Purely functional palindromic trees

Timur Khazhiev
Innopolis University
t.khazhiev@innopolis.ru

Nikolai Kudasov*
Innopolis University
n.kudasov@innopolis.ru

ABSTRACT

This is a BS-3 Project proposal and it contains a general project description, motivation and a brief plan of execution.

CCS CONCEPTS

• **Theory of computation → Data structures design and analysis**;

KEYWORDS

palidrome, eertree, purely functional, persistent

1 INTRODUCTION

A palindrome is a string that reads the same both ways. Palindromic patterns appear in many research areas, from formal language theory to molecular biology.

There are a lot of papers introducing algorithms and data structures to facilitate different problems that involve palindromes. One such data structure is EERTREE, a recently described linear-sized palindromic tree introduced by Rubinchik [4].

In this project we aim to design at least one purely functional and fully persistent version of a palindromic tree, implement it in Haskell programming language and compare it with other existing solutions. We are going to start with a naive implementation and gradually arrive at an efficient version, relying on some of the techniques described by Okasaki [3] for designing purely functional data structures.

We hope that purely functional variations will prove valuable for some divide-and-conquer approaches to palindromic analysis. We also believe that a fully persistent version might be useful for comparative analysis of closely-related strings (such as RNA string mutations).

2 WORK PLAN

2.1 Iteration I (Feb 11 - Feb 25)

Analyse EERTREE data structure closely. Collect common palindrome-related problems. Research alternative algorithms and data structures for those problems. Investigate existing approaches to palindromic analysis. Form a list of important operations on EERTREE to compare for time and space complexity.

2.2 Iteration II (Feb 25 - Mar 11)

Propose a purely functional, fully persistent palindromic tree. Analyse time and space complexities for the most important operations (both worst-case and amortized). Compare with the original palindromic tree.

2.3 Iteration III (Mar 11 - Mar 25)

Implement EERTREE and the purely functional palindromic tree. Compare implementations on sample problems. Analyse purely functional implementation performance.

2.4 Iteration IV (Mar 25 - Apr 08)

Investigate options to improve performance for the purely functional palindromic tree. Consider these options:

- fusion/deforestation optimisations;
- cache-oblivious model optimizations;
- linked vs. vector-based implementations;
- asymptotic improvements via partial evaluation;
- optimisations for small alphabet strings.

2.5 Iteration V (Apr 08 - Apr 22)

Design a generalised interface to the data structure, relying on parametric polymorphism and higher-order functions to facilitate reusability.

2.6 Iteration VI (Apr 22 - May 06)

Finalise the project by creating a Cabal package complete with examples, tests, code comments, documentation and, perhaps, a formal verification.

3 ITERATION RESULTS

3.1 Iteration I

During iteration I github repository[1] was created where all related to the topic materials were collected. In general this iteration was successful: the analysis of data structure revealed a lack of functionality, interesting problems was found and overview with comparison of similar approaches was made. Detailed results for this iteration can be found in separate pdf file[2] in github repository.

REFERENCES

[1] Timur Khazhiev. 2019. DS project repo. https://github.com/khazhix/ds-project. (2019).

[2] Timur Khazhiev. 2019. Iteration I detailed report. https://github.com/khazhix/ds-project/blob/iteration_i/Iteration_I.pdf. (2019).

[3] Chris Okasaki. 1998. Purely Functional Data Structures. (01 1998). https://doi.org/10.1017/CBO9780511530104

[4] Mikhail Rubinchik and Arseny M. Shur. 2018. EERTREE: An efficient data structure for processing palindromes in strings. *European Journal of Combinatorics* 68 (2018), 249 – 265. https://doi.org/10.1016/j.ejc.2017.07.021 Combinatorial Algorithms, Dedicated to the Memory of Mirka Miller.

*Project supervisor.