WAB

Provadis School of International Management and Technology

Exposé

Comparing Suffix Automata Against Suffix Arrays For Longest Common Substring Queries

When "Linear Time" is Not Enough

Rubin Chempananickal James
rubin.chempananickal-james@stud-provadis-hochschule.de
Matriculation Number: D876

Department: Information Technology
Module: Algorithmen und Datenstrukturen
Reviewer: Prof. Dr. Volker Scheidemann

February 18, 2026

Contents

Glossary

DNA	Deoxyribonucleic Acid, the molecule that carries genetic information in living organisms. 1
RNA	Ribonucleic Acid, a molecule that is transcribed from DNA and which then gets translated into proteins. 1

Exposé

Introduction

The Longest Common Substring (LCS) problem is a fundamental problem in computer science, with applications in multiple domains, bioinformatics being one of the most prominent. This is due to the fact that DNA and RNA sequences are usually encoded as strings for bioinformatics workflows, in a format called FASTA, originally described by Pearson and Lipman in 1985[1].

The LCS problem is defined as follows:

> Given two strings S and T, each of length at most n, the longest common substring (LCS) problem is to find a longest substring common to S and T.[2]

For example, given the strings "AG**CTAGC**" and "T**CTAGC**TA", the longest common substring is "CTAGC", which has a length of 5. The LCS problem can be solved using various algorithms, such as dynamic programming, suffix trees, suffix arrays, and suffix automata, each with different time and space complexities.

Of particular interest are the Suffix Automaton (SAM) (also known as Directed Acyclic Word Graph (DAWG)) and the Suffix Array (SA) with Longest Common Prefix (LCP) Array, which both yield the search result in linear time, but differ in their construction time and space requirements.

Research Question

The primary research question of this paper is:

> How do the Suffix Automaton and the Suffix Array (SA) with LCP Array compare in terms of time and space complexity when solving the Longest Common Substring problem in an everyday programming context?

Python[3] was chosen as the programming language for this paper due to the fact that it is by far the most popular programming language for bioinformatics[4].

[1]Lipman and Pearson 1985.
[2]Amir et al. 2020.
[3]Van Rossum and Drake 2009.
[4]Mariano et al. 2020.

Objectives

The objectives of this paper are as follows:

- To implement the Suffix Automaton and the Suffix Array (SA) with Longest Common Prefix (LCP) Array algorithms for solving the Longest Common Substring problem.
- To evaluate the time and space complexity of both algorithms using a Python script that measures their performance on a variety of test cases.
- To analyze the results and draw conclusions about the trade-offs between the two approaches in terms of their efficiency and practicality for real-world applications.

Methodology

The SAM and SA + LCP Array algorithms will be implemented in Python, and only synthetic test cases will be used to evaluate their performance. The amount of memory consumed by the algorithms will be measured using the `memory_profiler` library, while the time taken by the algorithms will be measured using the `timeit` library. To ensure a fair comparison, both algorithms will be tested on the same set of input strings and on the same machine, and the results will be averaged over multiple runs to account for any variability in performance.

Planned Structure

The paper will likely be structured as follows:

- Introduction
- Research Question and Objectives
- Literature Review
- Methodology
- Results and Discussion
- Limitations
- Conclusion and Future Work
- References
- AI Declaration
- Declaration of Authorship

Bibliography

Amir, Amihood et al. (2020). „Dynamic and internal longest common substring“. In: *Algorithmica* 82.12, pp. 3707–3743.

Blumer, A. et al. (1985). „The smallest automation recognizing the subwords of a text“. In: *Theoretical Computer Science* 40. Eleventh International Colloquium on Automata, Languages and Programming, pp. 31–55. ISSN: 0304-3975. DOI: `https://doi.org/10.1016/0304-3975(85)90157-4`. URL: `https://www.sciencedirect.com/science/article/pii/0304397585901574`.

Lipman, David J. and William R. Pearson (1985). „Rapid and Sensitive Protein Similarity Searches“. In: *Science* 227.4693, pp. 1435–1441. DOI: `10.1126/science.2983426`. eprint: `https://www.science.org/doi/pdf/10.1126/science.2983426`. URL: `https://www.science.org/doi/abs/10.1126/science.2983426`.

Manber, Udi and Gene Myers (1990). „Suffix arrays: a new method for on-line string searches“. In: *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA ’90. San Francisco, California, USA: Society for Industrial and Applied Mathematics, pp. 319–327. ISBN: 0898712513.

Mariano, Diego et al. (2020). „A Brief History of Bioinformatics Told by Data Visualization“. In: *Advances in Bioinformatics and Computational Biology*. Ed. by João C. Setubal and Waldeyr Mendes Silva. Cham: Springer International Publishing, pp. 235–246. ISBN: 978-3-030-65775-8. URL: `https://bioinfo.dcc.ufmg.br/history/`.

Van Rossum, Guido and Fred L. Drake (2009). *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace. ISBN: 1441412697.