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: Кнут-Моррис-Пратт

Студент гр. 9383 _____ Орлов Д.С.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2021

Цель работы.

Изучить и реализовать алгоритм Кнута-Морриса-Пратта для поиска всех подстрок по шаблону. Реализовать алгоритм проверки на циклический сдвиг.

Постановка задачи.

1 пункт. 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.

Вход:

Первая строка - P

Вторая строка - T

Выход:

индексы начал вхождений P в T, разделенных запятой, если P не входит в T, то вывести -1

2 пункт. 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.

Вход:

Первая строка - A

Вторая строка - B

Выход:

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

Описание алгоритма.

Математически определение префикс-функции можно записать следующим образом: $\pi[i] = max_{n=0\ldots i}\{n: s[0 \ldots n-1] = s[i-n+1 \ldots i]\}$, где s[0 … n-1] - данная строка.

Алгоритм Кнута-Морриса-Пратта, который находит позиции всех вхождений строки P в текст T, работает следующим образом.

Построим строку S=P@T, где @ - любой символ, не входящий в алфавит P и T. Посчитаем на ней значение префикс-функции p. Если в какой-то позиции i выполняется условие p[i]=|P|, то в этой позиции начинается очередное вхождение образца в цепочку.

Алгоритм поиска циклического сдвига отличается только тем, что первая строка удваивается, так как при сложении строк первая будет содержать в себе вторую строку, если она является циклическим сдвигом.

Алгоритм Кнута-Морриса-Пратта имеет сложность O(P+T) по времени и памяти.

Описание структур и функций.

- std::vector<int> prefixFunction(std::string &str) - формирует вектор значений префикс функции символов входной строки и возвращает его;
- void KMP(std::string &P, std::string &T) - функция реализует алгоритм Кнута-Морриса-Пратта;
- void KMP_Circle(std::string &P, std::string &T) - функция проверяет является ли строка P, циклическим сдвигом строки T;

Код разработанных программ см. в приложении А.

Тестирование.

Таблица 1 - результаты тестирования lab4_1.cpp

Тест	Входные данные	результат работы алгоритма
№1	ab abab	0,2
№2	qwerty uuiop	-1
№3	qwqw	0,2,4

	qwqwqwqw	

Таблица 2 - результаты тестирования lab4_circle.cpp

Тест	Входные данные	результат работы алгоритма
№1	qwerty wertyq	1
№2	qwer rewq	-1
№3	qwerty tyqwer	4

Вывод.

В ходе выполнения работы были изучены алгоритм Кнута - Морриса - Пратта для поиска всех подстрок и префикс функция. Также алгоритм Кнута - Морриса - Пратта был оптимизирован для решения задачи поиска циклического сдвига. Данные алгоритмы были реализованы на языке программирования C++.

ПРИЛОЖЕНИЕ А
Исходный код программы

Название файла: lab4_1.cpp

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


std::vector<int> prefixFunction(std::string &str)
{
    int n = str.length();
    std:: vector<int> prefixArray(n, 0);

    for(int i = 1; i < str.length(); i++)
    {
        int j = prefixArray[i - 1];

        while((j > 0) && (str[i] != str[j]))
        {
            j = prefixArray[j - 1];
        }

        if(str[i] == str[j])
        {
            j++;
        }

        prefixArray[i] = j;
    }

    return prefixArray;
}

void KMP(std::string &P, std::string &T)
{
    std::string result = "";
```

```
    std::vector <int> prefixArray;

    int lengthP = P.length();

    P += "@" + T;

    prefixArray = prefixFunction(P);

    for(int i = 0; i < prefixArray.size(); i++)
    {
        if(prefixArray[i] == lengthP)
        {
            result += std::to_string(i - 2 * lengthP) + ",";
        }
    }

    if(result.length() == 0)
    {
        std::cout<<"-1";
    }
    else
    {
        result.pop_back();
        std::cout<<result;
    }
}

int main()
{
    std::string T, P, s;
    std::cin>>P>>T;

    KMP(P, T);
```

```
    return 0;
}
```

Название файла: lab4_circle.cpp

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


std::vector<int> prefixFunction(std::string &str)
{
    int n = str.length();
    std:: vector<int> prefixArray(n, 0);

    for(int i = 1; i < str.length(); i++)
    {
        int j = prefixArray[i - 1];

        while((j > 0) && (str[i] != str[j]))
        {
            j = prefixArray[j - 1];
        }

        if(str[i] == str[j])
        {
            j++;
        }

        prefixArray[i] = j;
    }

    return prefixArray;
}

void KMP_Circle(std::string &P, std::string &T)
{
    std::vector <int> prefixArray;
```

```
    int lengthT = T.length();

    T += "@" + P;

    prefixArray = prefixFunction(T);

    for(int i = 0; i < prefixArray.size(); i++)
    {
        if(prefixArray[i] == lengthT)
        {
            std::cout<<i - 2 * lengthT;
            return;
        }
    }

    std::cout<<"-1";

}

int main()
{
    std::string T, P;
    std::cin>>P>>T;

    if(P.length() != T.length())
    {
        std::cout<<"-1";
        return 0;
    }

    P += P;

    KMP_Circle(P, T);
```

```
    return 0;
}
```