Doubly-periodic string comparison

Nikita Gaevoy, Boris Zolotov and Alexander Tiskin

Abstract. The longest common subsequence (LCS) problem is a fundamental algorithmic problem. Given a pair of strings, the problem asks for the length of the longest string that is a subsequence in both input strings. In previous works, the third author developed an algebraic framework for the LCS problem and its relatives, based on the Hecke monoid. Among the many algorithmic problems that can be approached efficiently by this technique, there is the natural problem of obtaining the LCS length for a pair of strings, one or both of which have a periodic structure. The case where one of the input strings is periodic has been considered before; in this work, we develop an efficient algorithm for the more general case where both input strings are periodic. This algorithm for doubly-periodic LCS has been engineered by the first author; the resulting code can process doubly-periodic inputs of sizes far beyond the reach of ordinary and singly-periodic LCS algorithms.

Introduction

The longest common subsequence (LCS) problem is a fundamental algorithmic problem. Given a pair of strings, the problem asks for the length of the longest string that is a subsequence in both input strings. The LCS and the equivalent problem of computing the *edit distance* between two strings have found many applications such as computing DIFF of two texts in the corresponding Linux utility, or, generally, the best weighted alignment of two strings, which is widely used in bioinformatics.

LCS-related problems display unexpected ties with semigroup algebra, computational geometry and transposition networks. In previous works, the third author developed an algebraic framework for the LCS problem and its relatives, based on the Hecke monoid. Surprisingly, multiplication in the Hecke monoid is found to reflect closely the behavior of the LCS structure under string concatenation.

Among the many algorithmic problems that can be approached efficiently by this technique, there is the natural problem of obtaining the LCS length for a pair of strings, one or both of which have a periodic structure. The case where one of

the input strings is periodic has been considered before: it is solved by reduction in the affine counterpart of the Hecke monoid.

In this work, we develop an efficient algorithm for the more general case where both input strings are periodic. It works in time $O(mn)$, under mild assumptions, where m and n are the lengths of the periods of these strings. This algorithm for doubly-periodic LCS has been engineered by the first author; the resulting code can process doubly-periodic inputs of sizes far beyond the reach of ordinary and singly-periodic LCS algorithms.

1. The framework

In our previous work [3, 5], we established a connection between string comparison and sticky multiplication of permutations, expressed as $P \boxdot Q = R$. A subquadratic sticky multiplication algorithm (called Steady Ant Algorithm) is given in [5].

Traditionally, the monoid of permutations considered under sticky multiplication is known as the *Hecke monoid* $\mathbb{H}_n$. An element of $\mathbb{H}_n$ can be represented by a *sticky braid* which is analogous to classical braids.

This work extends this connection to infinite periodic strings and affine permutations [2]. Sticky multiplication is generalised directly to affine permutations and expressed as $\tilde{P} \cdot \tilde{Q} = \tilde{R}$. The resulting monoid is known as *affine Hecke monoid* $\tilde{\mathbb{H}}_n$.

An (affine) sticky braid is called *reduced*, if every pair of its strands cross at most once. A sticky braid (finite or affine) can also be viewed as a *transposition network* (introduced as *primitive sorting networks* in [1]).

For a finite permutation P (affine permutation $\tilde{P}$) we choose an arbitrary reduced braid (or, equivalently, a transposition network) realizing this permutation and denote it by $\mathcal{B}(P)$ ($\mathcal{B}(\tilde{P})$, respectively).

Affine replication operator repl_n allows to extend a function $P\colon [0:n) \to \mathbb{Z}$ to an affine permutation $\tilde{P}$ of order n, provided all images of P are pairwisely non-congruent modulo n.

The $\Gamma\Phi$- or $\Phi\Gamma$-decomposition is a representation of an affine permutation $\tilde{P}$ of order n as a product of two affine permutations, one of which is a repl_n of a permutation in S_n, and the other is monotone increasing on $[0:n)$.

2. Affine sticky multiplication

We reduce sticky multiplication of affine permutations $\tilde{P}, \tilde{Q}$ to sticky multiplication of finite permutations. We compute the sticky product of three consecutive periods of $\tilde{P}$, $\tilde{Q}$, taken with respect to the codomain of $\tilde{P}$, which is also the domain of $\tilde{Q}$. We then cut out and replicate the middle period of the resulting product, obtaining $\tilde{P} \boxdot \tilde{Q}$; see Algorithm 1.

We prove the correctness of our algorithm by showing that the injection $Q^* = Q_{(P)}|_{[n:2n)} \cdot \Gamma'$ can be replicated to form an affine permutation, and the result

Algorithm 1 Affine Sticky Multiplication

Input: affine permutations $\tilde{P}$, $\tilde{Q}$ of order n.
Output: affine permutation $\tilde{P} \boxdot \tilde{Q}$.
Description:

1. Compute $\Gamma\Phi_{3n}$-decomposition $\tilde{P} = \Gamma\Phi$ and $\Phi\Gamma_{3n}$-decomposition $\tilde{Q} = \Phi'\Gamma'$. Let $P = \Phi|_{[0:3n)}$, $Q = \Phi'|_{[0:3n)}$.
2. Compute sticky product $P \boxdot Q$. Compute $Q_{(P)} = P^{-1} \cdot (P \boxdot Q)$.
3. Compute $Q^* = Q_{(P)}|_{[n:2n)} \cdot \Gamma'$. This is an injection $[n:2n) \to \mathbb{Z}$. We have $\tilde{Q}_{(\tilde{P})} = \mathrm{repl}_n\, Q^*$.
4. Output the affine permutation $\tilde{P} \cdot \tilde{Q}_{(\tilde{P})}$.

of the replication provides the correct value for $\tilde{Q}_{(\tilde{P})}$. Equivalently, Q^* coincides with a single period of $\tilde{Q}_{(\tilde{P})}$, that is, $Q^* = \tilde{Q}_{(\tilde{P})}|_{[n:2n)}$ as functions $[n:2n) \to \mathbb{Z}$.

We do this by showing that $\mathcal{B}(\tilde{Q})$ can simulate the behavior of $\mathcal{B}(Q)$ up to a subsequent decompression by Γ', if it is prohibited from untangling intersections between certain trajectories, while keeping the images of elements within $[n:2n)$ unchanged.

The time complexity of this algorithm is $O(n \cdot \log n)$ as it uses the Steady Ant Algorithm for computing the sticky product of two finite permutations, and otherwise only sorts preimage-image pairs of permutations within a period and takes standard compositions of permutations.

3. Algorithm for doubly-periodic LCS

Let $A = a^r$, $B = b^s$ be finite periodic strings of lengths $M = rm$, $N = sn$ with periods a, b, respectively. To find $\mathrm{LCS}(a, b)$, we:

1. extend B to an infinite n-periodic string b^∞,
2. obtain a skew-affine embedded braid for a vs. b^∞ by placing crosses and elbow joints on the grid in a straightforward manner,
3. reduce this braid using wraparound combing [4], compute the corresponding affine permutation $\tilde{P}$ describing implicitly the LCS of a vs. b^∞,
4. compute $\tilde{P}^{\boxdot r}$ by binary exponentiation using Algorithm 1 as a subroutine,
5. count the elements of $\tilde{P}^{\boxdot r}$ that fall within the bounds of B respecting the affine structure of the braid.

Acknowledgements

This work is dedicated to the memory of our late teacher and colleague Prof. Nikolai Vavilov, whose advice and inspiration have guided us from the very early stages of our research.

We would also like to thank Vladislav Makarov and Ivan Bochkov for their help with the engineering part of the work.

The work of B. Zolotov was performed at the Saint Petersburg Leonhard Euler International Mathematical Institute and supported by the Ministry of Science and Higher Education of the Russian Federation (agreement no. 075–15–2022–287).

References

[1] D. E. Knuth. *The Art of Computer Programming: Sorting and Searching*, volume 3. Addison Wesley, 1998.

[2] Joel Brewster Lewis. Affine symmetric group. *WikiJournal of Science*, 4(1):3, 2021.

[3] A. Tiskin. Semi-local string comparison: Algorithmic techniques and applications. *Mathematics in Computer Science*, 1(4):571–603, 2008.

[4] A. Tiskin. Periodic String Comparison. In *Proceedings of CPM*, volume 5577 of *Lecture Notes in Computer Science*, pages 193–206, 2009.

[5] A. Tiskin. Fast Distance Multiplication of Unit-Monge Matrices. *Algorithmica*, 71:859–888, 2015.

Nikita Gaevoy
Department of Mathematics and Computer Science
Saint Petersburg University, Russia and
Technion, Israel
e-mail: nikgaevoy@gmail.com

Boris Zolotov
Department of Mathematics and Computer Science
Saint Petersburg University
St. Petersburg, Russia
e-mail: boris.a.zolotov@yandex.com

Alexander Tiskin
Department of Mathematics and Computer Science
Saint Petersburg University
St. Petersburg, Russia
e-mail: alextiskin@gmail.com