Purely functional palindromic trees
Iteration I 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 PROBLEMS ON PALINDROMES

There are a lot of problems on palindromes can be found. Most of them are problems for programming contests but also there are practical applications of palindromes in genetics. We collected some problem statements.

Problem 1

This problem is taken from e-olymp online judge [1] Consider an arbitrary string g. We will call this string as palindrome generator. The set of palindromes $P(g)$ that are generated by this string is determined as follows.

Let string length be n. For all i from 1 to n in $P(g)$ strings $g[1..i]g[1..i]^r$ and $g[1..i]g[1..i-1]^r$ are included, where α^r means α, written in reversed order.

For example if g = "$olymp$", then $P(g)$ = {"oo", "o", "$ollo$", "olo", "$olyylo$", "$olylo$", "$olymmylo$", "$olymylo$", "$olymppmylo$", "$olympmylo$"}.

For a given generator of palindromes g and the string s, it is required to find the number of occurrences of string from $P(g)$ in s as substrings. Namely, it is required to find the number of pairs (i, j) such that $s[i..j] \in P(g)$.

Example input	Example output
olymp olleolleolympmyolylomylo	7

Problem 2

This problem was taken from SPOJ online judge[6]. Each palindrome can be always created from the other palindromes, if a single character is also a palindrome. For example, the string "$malayalam$" can be created by some ways:

$$malayalam = m + ala + y + ala + m$$
$$malayalam = m + a + l + aya + l + a + m$$

We want to take the value of function $NumPal(s)$ which is the number of different palindromes that can be created using the string S by the above method. If the same palindrome occurs more than once then all of them should be counted separately.

Example input	Example output
malayalam	15

Problem 3

This problem was taken from SPOJ online judge[5]. For a given string S we want to find minimum number of continuous palindromes in which this string can be broken.

Example input	Example output
abacdc	2
ababa	1
ababbacababbad	5
abcd	4

Problem 4

This problem was taken from informatics online judge[2]. For a given string S we want to find total number of continuous palindromes in which this string can be broken.

Example input	Example output
aaa	6
aba	4

Problem 5

For a given string s we want to find non palindrome of length n between 2 palindromes of length m. Can be practical for searching potential hairpins.

Example input	Example output
aaabcdaaa	aaa bcd aaa

Problem 6

For a given string s we want to find closest palindrome by edit distance.

Example input	Example output
abcda	abcba
abab	ababa

*Project supervisor.

2 ALTERNATIVE ALGORITHMS AND DATA STRUCTURES

In original paper [4] author referenced several solutions such as affix tree, affix arrays and Manacher's algorithm.

2.1 Affix tree and affix array

Affix tree[7] is just a combined suffix tree and reversed prefix tree in one structure. This combination allows us to match pattern from left to right, from right to left and even inside out.

Affix array[8] is equivalent to the affix tree with respect to its functionality, but with smaller memory requirements.

This data structure is used in genetics for searching different patterns in molecular sequences.

2.2 Manacher's algorithm

Manacher's algorithm[3] has one single application. It is used to find the longest palindromic substring in any string online without any structure behind it.

3 EERTREE DATA STRUCTURE

Motive problem for this structure is to find and count all distinct subpalindromes. Both methods mentioned above can be used for solving this problem but in a complex manner. Eertree is capable of solving this problem in easy manner with same asymptotics. Moreover it was found that eertree has more applications.

This data structure allows to keep track of palindrome substrings (also in an online fashion) in linear time and space. The eertree for a given string is a directed graph with nodes containing palindrome substring of string and two types of edges: insertion edge (weighted edge) and maximum palindromic suffix (unweighted).

In the basic version of eertree interface has only one operation $add(c)$ where c is a character added (at the end) to the processed string. Also author proposed version of structure where operation $pop()$ (deletion of last element from structure) is added and joint version for several strings.

4 EERTREE ANALYSIS

Versions from original paper do not support operation such as

prepend adding character at the beginning of processing string
set modifying any character in processing string
merge combining two structures in one, in processing strings it will be simple concatenation.

4.1 $prepend(c)$

In original paper $add(c)$ is a $append(c)$. Since it adds to end of sting, in structure it operates on suffix links. To support $prepend(c)$ operation we can add prefix links. Which will not affect on structure asymptotically.

4.2 $set(i, c)$

In the worst case scenario modifying a single character will produce or vice-versa remove $|S|/2$ palindromes. Thus $\Omega(min(i, |S|-i))$ is minimal. Naively we can $pop()$ till modified character and rebuild a tree so $O(min(i, |S| - i) \cdot \log \sigma)$ (where σ is a size of alphabet) is possible. But is it possible to lower this value down to, say, $O(\sigma)$ since edges represent insertion character?

4.3 $merge(eertree_1, eertree_2)$

For merge operation interesting property was found. It might be represented as extension of lemma from original paper.

Lemma 4.1 ([4, Lemma 1]). *Let S be a string and c be a symbol. The string Sc contains at most one subpalindrome which is not a substring of S. This new palindrome is the longest suffix-palindrome of Sc.*

Proposition 4.2. *Let S_1 be a first string, S_2 be a second string and $|S_1|, |S_2|$ be the length of first string and second one respectively. The concatenation of both strings S_1S_2 contains at most $min(|S_1|, |S_2|)$ subpalindromes which are not a substring of S_1 nor S_2.*

Thus theoretical minimum for merging two structures is $\Omega(min(|S_1|, |S_2|))$. Using version with $prepend(c)$ and $append(c)$ it may be possible to merge in $\Theta(min(|S_1|, |S_2|) \cdot \log \sigma)$. Interesting question emerges: is it possible to find merge algorithm with complexity $O(min(|S_1|, |S_2|) + \log \sigma)$?

5 COMPARISON OF APPROACHES

	Eertree	Affix tree	Manacher's algorithm
$build(S)$	$O(\|S\| \cdot \log \sigma)$	$O(\|S\| \cdot \log \|S\|)$	$O(\|S\|)$
$append(c)$	$O(\|S\|)$ $O(\log \|S\|)$ $\Theta(\log \sigma)$ are possible	$O(\log \|S\|)$	$O(1)$
$prepend(c)$	$O(\|S\|)$ $O(\log \|S\|)$ $\Theta(\log \sigma)$ are possible	$O(\log \|S\|)$	-
$pop()$	$O(1)$	-	-
$set(i, c)$	$O(min(i, \|S\| - i) \cdot \log \sigma)$	$O(1)$	-
$merge(S_1, S_2)$	$\Theta(min(\|S_1\|, \|S_2\|) \cdot \log \sigma)$	-	-

Manacher's algorithm does not build something, it just calculates.

REFERENCES

[1] e olymp. [n. d.]. Palindrome generator problem. ([n. d.]). https://www.e-olymp.com/en/problems/2468
[2] informatics. [n. d.]. Palindromes problem. ([n. d.]). https://informatics.mccme.ru//mod/statements/view.php?chapterid=1750
[3] Glenn Manacher. 1975. A New Linear-Time"On-Line"Algorithm for Finding the Smallest Initial Palindrome of a String. *Journal of the ACM (JACM)* 22, 3 (1975), 346–351.
[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.
[5] SPOJ. [n. d.]. "Let us play with strings" problem. ([n. d.]). https://www.spoj.com/problems/IITKWPCE/
[6] SPOJ. [n. d.]. Number of palindromes problem. ([n. d.]). https://www.spoj.com/problems/NUMOFPAL/
[7] Jens Stoye. 2000. Affix trees. (2000).
[8] Dirk Strothmann. 2007. The affix array data structure and its applications to RNA secondary structure analysis. *Theoretical Computer Science* 389, 1-2 (2007), 278–294.