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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Построение и анализ алгоритмов»

Тема: Алгоритм Кнута-Морриса-Пратта

Студент гр. 9383 Поплавский И.

Преподаватель Фирсов М.А.

Санкт-Петербург

2021

Цель работы

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

Постановка задачи.

1. Реализуйте 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.

   Входные данные

   Первая строка – P.

   Вторая строка – T.

   Выходные данные

   Индексы начал вхождений P в T, разделенных запятой, если P не входит в T, то вывести −1.

   Пример входных данных

   ab

   abab

   Соответствующие выходные данные

   0, 2

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.

   Входные данные

   Первая строка – A.

   Вторая строка – B.

   Выходные данные

   Если A является циклическим сдвигом B, индекс начала строки B в A, иначе вывести −1. Если возможно несколько сдвигов вывести первый индекс.

Пример входных данных

defabc

abcdef

Соответствующие выходные данные

3

Реализация задачи

Описание алгоритма и функций

1. Был реализован алгоритм Кнута-Морриса-Пратта для поиска подстроки в строке. Программа считывает строку-образец, вхождение которой нужно найти в строке-тексте.

На первом шаге была реализована префикс-функция для подстроки `vector <int> prefix(string arr)`, которая принимает строку-образец `arr` и 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.

Далее была написана функция `vector <int> KMP(string form, string line)`, которая непосредственно реализует поиск подстроки в строке. В качестве аргументов функция принимает две строки: строку – образ `form` и строку `line`, 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. После завершения сравнения строки и образа функция возвращает массив индексов начал вхождения подстроки в строку.

Функция вычисления префикс-функции проходится по строке-образцу 1 раз, поэтому требует O(m) времени, где m – длина строки-образца. Процесс поиска вхождений строки-образца в строке-текста выполняется, пока не закончится строка-текст, т.е. требует O(n) времени, где n - длина строки-текста. Итого общая оценка **сложности по времени** алгоритма Кнутта-Морриса-Пратта составляет O(m + n).

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

На первом шаге была реализована префикс-функция для подстроки `vector <int> prefix(string arr)`, которая принимает строку `arr` и 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.

Далее была реализована функция `int CYCLE_KMP(string s1, string s2)`, которая проверяет на цикличность две строки `s1` и `s2`. 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 подстроки 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.

Сложность алгоритма **по времени** составляет O(m), где m - длина строк.

Тестирование

№	Входные данные	Выходные данные
1	abababcab abababababcab	4
	ab abab	0, 2
	abcbcacb acbacbbcac	-1
	sssssss bvsssssssssb	2,3,4
	a aaaaaa	0,1,2,3,4,5
2	defabc abcdef	3
	ababab bababa	1
	abcdef abcd	-1
	abcd abcd	0
	abcd bcde	-1

Также были написаны модули для тестирования, которые программа успешно прошла.

Вывод

В ходе выполнения лабораторной работы был изучен и реализован на языке программирования C++ алгоритм Кнута-Морриса-Пратта для поиска подстроки в строке и поиска циклического сдвига строк. Для корректной работы алгоритма было изучено понятие префикс-функции, которая определяет длину наибольшего префикса подстроки, совпадающего с ее суффиксом.

ПРИЛОЖЕНИЕ А

ПОИСК ПОДСТРОКИ В СТРОКЕ

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

std::vector <int> prefix(std::string arr) {
	std::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;
}

std::vector <int> KMP(std::string form, std::string line) {
	std::vector <int> res;
	std::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() {
	setlocale(LC_ALL, "Russian");
	std::string form;
	std::string line;
	getline(std::cin, form);
	getline(std::cin, line);
	std::vector <int> res = KMP(form, line);
	if (res.size() == 0) {
		std::cout << "-1";
```

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

ПРИЛОЖЕНИЕ В

ЦИКЛИЧЕСКИЙ СДВИГ СТРОК

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


std::vector <int> prefix(std::string& arr) {//префикс функция
	std::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]) {//если символы совпадают
			p[i] = j + 1;//увеличиваем префикс
			i++;
			j++;
		}
		else if (j == 0) {//еще не было совпадений
			p[i] = 0;
			i++;
		}
		else {//символы не совпадают
			j = p[j - 1];
		}
	}
	return p;
}

int CYCLE_KMP(std::string& s1, std::string& s2) {
	size_t i = 0;
	size_t j = 0;
	if (s1.length() != s2.length()) {//длины строк не равны
		return -1;
	}
	else if (s1 == s2) {//строки равны
		return 0;
	}
	else {// кмп
		s2 += " " + s1 + s1;
		std::vector<int> p = prefix(s2);
		int res = -1;
		for (int k = s1.length() + 1; k < s2.length(); k++)
		{
			if (p[k] == s1.length())
			{
				res = k - 2 * s1.length();
				break;
			}
		}
		return res;
	}
}


int main() {
	setlocale(LC_ALL, "Russian");
	std::string a;
	std::string b;
	getline(std::cin, a);
	getline(std::cin, b);
```

```
        std::cout << CYCLE_KMP(a, b);
        return 0;
}
```