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

Студент гр. 7383 _____ Александров Р.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2019

Цель работы.

Познакомиться с алгоритмом Кнута-Морриса-Пратта и его реализацией.

Постановка задачи.

Реализовать алгоритм КМП и с его помощью для заданных шаблона P ($|P| \leq 15000$) и текста T ($|T| \leq 5000000$) найти все вхождения P в T.

Входные данные

Первая строка - P

Вторая строка - T

Выходные данные

Индексы начал вхождений P в T, разделенных запятой, если P не входит в T, то вывести −1.

Реализация задачи.

В ходе работы для алгоритма Кнута-Морриса-Пратта были написаны методы string runKmp(string pat, string txt) и void getLpsArray(string pat, int patLength, int *lps).

Метод runKmp вызывает метод getLpsArray для получения значения префикс-функции для строки-паттерна pat и ищет индексы первых вхождений паттерна pat в текст txt.

Метод runShift(string pat, string txt) на основе значений префикс-функции для строки-паттерна pat определяет, является ли pat циклическим сдвигом txt (это значит, что pat и txt имеют одинаковую длину и pat состоит из суффикса txt, склеенного с префиксом txt).

Исходный код представлен в приложении А.

Исследование алгоритма.

Время работы алгоритма Кнута-Морриса-Пратта оценивется как O(P+T), где P – длина строки-паттерна, T – длина текста. Алгоритм имеет аналогичную оценку по памяти.

Результаты работы алгоритма представлены в табл. 1.

Таблица 1 - Результаты работы алгоритма

Вход	Выход
ab abab	0,2
fff asdqwfffqwecx	5
fa safsafasfsffafvvcxxcfafxvxcxxczcfafxvxzv vxfa	5,11,20,32,42
123 zcxzvcbvxbx	-1

Выводы.

В ходе лабораторной работы был изучен и реализован алгоритм Кнута-Морриса-Пратта, осуществляющий поиск подстроки в строке за линейное время.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

```
#include <iostream>
#include <fstream>

using namespace std;

void getLpsArray(string pat, int patLength, int *lps) {
    // length of the previous longest prefix suffix
    int len = 0;
    int i = 1;
    // always 0
    lps[0] = 0;
    while (i < patLength) {
        if (pat[i] == pat[len]) {
            len++;
            lps[i] = len;
            i++;
        } else {
            if (len != 0) {
                len = lps[len - 1];
            } else {
                lps[i] = len;
                i++;
            }
        }
    }
}

string runKmp(string pat, string txt) {
    string answer;
    int patLength = pat.length();
    int txtLength = txt.length();
    int* lps = new int[pat.size()];
```

```
    int j = 0;

    getLpsArray(pat, patLength, lps);
    int i = 0;
    while (i < txtLength) {
        if (pat[j] == txt[i]) {
            j++;
            i++;
        }
        if (j == patLength) {
            string f = to_string(i - j) + ",";
            answer.append(f);
            j = lps[j - 1];
        } else if (pat[j] != txt[i]) {
            if (j != 0) {
                j = lps[j - 1];
            } else {
                i += 1;
            }
        }
    }
    delete[] lps;

    return answer;
}


int runShift(string pat, string txt) {
    int answer = -1;
    if (pat.length() != txt.length()) {
        return answer;
    }
    string doublePatText = pat + pat;
    int* lps = new int[pat.size()];
```

```
    getLpsArray(pat, pat.size(), lps);
    int index = 0;
    for (int i = 0; i < doublePatText.size(); i++) {
        while (index > 0 && txt[index] != doublePatText[i]) {
            index = lps[index - 1];
        }
        if (txt[index] == doublePatText[i]) {
            index++;
        }
        if (index == txt.size()) {
            answer = (i + 1 - txt.length());
            break;
        }
    }
    delete[] lps;

    return answer;
}

int main() {
    string p;
    string t;
    cin >> p >> t;
    string answer = runKmp(p, t);
    if (answer.length() == 0) {
        cout << "-1" << endl;
    } else {
        for (int i = 0; i < answer.length() - 1; i++) {
            cout << answer[i];
        }
        cout << endl;
    }
    return 0;
}
```