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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Построение и анализ алгоритмов»

Тема: «Алгоритм Кнута-Морриса-Пратта»

Студентка гр.7304 _____ Каляева А.В.

Преподаватель _____ Филатов А.Ю

г. Санкт-Петербург

г. 2019

Цель работы:

Изучить алгоритм Кнута-Морриса-Пратта поиска подстроки в строке, а так же реализовать данный алгоритм на языке программирования C++.

Задание:

1) Реализовать алгоритм КМП и с его помощью для заданных шаблона P ($|P|\leq 15000$) и текста T ($|T|\leq 5000000$) найдите все вхождения P в T.

2) Заданы две строки A ($|A|\leq 5000000$) и B ($|B|\leq 5000000$). Определить, является ли A циклическим сдвигом B (это значит, что A и B имеют одинаковую длину и A состоит из суффикса B, склеенного с префиксом B). Например, defabc является циклическим сдвигом abcdef.

Ход работы:

1) Был реализован алгоритм Кнута-Морриса-Пратта для поиска подстроки в строке на языке программирования C++.

a. Для корректной работы алгоритма КМП на первом шаге была реализована префикс-функция для подстроки, которая определяет наибольшую длину префикса, который одновременно является суффиксом для данной подстроки. Функция возвращает вектор типа int, который хранит максимальные длины суффиксов, которые одновременно являются суффиксами подстроки.

b. Далее была написана функция, которая непосредственно реализует поиск подстроки в строке. В качестве аргументов функция принимает две строки: строку – образ и строку, в которой необходимо производить поиск. Функция работает следующим образом: пока не был достигнут конец строки, выполняется сравнение символов строки и подстроки. Если символы равны, индексы увеличиваются, и функция переходит к сравнению следующих символов. Если символы оказались не равны и индекс образа не указывает на его начало, то новый индекс образа вычисляется с использованием префикс-функции. Иначе индекс строки увеличивается на 1.

c. После завершения сравнения строки и образа функция возвращает массив индексов начал вхождения подстроки в строку.

2) Был реализован алгоритм, который определяет, является ли одна строка циклическим сдвигом второй строки.

a. На первом шаге была реализована префик-функция, аналогичная той, что описана в пункте 1.a.

b. Была реализована функция, которая проверяет на цикличность

две строки. Если длины исходных строк не равны или не удалость найти вхождение подстроки в строку, то функция возвращает значение -1. Если строки, переданные данной функции, равны, то функция возвращает значение равное 0. Иначе выполняется поиск полстроки в строке, с условием, что, если был достигнут конец строки, в которой выполняется поиск, то индекс обнуляется, и поиск продолжается дальше. В результате функция возвращает индекс начала одной строки в другой строке.

Пример работы программы:

1) Входные данные:

```
ab
abab
```

Результат работы программы:

```
0,2
```

2) Входные данные:

```
defabc
abcdef
```

Результат работы программы:

```
3
```

Заключение:

В ходе выполнения лабораторной работы по теме «Алгоритм Кнута-Морриса-Пратта». Был изучен и реализован на языке программирования C++ алгоритм Кнута-Морриса-Пратта для поиска подстроки в строке. Для корректной работы алгоритма было изучено понятие префик-функции, которая определяет длину наибольшего префикса подстроки, который совпадает с ее суффиксом. Временная сложность вычисления префикс функции оценивается, как O(n). Временная сложность работы всего алгоритма оценивается, как O(n+m).

Приложение А
Исходный код программы lr4_1.cpp

```
#include <iostream>
#include <string>
#include <vector>

using namespace std;

vector <int> prefix(string arr) {
	vector <int> p(arr.size());
	size_t j = 0;
	size_t i = 1;
	p[0] = 0;
	while (i < arr.length()) {
		if (arr[i] == arr[j]) {
			p[i] = j + 1;
			i++;
			j++;
		}
		else if (j == 0) {
			p[i] = 0;
			i++;
		}
		else {
			j = p[j - 1];
		}
	}
	return p;
}

vector <int> KMP(string form, string line) {
	vector <int> res;
	vector <int> p = prefix(form);
	size_t i = 0;
	size_t j = 0;
	while ( i < line.length()) {
		if (line[i] == form[j]) {
			i++;
			j++;
			if (j == form.length()) {
				res.push_back(i - form.length());
			}
		}
		else if (j != 0) {
			j = p[j - 1];
		}
		else {
			i++;
		}
	}
	return res;
}

int main() {
	string form;
	string line;
	getline(cin, form);
	getline(cin, line);
	vector <int> res = KMP(form, line);
    if(res.size()==0){
        cout<<"-1";
    }
    else{
```

```
            for (size_t i = 0; i < res.size(); i++) {
                   cout << res[i];
                   if (i != res.size() - 1) {
                      cout << ",";
                   }
            }
    }
       return 0;
}
```

Приложение В
Исходный код программы lr4_2.cpp

```
#include <iostream>
#include <string>
#include <vector>
#include <cmath>

using namespace std;

vector <int> prefix(string arr) {
	vector <int> p(arr.size());
	size_t j = 0;
	size_t i = 1;
	p[0] = 0;
	while (i < arr.length()) {
		if (arr[i] == arr[j]) {
			p[i] = j + 1;
			i++;
			j++;
		}
		else if (j == 0) {
			p[i] = 0;
			i++;
		}
		else {
			j = p[j - 1];
		}
	}
	return p;
}

int CYCLE_KMP(string s1, string s2) {
	vector <int> p = prefix(s2);
	size_t i = 0;
	size_t j = 0;
	if (s1.length() != s2.length()) {
		return -1;
	}
	else if (s1 == s2) {
		return 0;
	}
	else {
		while (true) {
			if (s1[i] == s2[j]) {
				i++;
				if (i==s1.length()) {
					i -= s1.length();
				}
				j++;
				if (j == s2.length()) {
					return s2.length() - i;
				}
			}
			else if (j != 0) {
				j = p[j - 1];
			}
			else {
				i++;
				if (i == s1.length()) {
					return -1;
				}
```

```
                    }
                }
            }
        }

int main() {
        string a;
        string b;
        getline(cin, a);
        getline(cin, b);
        cout<<CYCLE_KMP(b, a);
        return 0;
}
```