First meeting (06-09-2022): The string indexing with compressed pattern problem

Mathias Søndergaard, s174426

September 5, 2022

What has been done initially

Project initiation. The readings I have done include reviewing old material on the LZ77 compression scheme and Suffix trees, as well as constructing these in $O(n)$ time (with constant sized alphabets for suffix trees). For the LZ77 scheme the KKP3 algorithm shown in [2] was reviewed (but not fully understood). Also I have begun reading and understanding [3], now reaching section 4 (so I have just covered the "simple" Phrase trie section). During this time, I also reviewed material on NCA and LCP. This included looking into on how to construct these in $O(n)$ time, however it took me some time to get hold of [1], alas the material has not been fully reviewed.

Implementation / Programming wise, the LZ77 implementation running in $O(n)$ time and space found at https://www.cs.helsinki.fi/group/pads/lz77.html has been tested after updating the code to work on my system. Fortunately, this code includes construction of suffix arrays in $O(n)$ time and space. I have implemented suffix trees running in $O(m)$ time and $O(n)$ space. The phrase trie with $O(n^2)$ space and $O(z + log(n) + occ)$ time has also been implemented. For now I chose a naive construction algorithm for both trees resulting in $O(n^2)$ prepossessing time.

Plans for the next weeks

For the next meeting the goal is to have implemented the Space efficient Phrase Trie found in section 4 of [3] and if time allows, also look into replacing the naive prepossessing algorithms mentioned above with optimal prepossessing algorithms. However this last part requires me to also implement NCA/LCP in an optimal manner, which requires a bit more work.

Questions for this meeting

For now, I have no theory related questions. I would like to discuss my strategy moving forward. The implementation aspect of this project has peaked my interest greatly. But time may become an issue, which is why I have initially chosen non-optimal construction methods for the data structures. This allows me to cover more ground in the beginning, and my plan is to return to the prepossessing later. Is this the correct way to go about things? Or should I implement the data structure in its full power before moving on? Obviously a report also needs to be written - is it required of me to "invent" new theory on this subject, or is reviewing the theory I use in a concise manner enough? Thus letting the implementation be the novel part of this thesis.

References

[1] D. Harel et al. "Fast Algorithms for finding nearest common ancestors". In: (1984).

[2] D. KEMPA et al. "Lazy Lempel-Ziv Factorization Algorithms". In: (2016).

[3] Philip Bille, Inge Li Gørtz, and Teresa Anna Steiner. "String Indexing with Compressed Patterns". In: (2020).

Problem Description: The string indexing with compressed pattern problem

Mathias Søndergaard, s174426

September 5, 2022

Problem definition

Given two strings, S and P, the string indexing problem consists of finding all starting positions of P in S. Commonly P is known as the pattern with $|S| > |P|$. In this thesis we shall investigate the problem where P is compressed, which is known as the string indexing with compressed pattern problem (SICP). We will use the LZ77 compression scheme to compress P, with $LZ(P)$ denoting the phrases obtained by compressing P. A typical use case which is affected by the SICP problem is in the client-server scenario. Here the server contains S. In order to save bandwidth, the client sends a request which includes $LZ(P)$, thus having the server do the indexing.

Previous work

A naive solution would be to decompress $LZ(P)$, transforming the SICP problem instance to a string indexing instance. Let $m = |P|$ and let occ denote the number of occurrences of P in S. The naive method results in $O(m + occ)$ time, as $LZ(P)$ has to be decompressed and the indexing is done using a suffix tree. Both require $O(m)$ and reporting occ elements require $O(occ)$. $O(n)$ space is required for the suffix tree, with $n = |S|$. Novel solutions exist which does not decompress $LZ(P)$ while obtaining better time bounds. These are covered in [2], with the main finding being a data structure using $O(n)$ space and $O(z + log(m) + occ)$ time. Here $z = |LZ(P)|$.

Goals of the project

The primary goal of this project is to review and implement algorithms / data structures for the string indexing with compressed pattern problem. These will be based on state of the art theory, given in [2]. Moreover, they will be tested and bench-marked, investigating whether the theory holds in practice. To reach this goal, other data structures and algorithms will be covered and implemented, including Nearest common ancestor, Longest common prefix, LZ77, Suffix trees, ART-decomposition ([1]) and their optimal prepossessing algorithms.

References

[1] Stephen Alstrup, Thore Husfeldt, and Theis Rauhe. “Marked ancestor problem”. In: (1998).

[2] Philip Bille, Inge Li Gørtz, and Teresa Anna Steiner. “String Indexing with Compressed Patterns”. In: (2020).

Project Plan: String indexing with compressed pattern problem

Week	Implementation	Thesis	Reading
1	LZ77, Suffix Array & Tree, Phrase Trie V1		
2	Phrase Trie V2 (Linear space)		Section 4
3	DS for section 5		Section 5
4	DS for section 5		
5	DS for section 6		Section 6
6	DS for section 6		
7	Buffer	Buffer	Buffer
8	**Fall Break**	**Fall Break**	**Fall Break**
9	Fix preprocessing		O(n) algorithms for NCA/LCP + ST
10	Fix preprocessing		[Preprocessing of section 5-6]
11		Preliminaries (ST, LZ77)	ST + LZ77 Material
12		Review section 4	
13		Review section 5	
14		Review section 6	
15	Testing, gather data, figures		
16		Results	
17		Conclusion, introduction, abstract, etc	
18		Proof reading, final touches	
19	?	?	?
20	?	?	?

* When refering to sections, it is a reference to sections in "String indexing with compressed patterns".