, R. Cummings, A. Lubiw, F. Manea, D. Nowotka, J.O. Shallit, Rollercoasters and caterpillars, in: Proc. ICALP, in: LIPIcs, vol. 107, 2018, pp. 18:1–18:15.
[6] T.C. Biedl, A. Biniaz, R. Cummings, A. Lubiw, F. Manea, D. Nowotka, J.O. Shallit, Rollercoasters: long sequences without short runs, SIAM J. Discrete Math. 33 (2) (2019) 845–861.
[7] P. Bille, I.L. Gørtz, F.R. Skjoldjensen, Subsequence automata with default transitions, J. Discret. Algorithms 44 (2017) 48–55.
[8] T.M. Chan, M. Patrascu, Counting inversions, offline orthogonal range counting, and related problems, in: Proc. SODA, 2010, pp. 161–173.
[9] P. Charalampopoulos, T. Kociumaka, S. Mozes, Dynamic string alignment, in: Proc. CPM, in: LIPIcs, vol. 161, 2020, pp. 9:1–9:13.
[10] S.R. Chowdhury, M.M. Hasan, S. Iqbal, M.S. Rahman, Computing a longest common palindromic subsequence, Fundam. Inform. 129 (4) (2014) 329–340.
[11] A. Conte, R. Grossi, G. Punzi, T. Uno, Polynomial-delay enumeration of maximal common subsequences, in: Proc. SPIRE, in: LNCS, vol. 11811, 2019, pp. 189–202.
[12] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, MIT Press, 2009.
[13] M. Crochemore, C.S. Iliopoulos, Y.J. Pinzón, J.F. Reid, A fast and practical bit-vector algorithm for the longest common subsequence problem, Inf. Process. Lett. 80 (6) (2001) 279–285.
[14] L. Duraj, A sub-quadratic algorithm for the longest common increasing subsequence problem, in: Proc. STACS, in: LIPIcs, vol. 154, 2020, pp. 41:1–41:18.
[15] K. Fujita, Y. Nakashima, S. Inenaga, H. Bannai, M. Takeda, Longest common rollercoasters, in: Proc. SPIRE, in: LNCS, vol. 12944, 2021, pp. 21–32.
[16] P. Gawrychowski, F. Manea, R. Serafin, Fast and longest rollercoasters, Algorithmica 84 (4) (2022) 1081–1106.
[17] D.S. Hirschberg, A linear space algorithm for computing maximal common subsequences, Commun. ACM 18 (6) (1975) 341–343.
[18] D.S. Hirschberg, Algorithms for the longest common subsequence problem, J. ACM 24 (4) (1977) 664–675.
[19] S. Inenaga, H. Hyyrö, A hardness result and new algorithm for the longest common palindromic subsequence problem, Inf. Process. Lett. 129 (2018) 11–15.
[20] T. Inoue, S. Inenaga, H. Hyyrö, H. Bannai, M. Takeda, Computing longest common square subsequences, in: Proc. CPM, in: LIPIcs, vol. 105, 2018, pp. 15:1–15:13.
[21] T. Inoue, S. Inenaga, H. Hyyrö, H. Bannai, M. Takeda, Computing longest common square subsequences, in: Proc. CPM, in: LIPIcs, vol. 105, 2018, pp. 15:1–15:13.
[22] T. Inoue, S. Inenaga, H. Bannai, Longest square subsequence problem revisited, in: Proc. SPIRE, in: LNCS, vol. 12303, 2020, pp. 147–154.
[23] M. Kiyomi, T. Horiyama, Y. Otachi, Longest common subsequence in sublinear space, Inf. Process. Lett. 168 (2021) 106084.
[24] M. Kosche, T. Koß, F. Manea, S. Siemer, Absent subsequences in words, in: Proc. RP, in: LNCS, vol. 13035, 2021, pp. 115–131.
[25] A. Kosowski, An efficient algorithm for the longest tandem scattered subsequence problem, in: Proc. SPIRE, in: LNCS, vol. 3246, 2004, pp. 93–100.
[26] Y. Sakai, Maximal common subsequence algorithms, Theor. Comput. Sci. 793 (2019) 132–139.
[27] C. Schensted, Longest increasing and decreasing subsequences, Can. J. Math. 13 (1961) 179–191.
[28] S. Schrinner, M. Goel, M. Wulfert, P. Spohr, K. Schneeberger, G.W. Klau, The longest run subsequence problem, in: Proc. WABI, in: LIPIcs, vol. 172, 2020, pp. 6:1–6:13.
[29] S. Schrödl, An improved search algorithm for optimal multiple-sequence alignment, J. Artif. Intell. Res. 23 (2005) 587–623.
[30] A. Tiskin, Semi-local string comparison: algorithmic techniques and applications, arXiv:0707.3619, 2007, http://arxiv.org/abs/0707.3619v21.
[31] A. Tiskin, Semi-local string comparison: algorithmic techniques and applications, Math. Comput. Sci. 1 (4) (2008) 571–603.
[32] A. Tiskin, Personal communication, February 2022.