The All-pairs Suffix-Prefix Matching Problem

William H. A. Tustumi (IC), Guilherme P. Telles (PQ), Felipe A. Louza (PG)

Abstract

The all-pairs suffix-prefix matching is a very important problem in string processing. Different solutions have been proposed to this problem. We present a new and improved algorithm that is 2.6 times faster and uses 15% less memory than the previous best known solution.

Key words: Suffix-prefix matching, Suffix array, LCP array

Introduction

The all-pairs suffix-prefix matching (APSP) is an important problem in string processing having application in the context of DNA sequencing. Given a set of k strings \( \{S_1, S_2, \ldots, S_k\} \), the APSP is the problem of finding, for all pairs \( S_i \) and \( S_j \), the longest suffix of \( S_i \) that is a prefix of \( S_j \).

This problem has been solved optimally by Gusfield et al.\(^1\) in 1992 (using suffix trees) and almost 20 years later by Ohlebusch Gog\(^2\) in 2010 (using enhanced suffix arrays). The later is about 3 times faster than the one that uses suffix trees.

Results and Discussion

We propose a new optimal algorithm that is faster and more space-efficient in practice. Our algorithm scans the enhanced suffix arrays in a different way and uses a different auxiliary data structure.

The algorithm was implemented in C++ using sdsl-lite library\(^3\). We used real DNA sequences of the EST database from C. elegans. We compared the performance of our algorithm with the algorithm OG\(^2\), best known solution.

Image 1 shows the total running time (in seconds) of each algorithm.

![Image 1. Running time performed by each algorithm.](image1.png)

Image 2 shows the amount of memory used by each algorithm.

![Image 2. Memory consumption of each algorithm.](image2.png)

One can see that our algorithm have outperformed algorithm OG by a factor of 2.6 on the average, and the total memory used by our algorithm was 15% less on average.

Conclusions

We presented a faster and more space-efficient algorithm to solve the APSP. Our algorithm can be easily parallelized and modified to work in semi-external fashion.

Acknowledgement

The project was financed by CNPQ (grant No 118372/2014-9).

