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. 8304 Рыжиков А.В.

Преподаватель Размочаева Н.В.

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 алгоритма.

На вход алгоритма передается строка-образец, 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-образца.

Далее посимвольно считывается строка-текст. Переменная-счетчик изначально k = 0. При каждом совпадении k-го символа образца и i-го символа текста счетчик увеличивается на 1. Если k = размер образца, значит вхождение найдено. Если очередной символ текста не совпал с k-ым символом образца, то сдвигаем образец, причем точно знаем, что первые k символов образца совпали с

символами строки и надо сравнить k+1-й символ образца (его индекс k) с i-м символом строки.

Сложность алгоритма по операциям: O (m + n), m – длина образца, n – длина текста.

Сложность алгоритма по памяти: O (m), m – длина образца.

Вывод промежуточной информации.

Во время основной части работы алгоритма происходит вывод промежуточной информации, а именно, значения префикс функции и проверка идентичности префикса и суффикса первой и второй строки соответственно.

Тестирование.

Таблица 1 – Результаты тестирование

Ввод	**Вывод**
ab abab	0,2
abababab abababab	0
alex alexalexalex	0,4,8
aba abababababa	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>
#include <fstream>

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

    int j = 0;
    int i = 1;

    vector->emplace_back(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 Kmp(std::istream &fin, int your_choose){


    std::ofstream fin2;
    fin2.open("C:\\Users\\Alex\\Desktop\\out.txt");

    std::string string;

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

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



    fin >> string;

    createPiArray(&vector, &string, string.length());

    char c;
    fin.get(c);

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

    int count = 0;
    fin.get(c);
    while (true) {
        bool isCinActive = true;
```

```
    if (c == string[l]) {
        if (your_choose==1){
            fin2 << c << "==" << string[l] << '\n';
        }
        std::cout << c << "==" << string[l] << '\n';
        l++;
        count++;
        if (l == n) {
            std::cout << "Occurrence of the string found ! " << count - n << '\n';
            if (your_choose==1){
                fin2 << "Occurrence of the string found ! " << count - n << '\n';
            }
            answer.emplace_back(count - n);
        }
    } else {
        if (l == 0) {
            if (your_choose==1){
                fin2 << c << "!=" << string[l] << '\n';
            }
            std::cout << c << "!=" << string[l] << '\n';
            count++;
        } else {
            if (your_choose == 1){
               fin2 << "moving the string " << '\n';
            }
            std::cout << "moving the string " << '\n';
            l = vector.at(l - 1);
            isCinActive = false;
        }
    }

    if (isCinActive) {
        fin.get(c);
    }

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

if (!answer.empty()) {
    for (size_t m = 0; m < answer.size(); ++m) {
        std::cout << answer[m];
        if (m != answer.size() - 1) std::cout << ",";
    }
} else {
    std::cout << -1;
}

if (your_choose == 1){


    if (!answer.empty()) {
        for (size_t m = 0; m < answer.size(); ++m) {
            fin2 << answer[m];
            if (m != answer.size() - 1) fin2 << ",";
        }
```

```
        } else {
            fin2 << -1;
        }
    }
}

int main() {

    int your_choose = 0;

    std::cout << "If you want to enter data from a file, enter \'1\'\n";

    std::cin >> your_choose;

    if (your_choose == 1) {
        std::ifstream fin;
        fin.open("C:\\Users\\Alex\\Desktop\\in.txt");

        if (fin.is_open()) {
            std::cout << "Reading from file:" << "\n";
            Kmp(fin,your_choose);
        } else {
            std::cout << "File not opened";
        }

        fin.close();
    } else {
        Kmp(std::cin,your_choose);
    }

    return 0;
}
```