МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Построение и анализ алгоритмов»

Тема: Алгоритм Кнута-Морриса-Пратта

Студент гр. 7383 _____ Бергалиев М.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург

2019

Цель работы

Реализовать и исследовать алгоритм Кнута-Морриса-Пратта поиска всех подстрок в строке.

Постановка задачи

Реализовать алгоритм Кнута-Морриса-Пратта и с его помощью для заданных шаблона P ($|P|\leq 15000$) и текста T ($|T|\leq 5000000$) найти все вхождения P в T. По памяти сложность не должна превышать O(|P|).

Ход работы

Была написана программа на языке программирования C++. Код представлен в приложении А.

Сначала для шаблона вычисляется значение префикс функции и записывается в вектор. Для i-го символа строки s префикс функция pi вычисляется по правилам: j=pi[i-1], пока s[j] не равно s[i] и j > 0, то j=pi[j-1], если j не равно 0, то pi[i]=j+1, иначе pi[i]=0. Далее для каждого символа текста вычисляется значение префикс функции, используя полученные значения для шаблона и значение, полученное для предыдущего символа. Если для i-го символа текста значение префикс функции равно |P|, то на i-ом символе заканчивается подстрока в тексте, соответствующая шаблону, и в ответ записывается индекс i-|P|+1.

Тестирование

Тестирование проводилось в Ubuntu 16.04 LTS. По результатам тестирования были выявлены ошибки в коде. Тестовые случаи представлены в приложении Б.

Исследование алгоритма

Исследование проводилось по количеству сравнений символов в зависимости от суммы длин строк. Данные представлены в табл. 1.

Таблица 1 — Результаты исследования зависимости числа сравнений от суммы длин строк

|P|+|T|	Число сравнений
1005	1079
10000	10867
101000	109191
500000	539545
1000005	1077852
1000100	1078385

По полученным данным был построен график, оценивающий сложность алгоритма, показанный на рис. 1. По результатам сложность алгоритма линейная.

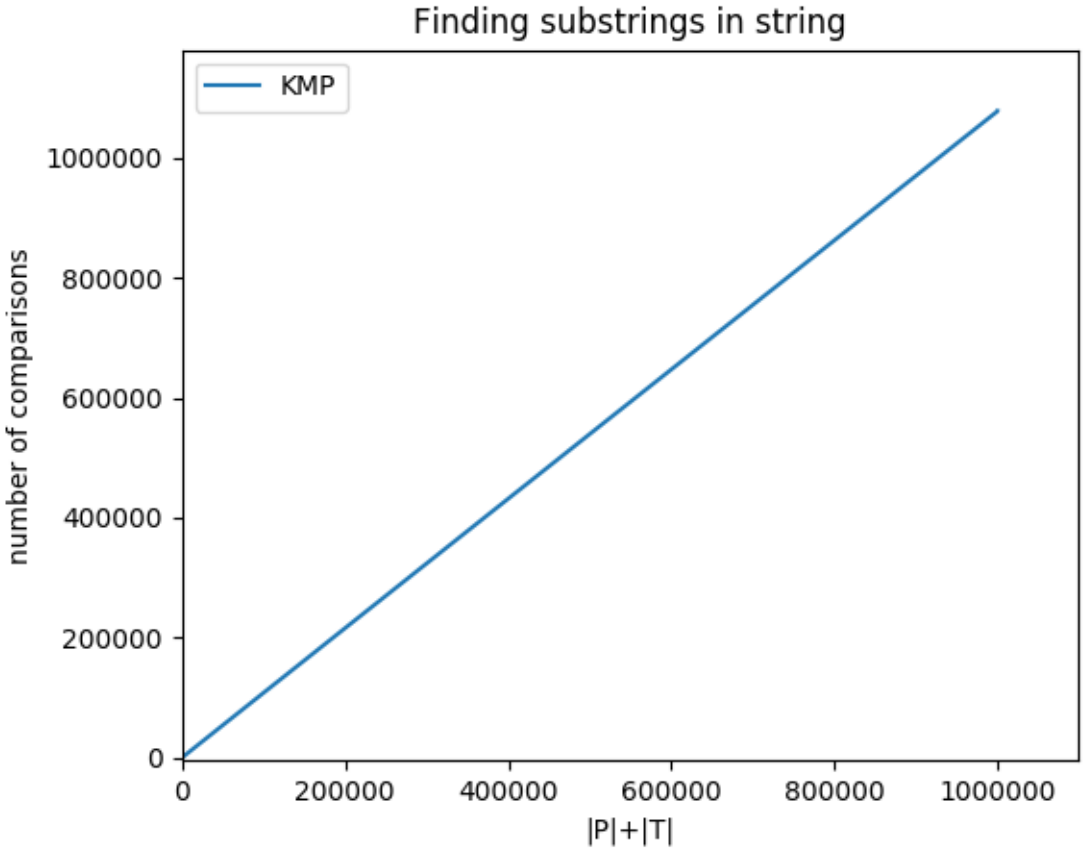

Рисунок 1 — Сложность алгоритма

Вывод

Был реализован и исследован алгоритм Кнута-Морриса-Пратта. Сложность по количеству сравнений равна O(|P|+|T|), по памяти O(|P|).

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <iostream>
#include <cstdio>
#include <vector>

std::vector<int> pref_func(std::string const & str){
	int n = str.length();
	std::vector<int> pi(n);
	int j;
	for(int i=1; i<n; ++i){
		j = pi[i-1];
		while(str[j] != str[i] && j > 0){
			j = pi[j-1];
		}
		if(str[j] == str[i]){
			j += 1;
		}
		pi[i] = j;
	}
	return pi;
}

std::string cmp(std::string const & p){
	int n = p.length();
	std::string ans;
	std::vector<int> pi = pref_func(p);
	int j = 0;
	int i = 0;
	char t;
	while(std::cin.get(t)){
		while(j == n || (p[j] != t && j > 0)){
			j = pi[j-1];
		}
		if(p[j] == t){
			j += 1;
		}
		if(j == n){
```

```
                ans += std::to_string(i-n+1);
                ans.push_back(',');
            }
            ++i;
        }
        if(ans == "")
            return "-1";
        ans.pop_back();
        return ans;
}

int find_cycle(std::string a, std::string b){
        int b_len = b.length();
        if(b_len != a.length())
            return -1;
        std::vector<int> pi = pref_func(b);
        int j = 0;
        for(int i=0; i<b_len; ++i){
            while(j == b_len || (b[j] != a[i] && j > 0)){
                j = pi[j-1];
            }
            if(b[j] == a[i]){
                j += 1;
            }
        }
        pi = pref_func(a);
        int k = 0;
        for(int i=0; i<b_len; ++i){
            while(k == b_len || (a[k] != b[i] && k > 0)){
                k = pi[k-1];
            }
            if(a[k] == b[i]){
                k += 1;
            }
        }
        if(j + k < b_len)
            return -1;
        return b_len - j;
}
```

```
int main(){
	std::string p, t;
	std::getline(std::cin, p);
	auto ans = cmp(p);
	std::cout << ans << std::endl;
	return 0;
}
```

ПРИЛОЖЕНИЕ Б

ТЕСТОВЫЕ СЛУЧАИ

Таблица 1 — Тестовые случаи

Входные данные	Результат
ab abab	0,2
121 121212123	0,2,4
ab aaaaaaaaaab	10