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. 9382 _____ Сорокумов С. В.

Преподаватель _____ Фирсов М. А.

Санкт-Петербург
2021

Цель работы.

Изучить алгоритм Кнута-Морриса-Пратта поиска подстроки в строке. Реализовать программу на C++, которая для заданного текста находит все вхождения заданной строки.

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

Функции и структуры данных.

```
std::vector<int> prefix(const std::string& data)
```

Префикс функция, которая принимает на вход константную ссылку на строку data.

Функция возвращает вектор целых чисел.

```
std::vector<int> kmp(std::string& A, std::string& B)
```

Функция реализует алгоритм Кнута-Морриса-Пратта. Принимает на вход 2 ссылки на строки A и B.

Функция возвращает вектор целых чисел.

Описание алгоритма.

Сначала была написана префикс функция, которая возвращает вектор, элементы которого обозначают длину максимального префикса строки *s[0..i]*, где *i* - индекс элемента массива, совпадающего с суффиксом данной строки. Этот вектор используется в функции *kmp()*, которая находит индексы вхождений подстроки в строку. Для этого, нужно всего лишь найти элементы вектора такие,

что они равны длине подстроки. В таком случае, получим начало вхождения подстроки P в текст T.

Алгоритм Кнута-Морриса-Пратта позволяет находить префикс функцию для заданной строки за линейное время и осуществлять поиск подстроки в строке. Изначально алгоритм вычисляет префикс-функцию для строки вида $P\#T$, где P — образец, который ищется в тексте T, значок # гарантированно не встречается ни в P, ни в T. Если префикс-функция содержит значение, равные длине P, значит P входит в T. Таким образом решается первая задача.

Вторая задача решается тем, что с помощью алгоритма Кнута-Морриса-Пратта происходит поиск подстроки B в строке $A + A$. Для этого необходимо вычислить значение префикс-функции для строки вида $B\#AA$.

Оценка сложности алгоритма.

Пусть длина строки, для которой вычисляется префикс-функция равна m, а текста T — n. Тогда сложность алгоритма равна $O(m + n)$, т. к. алгоритм сначала проходится по строке, переданной в префикс-функцию, а после проходит по строке T.

Тестирование.

Все тесты продемонстрированы в таблице 1.

Таблица 1 – тестирование программ

Номер теста	Входные данные	Ответ	Тестирование алгоритма (1/2)
1	ab abab	0,2	1
2	we wertwengkwe	0,4,9	1
3	lpkps knfkndlngvflspe	-1	1
4	an annaalenann	0,8	1
5	aslf dfdg	-1	2

6	defabc abcdef	3	2
7	wer erw	1	2

Выводы.

Изучили алгоритм Кнута-Морриса-Пратта поиска подстроки в строке. Реализовали программу на C++, которая для заданного текста находит все вхождения заданной строки.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

find.cpp:

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

void print_vector(std::vector<int> vec){
    for (int i : vec){
        std::cout << i;
    }
    std::cout << std::endl;
}

//Префикс функция
std::vector<int> prefix(const std::string& data){
    // Получаем длину строки
    std::cout << "Префикс функция получила на вход строку: " << data <<
std::endl;
    int n = data.length();
    //Инициализация вектора pi
    std::vector<int> pi(data.length(), 0);
    std::cout << "Вектор pi равен: ";
    print_vector(pi);
    // Проход по всему вектору с 1 элемента
    for(int i = 1; i < n; i++){
        std::cout << "Шаг: " << i << std::endl;
        int j = pi[i - 1];
        std::cout << "j = " << j << std::endl;
        //Выбор индекса из начала строки (строки которую ищем в другой)
        while(j > 0 && data[i] != data[j]){
            j = pi[j - 1];
            std::cout << "j = " << j << std::endl;
        }
        std::cout << "Сравнение элементов " << data[i] << " и " <<
data[j] << std::endl;
        if(data[i] == data[j]){
            std::cout << "pi[" << i <<"] = " << j+1 << std::endl;
            //Добавляем номер повторенной буквы в строке где ищем в
вектор префикс функции
            pi[i] = j + 1;
```

```
        }else{
            std::cout << "pi[" << i <<"] = " << j << std::endl;
            //Добавляем номер повторенной буквы в строке где ищем в
вектор префикс функции
            pi[i] = j;
        }
        std::cout << "Вектор pi на " << i << " шаге равен: ";
        print_vector(pi);
        std::cout << std::endl;
    }
    return pi;
}

std::vector<int> kmp(std::string& T, std::string& P){
    std::vector<int> res;
    //Составление строки для префикс функции
    std::string tmp = P + "#" + T;
    //Вызов префикс функции
    std::cout << "Вызов префикс функции" << std::endl;
    std::vector<int> pi = prefix(tmp);
    std::cout << "Префикс функция вернула: ";
    print_vector(pi);
    //Получает длину строки P
    int pLen = P.size();
    std::cout << "Длина строки P равна: " << pLen << std::endl;
    //Получает длину строки T
    int tLen = T.size();
    std::cout << "Длина строки T равна: " << tLen << std::endl;
    for(int i = 0; i < tLen; i++){
        std::cout << "Сравнение элементов " << "pi[pLen + 1 + i] ( = " <<
pi[pLen + 1 + i] << ") и pLen ( =" << pLen << ")" << std::endl;
        if(pi[pLen + 1 + i] == pLen){
            //Если значение в векторе префикс функции совпало с длиной
строки, то заносим в итоговый вектор элемент откуда началась строка
            std::cout << "Добавление в результирующий вектор индекса " <<
(i - pLen +1) << std::endl;
            res.push_back(i - pLen + 1);
        }
```

```
    }
    return res;
}

int main(){
    //Объявление переменных, где будут храниться входдные данные
    std::string P, T;
    //Считывание данных
    std::cin >> P >> T;
    std::cout << "Вы ввели:\n\t"
              << P << "\n"
              << "и\n\t"
              << T << "\n";
    //Вызов алгоритма КМП
    std::vector<int> res = kmp(T, P);
    std::cout << "Ответ: ";
    //Если выходной вектор пустой, то печатается -1
    if(res.empty()){
        std::cout << -1;
        return 0;
    }
    //Если вектор не пустой, то выводится на экран ответ
    for (int i = 0; i < res.size(); ++i) {
        if (i != res.size()-1){
            std::cout << res[i] << ',';
        } else{
            std::cout << res[i];
        }
    }
    return 0;
}
```

shift.cpp

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

```
void print_vector(std::vector<int> vec){
    for (int i : vec){
        std::cout << i;
    }
    std::cout << std::endl;
}

//Префикс функция
std::vector<int> prefix(const std::string& data){
    // Получаем длину строки
    std::cout << "Префикс функция получила на вход строку: " << data <<
std::endl;
    int n = data.length();
    //Инициализация вектора pi
    std::vector<int> pi(data.length(), 0);
    std::cout << "Вектор pi равен: ";
    print_vector(pi);
    // Проход по всему вектору с 1 элемента
    for(int i = 1; i < n; i++){
        std::cout << "Шаг: " << i << std::endl;
        int j = pi[i - 1];
        std::cout << "j = " << j << std::endl;
        //Выбор индекса из начала строки (строки которую ищем в другой)
        while(j > 0 && data[i] != data[j]){
            j = pi[j - 1];
            std::cout << "j = " << j << std::endl;
        }
        std::cout << "Сравнение элементов " << data[i] << " и " <<
data[j] << std::endl;
        if(data[i] == data[j]){
            std::cout << "pi[" << i <<"] = " << j+1 << std::endl;
            //Добавляем номер повторенной буквы в строке где ищем в
вектор префикс функции
            pi[i] = j + 1;
        }else{
            std::cout << "pi[" << i <<"] = " << j << std::endl;
```

```
            //Добавляем номер повторенной буквы в строке где ищем в
вектор префикс функции
            pi[i] = j;
        }
        std::cout << "Вектор pi на " << i << " шаге равен: ";
        print_vector(pi);
        std::cout << std::endl;
    }
    return pi;
}

std::vector<int> kmp(std::string& A, std::string& B){
    std::vector<int> res;
    //Составление строки для префикс функции
    std::string tmp = B + "#" + A + A;
    //Вызов префикс функции
    std::cout << "Вызов префикс функции" << std::endl;
    std::vector<int> pi = prefix(tmp);
    std::cout << "Префикс функция вернула: ";
    print_vector(pi);
    //Получает длину строки A
    int aLen = A.size();
    std::cout << "Длина строки A равна: " << aLen << std::endl;
    //Получает длину строки B
    int bLen = B.size();
    std::cout << "Длина строки B равна: " << bLen << std::endl;
    for(int i = 0; i < 2 * bLen; i++){
        std::cout << "Сравнение элементов " << "pi[aLen + 1 + i] ( = " <<
pi[aLen + 1 + i] << ") и aLen ( =" << aLen << ")" << std::endl;
        if(pi[aLen + 1 + i] == aLen){
            res.push_back(i - aLen + 1);
            //Если значение в векторе префикс функции совпало с длиной
строки, то заносим в итоговый вектор элемент откуда началась строка
            std::cout << "Добавление в результирующий вектор индекса " <<
(i - aLen +1) << std::endl;
        }
    }
    return res;
```

```
}

int main(){
    //Объявление переменных, где будут храниться входдные данные
    std::string A, B;
    //Считывание данных
    std::cin >> A >> B;
    std::cout << "Вы ввели:\n"
              << A << "\n"
              << "и\n"
              << B << "\n";
    //Вызов алгоритма КМП
    std::vector<int> res = kmp(A, B);
    std::cout << "Ответ: ";
    //Если выходной вектор пустой, то печатается -1
    if(res.empty()){
        std::cout << -1;
        return 0;
    }
    //Если вектор не пустой, то выводится на экран ответ

    std::cout << res[0];
    return 0;
}
```