Purely functional palindromic trees
Iteration II results

Timur Khazhiev
Innopolis University
t.khazhiev@innopolis.ru

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

ABSTRACT

This is the results of BS-3 Project and it contains a general project results so far.

CCS CONCEPTS

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

KEYWORDS

palindrome, eertree, purely functional, persistent

1 2 SIDED TREE

As the original tree holds only maximum palindrome suffix, we can only append a new symbol. But what if we want to add a new symbol at the beginning? To allow that we can add maximum palindrome prefix support. So we might need to keep track of max prefixes, but as palindrome is symmetric, maximum palindrome suffix of palindrome is maximum palindrome prefix as well. Thus we can simply keep information about maximum palindrome prefix of processed string (as we keep information about maximum suffix). So no major memory usage is added, only one more entry.

This extension allows us merge two eertrees more efficient than simple naive approach (adding symbol by symbol from string of eertree_2 to eerteee_1). Let's consider example: "*abbabba*" + "*bbbab*". Palindromes of concatenated string includes all processed palindromes of both strings and new ones. It's pretty obvious that new palindromes can be found somewhere around joint of two strings.

There are three cases: (1) center of new palindromes is joint (2) center of new palindromes is max suffix (and/or its max suffixes) of first string (3) center of new palindromes is max prefix (and/or its max prefixes) of second string. For current example blue represents max palindrome prefix, red represents max palindrome suffix and black represents symbols that we are checking: "*abbabba*" + "*bbbab*"

(1) "*abbabba*" + "*bbbab*", "*ab*" is not a palindrome, stop here.
(2) _"*abbabba*" + "*bbbab*", nothing to compare, moving to next max suffix
"*abbabba*" + "*bbbab*", "*babbab*" is potential new palindrome, expanding more
"*abbabba*"+"*bbbab*", "*bbabbabb*" is potential new palindrome, expanding more
"*abbabba*"+"*bbbab*", "*abbabbabbb*" is not a palindrome, moving to next max suffix.
"*abbabba*" + "*bbbab*", "*bab*" is potential new palindrome, expanding more
"*abbabba*" + "*bbbab*", "*bbabb*" is potential new palindrome, expanding more
"*abbabba*" + "*bbbab*", "*abbabbb*" is not a palindrome, stop here.
(3) "*abbabba*" + "*bbbab*", "*abbba*" is potential new palindrome, expanding more
"*abbabba*" + "*bbbab*", "*babbbab*" is potential new palindrome, expanding more
"*abbabba*" + "*bbbab* ", nothing to compare, moving to next max prefix
"*abbabba*" + "*bbbab*", "*abbb*" is not a palindrome, moving to next max prefix
"*abbabba*" + "*bbbab*", "*abb*" is not a palindrome, stop here

All found palindromes are added to the a tree. Max palindrome suffix of merged tree is max palindrome suffix of second string. If one of new found palindromes includes whole second string, it becomes max palindrome suffix. And visa-versa max palindrome prefix of merged tree is max palindrome prefix of first string. If one of new found palindromes includes whole first string, it becomes max palindrome prefix. Thus max suffix can be same sequence as max prefix. Nabludenie! Also number of palindromes which includes whole string is exactly one?

2 INFINITE TREE

Some problems does not require information about original string as an output, just raw palindromes information. For some alphabet σ we can simply generate palindromes of length n. The number of palindromes of length n will be $g(n, \sigma)$ where

$$g(n,\sigma) = \begin{cases} \sqrt{\sigma^n} \cdot \sigma & n \text{ is odd} \\ \sqrt{\sigma^n} & n \text{ is even} \end{cases}$$

That tree will represent all possible palindromes and their dependencies. So the number nodes of infinite tree for fixed alphabet will be $\sum_{n=1}^{\infty} g(n, \sigma)$. Here where haskell's lazy evaluation will come in handy.

REFERENCES

*Project supervisor.