

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. 9383

Моисейченко К.А.

Преподаватель

Фирсов М.А.

Санкт-Петербург

2021

Цель работы.

Изучить алгоритм Кнута-Морриса-Пратта поиска подстроки в строке.
Реализовать данный алгоритм на языке программирования C++.

Задание.

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

Вход:

Первая строка - Р Р

Вторая строка - ТТ

Выход:

индексы начал вхождений Р Р в ТТ, разделенных запятой, если РР не входит в ТТ, то вывести -1-1

Пример входных данных:

ab

abab

Пример выходных данных:

0,2

2. Заданы две строки АА ($|A| \leq 5000000$) и ВВ ($|B| \leq 5000000$).

Определить, является ли АА циклическим сдвигом ВВ (это значит, что АА и ВВ имеют одинаковую длину и АА состоит из суффикса ВВ, склеенного с префиксом ВВ). Например, defabc является циклическим сдвигом abcdef.

Вход:

Первая строка - АА

Вторая строка - ВВ

Выход:

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

Пример входных данных:

defabc

abcdef

Пример выходных данных:

3

Основные теоретические положения.

Префикс строки - какое-то количество начальных символов строки, но не равное всей строке.

Суффикс строки - какое-то количество конечных символов строки, но не равное всей строке.

Префикс-функция - длина наиболее длинного префикса, являющегося одновременно суффиксом, есть значение префикс-функции от строки для индекса j .

Выполнение работы.

Реализация алгоритма префикс-функции:

1. Элемент с индексом 0 всегда имеет значение префикс-функции равное 0.
2. Для работы алгоритма потребуются два указателя i и j . i отвечает за текущий индекс суффикса в строке, а j за текущий индекс префикса.
3. Если значения по индексам i и j не равны и $j = 0$, то по индексу i значение префикс-функции равно 0. Если $j \neq 0$, то по индексу j значение префикс-функции равно $j-1$.
4. Если значения по индексу i и j совпадают, то по индексу i значение префикс-функции равно $j+1$ и оба индекса i и j сдвигаются на единицу в сторону конца строки.
5. Алгоритм завершает работу, если символ i достиг конца строки.

Реализация алгоритма Кнута-Морриса-Пратта:

1. Считается префикс-функцию подстроки

2. Имеются два указателя `strInd` и `textInd`. `textInd` - текущий индекс элемента, с которым происходит сравнение строки, `strInd` - текущий индекс элемента подстроки
3. Если символы по текущим индексам совпадают, прибавляется по обоим индексам по единице и продолжается сравнение. Если индекс `strInd` достиг конца строки, значит одно из вхождений подстроки в строку найдено.
4. Если символы по текущим индексам не совпадают и `strInd != 0`, то в `strInd` кладется значение префикс-функции по индексу `strInd-1`. Если же `strInd=0`, то `textInd` увеличивается на единицу и алгоритм продолжает работу.
5. Алгоритм завершает работу, если индекс `textInd` достигает конца строки.

Сложность.

Временная сложность алгоритма можно оценить как $O(m+n)$, где n – длина подстроки, m - длина строки.

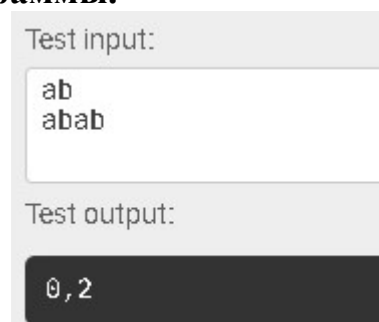
Описание функций и структур данных.

`std::vector<int> prefix(const std::string& text)` - алгоритм префикс-функции.

`std::vector<int> KMP(const std::string& text, const std::string& str)` - алгоритм Кнута-Морриса-Пракса.

`std::vector<int> KMP_2(const std::string& text, const std::string& str)` - алгоритм для решения задачи 2.

Примеры работы программы.



Test input:

ab
abab

Test output:

0, 2

Рисунок 1 - Пример работы программы из задания 1 №1.



Рисунок 2 - Пример работы программы из задания 1 №2.

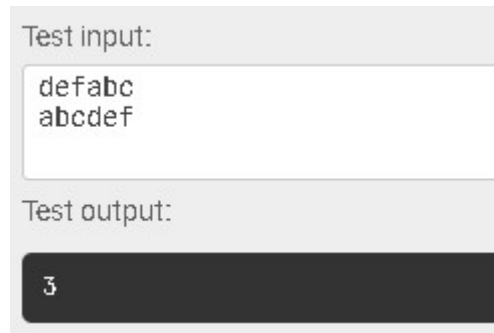


Рисунок 3 - Пример работы программы из задания 2 №1.

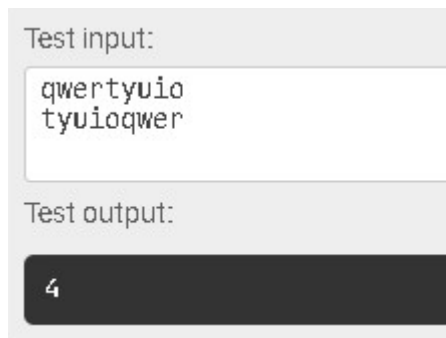


Рисунок 4 - Пример работы программы из задания 2 №2.

Выводы.

Изучен алгоритм Кнута-Морриса-Пратта – для поиска подстроки в строки и успешно написана программа, реализующая данный алгоритм.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

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

#define RES
//#define RES2

std::vector<int> prefix(const std::string& text) {
    int n = text.length();
    std::vector<int> pi(text.length(), 0);

    for (size_t i = 1; i < n; ++i) {
        size_t j = pi[i - 1];
        while (j > 0 && text[i] != text[j]) {
            j = pi[j - 1];
        }
        if (text[i] == text[j]) {
            pi[i] = j + 1;
        }
        else {
            pi[i] = j;
        }
    }
    return pi;
}

std::vector<int> KMP(const std::string& text, const std::string& str)
{
    std::vector<int> result;
    std::vector<int> pi = prefix(str);
    size_t strInd = 0;
    for (size_t textInd = 0; textInd < text.size(); textInd++) {
        if (text[textInd] == str[strInd]) {
            strInd++;
            if (strInd == str.size()) {
                result.push_back(textInd - str.size() + 1);
            }
            continue;
        }
        if (strInd == 0) continue;
        strInd = pi[strInd - 1];
        textInd--;
    }
    if (result.empty())
        result.push_back(-1);
    return result;
}

std::vector<int> KMP_2(const std::string& text, const std::string&
str) {
    const int LenText = text.length();
```

```

    const int LenStr = str.length();
    if (LenText != LenStr) {
        std::vector<int> result = { -1 };
        return result;
    }
    std::vector<int> result = KMP(str + str, text);
    return result;
}

int main() {
    std::vector<int> result;
    std::string str;
    std::string text;
    std::cin >> str;
    std::cin >> text;
#ifdef RES
    result = KMP(text, str);
    for (auto i = 0; i < result.size(); i++) {
        std::cout << result[i];
        if (i + 1 != result.size())
            std::cout << ",";
    }
#endif
#ifdef RES2
    result = KMP_2(text, str);
    std::cout << result[0];
#endif
    std::cout << std::endl;
    return 0;
}

```