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 Кнутта-Морриса-Пратта

Студент гр. 8304 _____ Бутко А.М.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург
2020

Цель работы.

Реализовать алгоритм Кнута-Морриса-Пратта, оптимизировать программу по памяти.

Постановка задачи.

(Вариант 2)

Оптимизация по памяти: программа должна требовать O(m) памяти, где m — длина образца. Реализуйте алгоритм КМП и с его помощью для заданных шаблона P и текста найдите все вхождения P в T.

Реализация алгоритма.

Для оптимизации алгоритма по памяти будем считывать текст посимвольно. Алгоритм использует префикс-функцию, которая считает, сколько символов совпало у префикса строки P и у строки T, заканчивающимся в i-ой позиции.

Оценка сложности алгоритма.

Сложность алгоритма по памяти O(m) , где m — длина P.

Сложность алгоритма по времени O(m + n), где n — суммарная длина текста.

Описание структур данных и функций.

`std::vector<int> getPrefix(std::string pattern)`–функция, возвращающая префикс строки pattern.

`void algorithmKMP(std::string pattern, std::vector<int>& result)` — функция, выполняющая вызов префикс-функции и осуществляющая поиск вхождений строки P в строку T.

Тестирование.

Ввод	Вывод
ab abab	0, 2
ab dadada	-1
abc abcdabcdabceabc	0, 4, 8, 12

Вывод.

В ходе выполнения работы был найден максимальный поток в сети, а также фактическая величиа потока, протекающего через каждое ребро, используя алгоритм Форда-Фалкерсона.

ПРИЛОЖЕНИЕ А.
ИСХОДНЫЙ КОД.

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

std::vector<int> getPrefix(std::string pattern)
{
    std::vector<int> prefix{0};
    int index = 0;

    for (int i = 1; i < pattern.size(); ++i)
    {
        while (index != -1 && pattern[index] != pattern[i]) index--;
        index++;
        prefix.push_back(index);
    }
    return prefix;
}

void algorithmKMP(std::string pattern, std::vector<int>& result)
{
    int res = -1;
    std::vector<int> prefix = getPrefix(pattern);
    int index = 0;
    char symb;
    int counter = 0;
    while (std::cin >> symb)
    {
        counter++;
        while (index > 0 && pattern[index] != symb) index = prefix[index - 1];
        if (pattern[index] == symb) index++;
        if (index == pattern.size())
        {
            res = counter - index + 1;
            result.push_back(res);
        }
    }
    if (result.empty()) result.push_back(res);
}

void output(std::vector<int> result)
{
    for(int i = 0; i < result.size(); ++i)
    {
        std::cout << result[i];
        if (i + 1 != result.size()) std::cout << ",";
    }
}

int main()
{
    std::string prefix;
    std::vector<int> result;

    std::getline(std::cin, prefix);

    algorithmKMP(prefix, result);
```

```
    output(result);
}
```