Enhanced and Extended Suffix Arrays

Adrian Regenfuß

June 22, 2020

Abstract

In this report, I review the literature on enhanced and extended suffix arrays in the context of searching long strings. I examine the different algorithms used for both constructing enhanced and extended suffix arrays and for using them in searching long strings. In the end, I compare enhanced and extended suffix arrays with suffix arrays and suffix trees.

Introduction

Finding the occurences of one string in another string, longest repeated substrings and longest shared substrings of two different strings are fundamental problems for many kinds of computing systems.

As a result, many different algorithms have been developed for these kinds of problems: For finding the occurences of one string in another one the naive algorithm and the Boyer-Moore algorithm (Boyer and Moore 1977), and for all three of these problems (and more) three different data structures: the suffix tree (Weiner 1973), the suffix array (Manber and Myers 1993) and the enhanced suffix array (Abouelhoda et al. 2002).

Suffix trees, suffix arrays and enhanced suffix arrays have the disadvantage of requiring to be constructed for a specific string, which has time and space requirements. Because of this, they are better suited for tasks where immutable strings have to be searched or matched repeatedly, although there has been some work to extend the suffix array to dynamic strings (Salson et al. 2010).

Since searching and matching very long immutable strings is very common in genome analysis, it doesn't surprise that both suffix arrays and enhanced suffix arrays were developed in that context.

This report first describes the enhanced suffix array as a data structure, then sketches the algorithms used for constructing it, and afterwards describes different string matching problems and how they are solved by enhanced and extended suffix arrays. Finally, it compares enhanced suffix arrays to normal suffix arrays and suffix trees, and closes with an overview of tools that implement enhanced and extended suffix arrays.

Enhanced and Extended Suffix Arrays

Enhanced suffix arrays were first proposed in Abouelhoda et al. 2002 as an improvement over normal suffix arrays. An enhanced suffix array contains a

suffix array together with the LCP-array of the string, and sometimes a Burrows-Wheeler transformation and an inverse of the suffix table.

For the following, let S be a finite string of length n over the finite alphabet Σ.

suftab

The suffix array suftab is an array of integers describing the positions of of sorted suffixes of S in S.

More formally, let Suf_S be the set of suffixes of S. Let then SortSuf_S be the array of lexically sorted suffixes of S. Then suftab$[i] = k$ if and only if $S[k..n] = \text{SortSuf}[i]$.

lcptab

The LCP (longest common prefix) table describes the length of the longest common prefix of two neighbouring entries in the array of sorted suffixes.

Formally, lcptab$[i] = k$ iff $\text{SortSuf}[i][0..k] = \text{SortSuf}[i-1][0..k]$. The zeroth entry in lcptab is always 0.

bwttab

Informally, bwttab contains the character before the suffix in suftab.

suftab^{-1}

Example

For example, let $S =$ "cagccacat". Then suftab, lcptab, bwttab, suftab^{-1} and suftab are the following:

i	suftab	lcptab	SortSuf
0	5	0	acat$
1	1	1	agccacat$
2	7	1	at$
3	4	0	cacat$
4	0	2	cagccacat$
5	6	2	cat$
6	3	1	ccacat$
7	2	0	gccacat$
8	8	0	t$
9	9	0	$

LCP-Interval Trees

Construction

Different String Matching Problems

A plethora of different string matching problems have been identified by computer scientists, for many of which suffix arrays and enhanced suffix arrays are useful.

Abouelhoda et al. 2004, pg. 2 summarizes suffix tree applications from Gusfield 1997, chap. 2 and classify them after their type of tree traversal.

Exact String Matching

Description

Give a string S of length n and a string T of length m with $m \leq n$ both using them same alphabet Σ, the exact matches of T in S is the set of indices $I = \{i_1, \ldots, i_k\}$ for which holds that $\forall i \in I : S[i..i+m] = T$.

In other words, I is the set of indices where T is a substring in S.

Suffix Array Algorithm

Manber and Myers 1993 describes an exact string matching algorithm that runs in $\mathcal{O}(m \log |\Sigma)$ time and constant space.

Their proposed algorithm uses binary search to sequentially find the first (smallest) index L_W and the last (biggest) index R_W in suftab so that $S[\text{suftab}[L_W]]$ and $S[\text{suftab}[R_W]]$ have the prefix T.

First, it searches L_W using binary search on the whole array suftab and then uses L_W as a left boundary to find R_W, which often improves runtime as opposed to two independent searches over the whole array, although the latter might be easier to parallelize.

Extended Suffix Array Algorithm

Manber and Myers 1993 then propose a speed improvement based on longest common prefixes. Their method attempts to reduce the number of single-character comparisons by only comparing characters that occur after the longest common prefix of T and $S[M_W]$ (M_W being the index in the middle between L_W and R_W).

I have not come across a proposal to use interpolation search first described in Perl et al. 1978 to search the suffix array with an improved $\mathcal{O}(\log \log n)$ runtime. This perhaps stems from the fact that interpolation search assumes uniform distribution of the alphabet, and has a worst-case runtime of $\mathcal{O}(n)$ runtime. It still might be useful to empirically test speed differences in binary and interpolation search.

Supermaximal Repeats

Description

Enhanced Suffix Array Algorithm

Comparison

Applications

Conclusion

References

Mohamed Ibrahim Abouelhoda, Stefan Kurtz, and Enno Ohlebusch. The enhanced suffix array and its applications to genome analysis. In *International Workshop on Algorithms in Bioinformatics*, pages 449–463. Springer, 2002.

Mohamed Ibrahim Abouelhoda, Stefan Kurtz, and Enno Ohlebusch. Replacing suffix trees with enhanced suffix arrays. *Journal of discrete algorithms*, 2(1): 53–86, 2004.

Robert S Boyer and J Strother Moore. A fast string searching algorithm. *Communications of the ACM*, 20(10):762–772, 1977.

Dan Gusfield. Algorithms on strings, trees, and sequences. 1997. *Computer Science and Computational Biology. New York: Cambridge University Press*, 1997.

Udi Manber and Gene Myers. Suffix arrays: a new method for on-line string searches. *siam Journal on Computing*, 22(5):935–948, 1993.

Yehoshua Perl, Alon Itai, and Haim Avni. Interpolation search—a log log n search. *Communications of the ACM*, 21(7):550–553, 1978.

Mikaël Salson, Thierry Lecroq, Martine Léonard, and Laurent Mouchard. Dynamic extended suffix arrays. *Journal of Discrete Algorithms*, 8(2):241–257, 2010.

Peter Weiner. Linear pattern matching algorithms. In *14th Annual Symposium on Switching and Automata Theory (swat 1973)*, pages 1–11. IEEE, 1973.