An Empirical Analysis of Data Streaming Algorithms

Dominic Moylett, supervised by Dr. Raphaël Clifford, Dr. Markus Jalsenius and Dr. Benjamin Sach

University of Bristol, Department of Computer Science

Introduction

In the wake of Big Data, a popular area of research in Theoretical Computer Science is the data streaming model. Under this model, the algorithm only has access to a window of the input, and the objective is to compute the solution with little space. There have been a number of developments in data streaming which work well in theory, but many have never been implemented in practice.

I have implemented two algorithms in C for stream-based pattern matching. Note the standard notation, where m is the length of the pattern and Σ is the alphabet:

1. An algorithm for parameterised matching with $O(m + |\Sigma|)$ space and $O(\log(\min(m, |\Sigma|))$ amortised time per character by Amir et al.[a]
2. An algorithm for exact matching with $O(\log m)$ space and $O(1)$ time per character by Breslauer and Galil,[b] and deamortised by Simon's Algorithm.[c] Sublinear space is achieved via Karp-Rabin fingerprints.

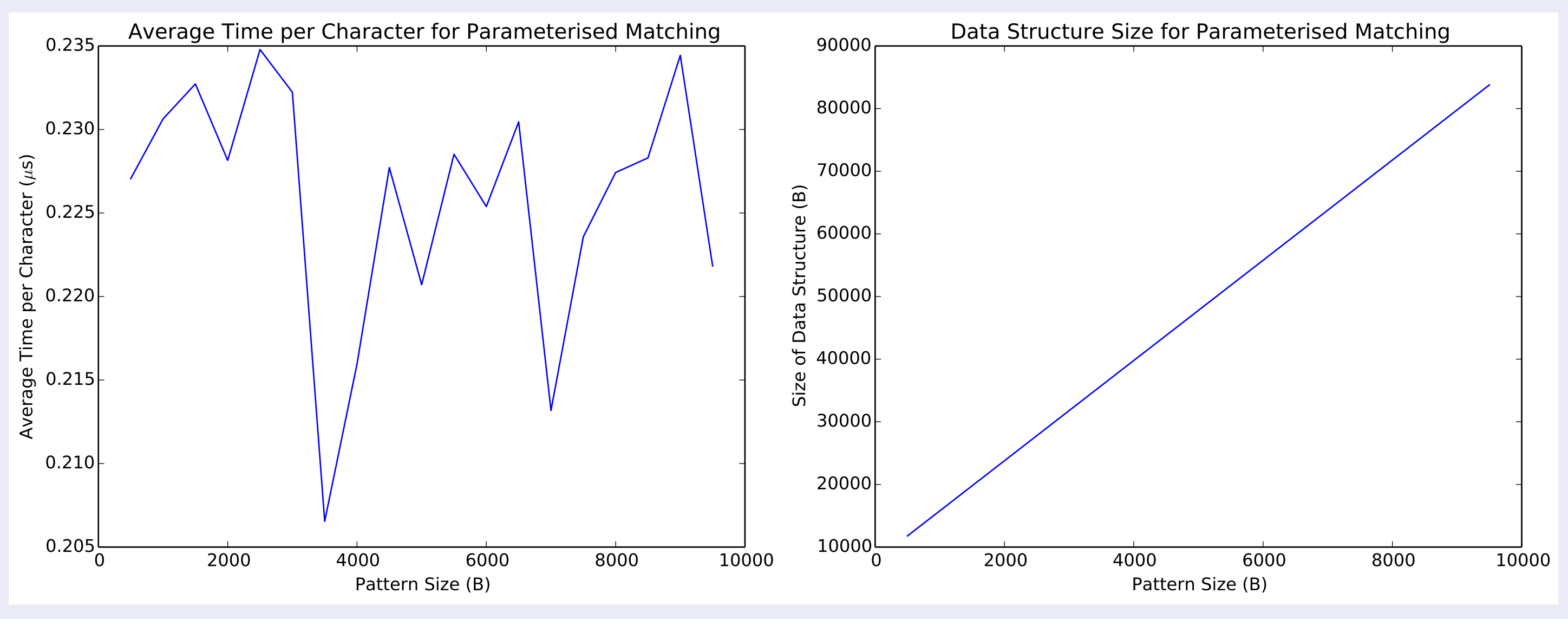
All of these algorithms have been tested on 50MB of English text from the Pizza and Chili Corpus.

[a] Alphabet Dependence in Parameterized Matching by Amihood Amir, Martin Farach and S. Muthukrishnan

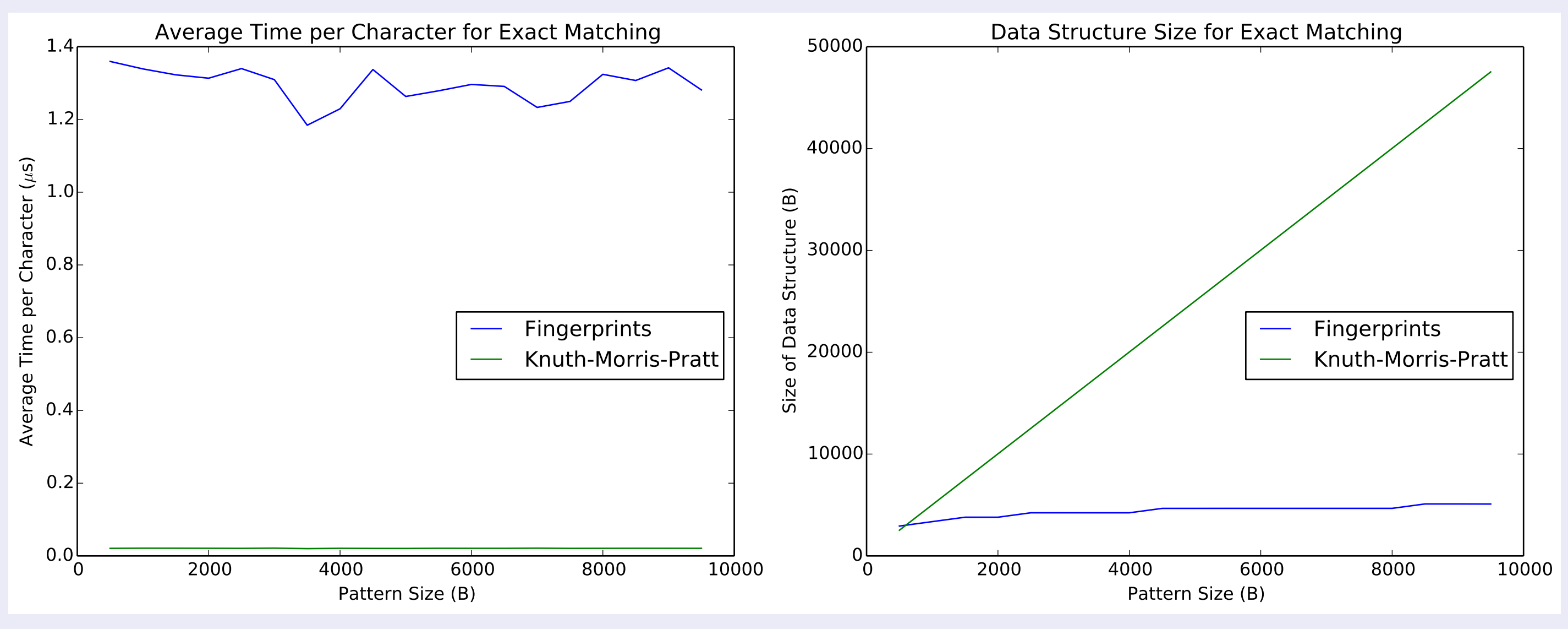
[b] Real-Time Streaming String-Matching by Dany Breslauer and Zvi Galil

[c] String Matching Algorithms and Automata by Imre Simon

Parameterised Matching Results



Results for Exact Matching



Parameterised Matching Conclusions

The main bottleneck in Parameterised Matching is that previous occurances of characters in the text are stored in a search tree, which takes $O(\log \pi)$ time to query and edit.

Question: Can this be improved with dynamic hashing, for which the best results are $O(\sqrt{\log \pi / \log\log \pi})$?

Exact Matching Conclusions

Exact matching with Fingerprints takes 60-70 times longer per character and a significantly longer build time than Knuth-Morris-Pratt, but requires less space than even storing the pattern.

Question: The practical bottleneck is that the fingerprints use a lot of modular multiplications. Could these be optimised via precomputation or Montgomery Multiplication?