

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ
по лабораторной работе №4
по дисциплине «Построение и анализ алгоритмов»
Тема: Алгоритм Кнута-Морриса-Пратта

Студентка гр. 8383

Ишанина Л.Н.

Преподаватель

Фирсов М.А.

Санкт-Петербург

2020

Цель работы.

Реализовать алгоритм Кнута-Морриса-Пратта, найти индексы вхождения подстроки в строку, а также разработать алгоритм проверки двух строк на циклический сдвиг.

Вариант 2.

Оптимизация по памяти: программа должна требовать $O(m)$ памяти, где m - длина образца. Это возможно, если не учитывать память, в которой хранится строка поиска.

Задание.

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.

Пример входных данных

aba

ababa

Пример выходных данных

0, 2

Описание структуры данных.

В данной программе используются векторы.

Вектор `std::vector<int> vector` – данный контейнер является аналогом массива $\pi[i]$, он хранит в себе максимальную длину совпадающих префикса и суффикса подстроки в образе, которая заканчивается i -м символом.

Вектор `std::vector<int> answer` – это вектор для записи в него ответа.

Описание алгоритма.

На вход алгоритма передается строка-образ, вхождения которой нужно найти, и строка-текст, в которой нужно найти вхождения.

Оптимизация – строка-текст считывается посимвольно, в памяти хранится текущий символ.

Алгоритм сначала вычисляет префикс-функцию строки-образа.(
`createPiArray(&vector, &string, string.length())`)

Далее посимвольно считывается строка-текст. В функции `void function(char c, std::string string, int k, int n, int count, std::vector<int>* answer, std::vector<int>* vector)` происходит следующее: изначально сравнивается рассматриваемый текущий символ(`char c`) строки-текста и текущий l -й элемент строки-образа.

В случае их равенства, происходит сдвиг l -го элемента строки-образа и также увеличивается переменная(`int count`), которая олицетворяет индекс, указывающий на символ в строке-тексте. Затем после того как выявилось совпадение символов, происходит проверка на равенство l номера строки-образа и переменной `n` (`n` – это длина строки образа), если это верно, то значит, что вхождение найдено и происходит запись индекса начала вхождения вектор ответа(`vector<int> answer`).

В случае, когда текущий символ(`char c`) строки-текста и текущий l -й элемент строки-образа не равны, то происходит проверка, не находится ли сейчас в начале(в нуле) индекс, указывающий на текущий элемент строки-образа. Если это верно, увеличиваем на единицу индекс, который указывает на символ в строке-тексте. Иначе, если индекс не равен 0, то происходит перемещение позиции индекса l из одной позиции в другую.

Алгоритм завершает работу по окончании строки-текста.

Описание main () :

В функции прописан ввод строки-образа(т.е.вхождение которой программа будет искать) и посимвольное считывание строки-текста(т.е. где будет совершаться поиск), а также вызов функции для составления массива `pi` для заданного образа и функции нахождения образа в тексте, а также вывод ответа.

Кроме того, для удобства, во время работы алгоритма, происходит вывод промежуточной информации.

Описание дополнительных функций.

Функция `void createPiArray(std::vector<int>* vector, std::string* string, int length)` принимает на вход вектор `vector`, в который будет записываться максимальная длина совпадающих префикса и суффикса, строка-образ, и длина строки-образа. Данная префикс-функция для i -го символа образа сопоставляет значение, равное максимальной длине совпадающих префикса и суффикса подстроки в образе, которая заканчивается i -м символом. Именно это значение(длина) будет храниться в векторе `vector`, который будет использоваться в функции самого алгоритма.

Сложность алгоритма.

Сложность алгоритма по времени: $O(m + n)$, m – длина образа, n – длина текста.

Сложность алгоритма по памяти: $O(m)$, m – длина образа, так как программа хранит только строку-образ, которая считывается в самом начале.

Тестирование.

Входные данные:

ab

abab

Вывод:

```
Консоль отладки Microsoft Visual Studio
Здравствуйте! Введите, пожалуйста, строку-образец, вхождение которой предстоит искать!
ab
Вызывается функция для составления массива pi для заданного образа!
Функция для составления массива pi для заданного образа завершает работу!
Пожалуйста, введите строку-текст, для дальнейшего поиска в ней!
abab
Был считан символ a
Запускается цикл проверки на вхождение!
c = a
Запускается цикл проверки на вхождение!
c = b
Вхождение строки найдено! 0
Запускается цикл проверки на вхождение!
Двигаем строку.
Запускается цикл проверки на вхождение!
c = a
Запускается цикл проверки на вхождение!
c = b
Вхождение строки найдено! 2
0,2
D:\4 сем\ПиАА\lr4\Project1\Debug\Project1.exe (процесс 23204) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрывать консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...
```

Тест №2

Входные данные:

time

hitiime

Вывод:

```
Консоль отладки Microsoft Visual Studio
Здравствуйте! Введите, пожалуйста, строку-образец, вхождение которой предстоит искать!
time
Вызывается функция для составления массива pi для заданного образа!
Функция для составления массива pi для заданного образа завершает работу!
Пожалуйста, введите строку-текст, для дальнейшего поиска в ней!
hitiime
Был считан символ h
Запускается цикл проверки на вхождение!
c: h != t
Запускается цикл проверки на вхождение!
c: i != t
Запускается цикл проверки на вхождение!
c = t
Запускается цикл проверки на вхождение!
c = i
Запускается цикл проверки на вхождение!
Двигаем строку.
Запускается цикл проверки на вхождение!
c: i != t
Запускается цикл проверки на вхождение!
c: m != t
Запускается цикл проверки на вхождение!
c: e != t
Не было найдено ни одного вхождения образа в тексте.
-1
D:\4 сем\ПиАА\lr4\Project1\Debug\Project1.exe (процесс 16312) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрывать консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...
```

Таблица 1 – Результаты тестирования

Ввод	Вывод
ab abab	0,2
ababab ababab	0
work workworkwork	0,4,8
aba ababababa	0,2,4,6

Вывод.

В ходе работы был построен и анализирован алгоритм КМП. Код программы представлен в приложении А.

ПРИЛОЖЕНИЕ А.

ИСХОДНЫЙ КОД

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

void createPiArray(std::vector<int>* vector, std::string* string, int length) {

    int j = 0;
    int i = 1;

    vector->emplace_back(0); //для первого символа любого образа записываем всегда 0

    while (length > i) {
        if (string->at(i) == string->at(j)) {
            vector->emplace_back(j + 1);
            i++;
            j++;
        }
        else {
            if (j == 0) {
                vector->emplace_back(0);
                i++;
            }
            else {
                j = vector->at(j - 1);
            }
        }
    }
}

void function(char c, std::string string, int l, int n, int count, std::vector<int>* answer,
std::vector<int>* vector)
{
    while (true) {
        std::cout << "Запускается цикл проверки на вхождение!" << std::endl;
        bool isCinActive = true; //флаг, необходимый для считывания

        if (c == string[l]) { //проверка совпадает ли текущий символ текста с символом строки-
образца
            std::cout << "c = " << c << std::endl;
            l++;
            count++;
            if (l == n) {
                std::cout << "Вхождение строки найдено!      " << count - n << std::endl;
                answer->emplace_back(count - n);
            }
        }
        else {
            if (l == 0) {
                std::cout << "c:  " << c << " != " << string[l] << std::endl;
                count++;
            }
            else {
                std::cout << "Двигаем строку." << std::endl;
                l = vector->at(l - 1);
                isCinActive = false;
            }
        }
    }

    if (isCinActive) {
        std::cin.get(c);
    }
}
```

```

        if (c == '\n') {
            break;
        }
    }
}

int main() {
    setlocale(LC_ALL, "Russian");
    std::string string;
    std::cout << "Здравствуй! Введите, пожалуйста, строку-образец, вхождение которой
предстоит искать!" << std::endl;
    std::cin >> string;

    std::vector<int> vector;
    vector.reserve(0);

    std::vector<int> answer;
    vector.reserve(0);
    std::cout << "Вызывается функция для составления массива pi для заданного образа!" <<
std::endl;
    createPiArray(&vector, &string, string.length());
    std::cout << "Функция для составления массива pi для заданного образа завершает работу!" <<
std::endl;
    std::cout << "Пожалуйста, введите строку-текст, для дальнейшего поиска в ней!" <<
std::endl;
    char c;
    std::cin.get(c); //первый раз, чтобы считать \n

    int l = 0;
    int n = string.size();

    int count = 0;
    std::cin.get(c); //считываем первый символ строки-текста
    std::cout << "Был считан символ " << c << std::endl;

    function(c, string, l, n, count, &answer, &vector);
    if (!answer.empty()) {
        for (size_t m = 0; m < answer.size(); ++m) { //вывод ответа
            std::cout << answer[m];
            if (m != answer.size() - 1)
                std::cout << ",";
        }
    }
    else {
        std::cout << "Не было найдено ни одного вхождения образа в тексте." << std::endl;
        std::cout << -1 << std::endl;
    }

    return 0;
}

```