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

Студентка гр. 9382 _____ Сорочина М.В.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2021

Цель работы.

Изучить принцип действия алгоритма Кнута-Морриса-Пратта. А также на основе данного алгоритма реализовать программы, выполняющие поиск шаблона в тексте и проверку на сдвиг.

Задание.

Задание 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.

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

Sample Input:
ab
abab
Sample Output:
0,2

Задание 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.
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.

Sample Input:

defabc

abcdef

Sample Output:

3

Описание функций и структур данных.

Задание 1.

1) `void input(std::string &p, std::string &t);`

Функция, считывающая и записывающая ввод. p и t - строки, куда записывается ввод.

2) `std::vector<int> initialPi(std::string p);`

Функция, считающая префикс-функцию для переданной строки. p - строка, для которой нужно посчитать префикс функцию. Возвращает вектор целых чисел, который и является префикс-функцией.

3) `std::vector<int> answer(std::string p, std::string t);`

Функция, находящая все вхождения шаблона p в текст t. p -шаблон, t - текст, в котором ищется вхождение шаблона. Возвращает вектор целых чисел - начал вхождений шаблона в текст. Если вектор пустой, значит шаблона нет в тексте.

Задание 2.

1) `void input(std::string &A, std::string &B);`

Функция, считывающая и записывающая ввод. A и B - строки, куда записывается ввод.

2) `std::vector<int> initialPi(std::string p);`

Функция, считающая префикс-функцию для переданной строки. p - строка, для которой нужно посчитать префикс функцию. Возвращает вектор целых чисел, который и является префикс-функцией.

3) `int answer(std::string p, std::string t);`

Функция, определяющая является ли t циклическим сдвигом p. p и t - строки для проверки. Возвращает целое число: -1, если не является циклическим сдвигом, иначе индекс начала строки p в t.

Описание алгоритма.

Задание 1.

Строится префикс-функция для шаблона. (Префикс-функция - массив максимальных длин совпадающих суффиксов и префиксов). После построения начинается проход по тексту, во время которого сравниваются символы в шаблоне и тексте. Если символы равны, то оба индекса увеличиваются на 1 для сравнения следующих символов. Если при этом индекс, проходящий по шаблону стал равен длине шаблона, то нашлось вхождение, и оно добавляется в вектор ответа.

Задание 2.

Для начала проверяется длина строк, если она разная, то сразу выводится -1, что означает строки не являются циклическим сдвигом друг друга. Далее проверяется равны ли строки, если равны, то значение сдвига 0. Если строки одной длины, но не равны, используется алгоритм КМП для проверки циклического сдвига. В становится шаблоном и дважды проходит по А, тк если В является циклическим сдвигом А, то она будет содержаться в удвоенной строке. Как только находится вхождение, возвращается индекс начала вхождения, если вхождение не нашлось, возвращается -1.

Оценка сложности.

Задание 1.

По времени: O(m), m - длина текста, тк алгоритм проходит по тексту

По памяти: O(n+m), где n - длина шаблона, m - длина текста, тк при построении префикс-функции создается вектор длины n, а вхождений шаблона не может быть больше, чем длина текста.

Задание 2.

По времени: O(n), где n - длина каждой из строк, тк алгоритм дважды проходит по строке.

По памяти: O(n), где n - длина каждой из строк, тк создается префикс-функция размера n.

Тестирование.

№ теста	Ввод	Задание 1.	Задание 2.
1	ab abab	0,2	-1
2	defabc abcdef	-1	3
3	abcabd abcabeabcabcabd	9	-1
4	qqq qqqqqq	0,1,2,3	-1
5	www www	0	0
6	qwertyqwerty ertyqwertyqw	-1	2

Выводы.

В ходе выполнения работы была написана программа, реализующая поиск шаблона в тексте, а также программа, определяющая являются ли строки циклическим сдвигом друг друга.

ПРИЛОЖЕНИЕ А.

ИСХОДНЫЙ КОД ПРОГРАММЫ lab4_1.cpp.

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

#define COMMENTS

void input(std::string &p, std::string &t);
//функция, считывающая ввод
std::vector<int> initialPi(std::string &p);
//функция вычисления префикс-функции
std::vector<int> answer(std::string &p, std::string &t);
//функция поиска ответа

int main()
{
    std::string p, t; //p - шаблон, t - текст
#ifdef COMMENTS
    std::cout << "Введите шаблон и текст\n";
#endif
    input(p, t);

    std::vector<int> ans = answer(p, t);
#ifdef COMMENTS
    std::cout << "Ответ:\n";
#endif
    if (ans.size() == 0)
    {
#ifdef COMMENTS
        std::cout << "Текст не содержит вхождений шаблона\n";
#endif
        std::cout << "-1\n";
        return 0;
    }
    for (int i = 0; i < ans.size() - 1; i++)
    {
        std::cout << ans[i] << ",";
    }
    std::cout << ans[ans.size() - 1] << '\n';
    return 0;
}

void input(std::string &p, std::string &t)
//функция, считывающая ввод
{
    getline(std::cin, p);
    getline(std::cin, t);
}

std::vector<int> initialPi(std::string &p)
```

```
//функция вычисления префикс-функции
//возвращает вектор - префикс-функцию
{
    std::vector<int> pi;
    int len = p.length();
    pi.resize(len);             //меняем размер массива на длину шаблона
    pi[0] = 0;                  //первый элемент всегда 0
     int suffix = 1, prefix = 0; //suffix - индекс для суффикса, prefix -
для префикса
    while (suffix < len)
    {
        if (p[suffix] == p[prefix])
        {
            pi[suffix] = prefix + 1;
            suffix++;
            prefix++;
        }
        else
        {
            if (prefix == 0)
            {
                pi[suffix] = 0;
                suffix++;
            }
            else
            {
                prefix = pi[prefix - 1];
            }
        }
    }
    return pi;
}

std::vector<int> answer(std::string &p, std::string &t)
//функция поиска ответа
//возвращает вектор, элементы которого - индексы начал вхождения. Если он
пуст, то вхождений нет
{
    std::vector<int> ans; //вектор для записи ответа
    std::vector<int> pi;
    int p_len = p.length();
    pi.resize(p_len);
    pi = initialPi(p); //вычисляем префикс-функцию для шаблона
#ifdef COMMENTS
    std::cout << "\tПрефикс-функция шаблона:\n\t";
    for (auto i : p)
    {
        std::cout << i << " ";
    }
    std::cout << "\n\t";
    for (auto i : pi)
    {
        std::cout << i << " ";
```

```
    }
    std::cout << '\n';
#endif

     int t_ind = 0, p_ind = 0; //t_ind - индекс для прохода по тексту,
p_ind - по шаблону
    int t_len = t.length();
    while (t_ind < t_len)
    {
#ifdef COMMENTS
        std::cout << "\tСравним t[" << t_ind << "] '" << t[t_ind] << "' и
p[" << p_ind << "] '" << p[p_ind] << "'\n";
#endif
        if (t[t_ind] == p[p_ind])
        {
#ifdef COMMENTS
            std::cout << "\t  Равны\n";
#endif
            t_ind++;
            p_ind++;
            if (p_ind == p_len)
            {
#ifdef COMMENTS
                        std::cout << "\t  Дошли до конца шаблона. Шаблон
содержится в тексте с " << t_ind - p_len << "\n";
#endif
                ans.push_back(t_ind - p_len);
                p_ind = pi[p_ind - 1];
            }
        }
        else
        {
#ifdef COMMENTS
            std::cout << "\t  НЕ равны\n";
#endif
            if (p_ind == 0)
            {
                t_ind++;
            }
            else
            {
                p_ind = pi[p_ind - 1];
            }
        }
    }
#ifdef COMMENTS
    std::cout << "Дошли до конца текста. Конец поиска.\n";
#endif
    return ans;
}
```

ПРИЛОЖЕНИЕ Б.

ИСХОДНЫЙ КОД ПРОГРАММЫ lab4_2.cpp.

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

#define COMMENTS

void input(std::string &A, std::string &B);
//функция, считывающая ввод
std::vector<int> initialPi(std::string &p);
//функция вычисления префикс-функции
int answer(std::string &p, std::string &t);
//функция поиска ответа

int main()
{
    std::string A, B;
#ifdef COMMENTS
    std::cout << "Введите 2 строки\n";
#endif
    input(A, B);
    if (A.length() != B.length())
    {
#ifdef COMMENTS
        std::cout << "Строки разной длины\n";
#endif
        std::cout << "-1\n";
        return 0;
    }
    if (A == B)
    {
#ifdef COMMENTS
        std::cout << "Строки совпадают\n";
#endif
        std::cout << "0\n";
        return 0;
    }
    int ans = answer(B, A);
#ifdef COMMENTS
    std::cout << "Ответ:\n";
    if (ans == -1)
    {
        std::cout << "Не является сдвигом\n";
    }
#endif
    std::cout << ans << "\n";
    return 0;
}

void input(std::string &A, std::string &B)
```

```
//функция, считывающая ввод
{
    getline(std::cin, A);
    getline(std::cin, B);
}

std::vector<int> initialPi(std::string &p)
//функция вычисления префикс-функции
//возвращает вектор - префикс-функцию
{
    std::vector<int> pi;
    int len = p.length();
    pi.resize(len);             //меняем размер массива на длину шаблона
    pi[0] = 0;                  //первый элемент всегда 0
     int suffix = 1, prefix = 0; //suffix - индекс для суффикса, prefix -
для префикса
    while (suffix < len)
    {
        if (p[suffix] == p[prefix])
        {
            pi[suffix] = prefix + 1;
            suffix++;
            prefix++;
        }
        else
        {
            if (prefix == 0)
            {
                pi[suffix] = 0;
                suffix++;
            }
            else
            {
                prefix = pi[prefix - 1];
            }
        }
    }
    return pi;
}

int answer(std::string &p, std::string &t)
{
    int ans = -1;
    std::vector<int> pi;
    int p_len = p.length();
    pi.resize(p_len);
    pi = initialPi(p); //считаем префикс-функцию для p aka A
#ifdef COMMENTS
    std::cout << "\tПрефикс-функция шаблона:\n\t";
    for (auto i : p)
    {
        std::cout << i << " ";
    }
```

```
    std::cout << "\n\t";
    for (auto i : pi)
    {
        std::cout << i << " ";
    }
    std::cout << '\n';
#endif

    int t_ind = 0, p_ind = 0;
     bool flag = false; //флаг для отметки первый раз идем по строке или
уже второй
    int t_len = t.length();
    while (t_ind < t_len)
    {
#ifdef COMMENTS
        std::cout << "\tСравним A[" << t_ind << "] '" << t[t_ind] << "' и
B[" << p_ind << "] '" << p[p_ind] << "'\n";
#endif
        if (t[t_ind] == p[p_ind])
        {
#ifdef COMMENTS
            std::cout << "\t\tРавны\n";
#endif
            t_ind++;
            p_ind++;
            if (p_ind == p_len)
            {
                if (flag)
                {
                    t_ind += t_len;
                }
#ifdef COMMENTS
                std::cout << "\t\tНашли циклический сдвиг. Начинается с "
<< t_ind - p_len << "\n";
#endif
                ans = t_ind - p_len;
                return ans;
            }
        }
        else
        {
#ifdef COMMENTS
            std::cout << "\t\tНЕ равны\n";
#endif
            if (p_ind == 0)
            {
                t_ind++;
            }
            else
            {
                p_ind = pi[p_ind - 1];
            }
        }
```

```
        if (t_ind == t_len && flag == false) //если дошли до конца строки
первый раз,
        {
            t_ind = 0;   //то переход на начало
            flag = true; //меняем флаг
        }
    }
#ifdef COMMENTS
    std::cout << "Дошли до конца текста. Конец поиска.\n";
#endif
    return ans;
}
```