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. 7383 _____ Власов Р.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2019

Содержание

Цель работы

Цель работы: познакомиться с алгоритмом поиска подстроки в строке Кнута-Морриса-Пратта, создать программу, осуществляющую поиск подстроки в строке с помощью алгоритма Кнута-Морриса-Пратта и проверку строк на циклический сдвиг.

Формулировка задачи: 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. 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. Вариант 2. Оптимизация по памяти: программа должна требовать O(m) памяти, где m - длина образца. Это возможно, если не учитвать память, в которой хранится строка поиска.

Реализация задачи

Программу было решено писать на языке программирования C++.

Для реализации поставленной задачи были созданы функции `prefix_function` и `find_shift`.

Функция `std::vector<size_t> prefix_function(const std::string& p, const std::string& t)` осуществляет поиск строки `p` в строке `t` с помощью алгоритма Кнута-Морриса-Пратта. Для этого она вычисляет префикс-функцию для строки `p` и строки `t`, опираясь на результат строки `p`. В выходном массиве символы с результатом, равным длине строки `p` являются концом ее вхождения в данную строку.

Функция `int check_shift(const std::string& a, const std::string& b)` возвращает индекс начала строки a в строке .

Исходный код программы представлен в приложении Б.

Исследование алгоритма

Сложность алгоритма Кнута-Морриса-Пратта линейная и совпадает со сложностью вычисления префикс-функции. Сложность алгоритма поиска цикла равно $O(n^2)$, где n - длина строк.

Тестирование

1. Процесс тестирования

Программа собрана в операционной системе Ubuntu 18.04.2 LTS bionic компилятором g++ version 7.3.0 (Ubuntu 7.3.0-27ubuntu1~18.04). В других ОС и компиляторах тестирование не проводилось.

2. Результаты тестирования

В результате тестирования программы ошибок выявлено не было. Тестовые случаи представлены в приложении А.

Вывод

В ходе выполнения данной работы был изучен метод поиска подстроки в строке с помощью алгоритма Кнута-Морриса-Пратта. Была написана программа, применяющая алгоритм Кнута-Морриса-Пратта для поиска подстроки в строке, а также осуществляющая поиск циклического сдвига в двух строках. Сложность алгоритма Кнута-Морриса-Пратта составляет O(n), а алгоритма поиска цикла - $O(n^2)$.

ПРИЛОЖЕНИЕ А. ТЕСТОВЫЕ СЛУЧАИ

Входные данные	Результат
ab abab	0,2
aa caaaa	1,2,3
caa caaaa	1

Входные данные	Результат
defabc abcdef	3
aaa aa	-1
abcdef cdefba	-1

ПРИЛОЖЕНИЕ Б. ИСХОДНЫЙ КОД

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

std::vector<size_t> prefix_function(const std::string& p, const
std::string& t)
{
    std::vector<size_t> pref_t(p.length());
    pref_t[0] = 0;
    size_t k = 0;
    for (size_t i = 1; i < p.length(); i++)
    {

        while (k > 0 && p[k] != p[i])
            k = pref_t[k-1];
        if (p[k] == p[i])
            k++;
        pref_t[i] = k;
    }

    std::vector<size_t> pref(t.length());
    k = (p[0] == t[0] ? 1 : 0);
    pref[0] = k;
    for (size_t i = 1; i < t.length(); i++)
    {

        while (k >= p.length() || (k > 0 && p[k] != t[i]))
            k = pref_t[k-1];
        if (p[k] == t[i])
            k++;
        pref[i] = k;
    }
    return pref;
}

void print_ans(size_t l, std::vector<size_t> pref)
{
    std::vector<size_t> a;
    for (size_t i = 0; i < pref.size(); i++)
    {
        if (pref[i] == l)
            a.push_back(i - l + 1);
    }
    if (a.size())
    {
```

```
        std::cout << a[0];
        for (size_t i = 1; i < a.size(); i++)
            std::cout << ',' << a[i];
    }
    else {
        std::cout << -1;
    }
}

int check_shift(const std::string& a, const std::string& b)
{
    bool check = true;
    if (a.length() != b.length())
        return -1;
    for (size_t  i = 0; i < a.length(); i++)
    {
        check = true;
        if (a[0] == b[i])
        {
            for (size_t j = 0; j < a.length(); j++)
            {
                if (a[j] != b[(i + j) % a.length()])
                {
                    check = false;
                    break;
                }
            }
            if (check)
                return (int)i;
        }
    }
    return -1;
}

int main()
{
    std::string P;
    std::string T;
    std::cin >> P;
    std::cin >> T;
    std::vector<size_t> pref = prefix_function(P, T);
    print_ans(P.length(), pref);
    std::cout << check_shift(T, P);

    return 0;
}
```