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

Студентка гр. 8304 _____ Мельникова О.А.

Преподаватель _____ Размочаева Н. В.

Санкт-Петербург

2020

Цель работы.

Изучить алгоритм Кнута-Морриса-Пратта для оптимального поиска всех вхождений подстроки в строку.

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.

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;
- 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);

Описание алгоритма.

Оптимизация — строка-текст считывается посимвольно.

Сложность алгоритма O(|P| + |T|).

1. Считать значения префикс-функции $\pi[i]$ будем по очереди: от $i = 1$ к $i = n - 1$ (значение $\pi[0]$ просто присвоим равным нулю).
2. Для подсчёта текущего значения $\pi[i]$ мы заводим переменную j, обозначающую длину текущего рассматриваемого образца. Изначально $j = \pi[i - 1]$.
3. Тестируем образец длины j, для чего сравниваем символы $s[j]$ и $s[i]$. Если они совпадают — то полагаем $\pi[i] = j + 1$ и переходим к следующему индексу $i + 1$. Если же символы отличаются, то уменьшаем длину j, полагая её равной $\pi[j - 1]$, и повторяем этот шаг алгоритма с начала.

4. Если мы дошли до длины $j = 0$ и так и не нашли совпадения, то останавливаем процесс перебора образцов и полагаем $\pi[i] = 0$ и переходим к следующему индексу $i + 1$.

Описание функций и структур данных.

void prefixFunction (std::vector<int>& vectorPi, const std::string& str)	Функция для вычисления префикс-функции строки.
void **KMP**(std::istream& input, std::ostream& output)	Функция поиска всех вхождений
void **KMP**(std::string& B, std::string& A, std::ostream& fout)	Функция проверки на циклический сдвиг

Тестирование.

Входные данные: aaba aabaabaabaaababbabbaabaa	Результат работы программы: Начало работы алгоритма Кнута-Морриса-Пратта Совпадение!!! Элемент шаблона: a; Элемент текста: a Совпадение!!! Элемент шаблона: a; Элемент текста: a Совпадение!!! Элемент шаблона: b; Элемент текста: b Совпадение!!! Элемент шаблона: a; Элемент текста: a Вхождение найдено, индекс начала: 0 Совпадение!!! Элемент шаблона: a; Элемент текста: a Совпадение!!! Элемент шаблона: b; Элемент текста: b

	Совпадение!!! Элемент шаблона: a; Элемент текста: a Вхождение найдено, индекс начала: 3 Совпадение!!! Элемент шаблона: a; Элемент текста: a Совпадение!!! Элемент шаблона: b; Элемент текста: b Совпадение!!! Элемент шаблона: a; Элемент текста: a Вхождение найдено, индекс начала: 6 Совпадение!!! Элемент шаблона: a; Элемент текста: a Совпадение!!! Элемент шаблона: a; Элемент текста: a Совпадение!!! Элемент шаблона: b; Элемент текста: b Совпадение!!! Элемент шаблона: a; Элемент текста: a Вхождение найдено, индекс начала: 10 Совпадение!!! Элемент шаблона: a; Элемент текста: a Совпадение!!! Элемент шаблона: a; Элемент текста: a Совпадение!!! Элемент шаблона: a; Элемент текста: a Совпадение!!! Элемент шаблона: b; Элемент текста: b Совпадение!!! Элемент шаблона: a; Элемент текста: a Вхождение найдено, индекс начала: 15

	Конец работы алгоритма Кнута-Морриса-Пратта
Входные данные: qwertyabcdef abcdefqwerty	Результат работы программы: Начало работы алгоритма Кнута-Морриса-Пратта Элемент образца a; Элемент цепочки q Элемент образца a; Элемент цепочки w Элемент образца a; Элемент цепочки e Элемент образца a; Элемент цепочки r Элемент образца a; Элемент цепочки t Элемент образца a; Элемент цепочки y Элемент образца a; Элемент цепочки a Элемент образца b; Элемент цепочки b Элемент образца c; Элемент цепочки c Элемент образца d; Элемент цепочки d Элемент образца e; Элемент цепочки e Элемент образца f; Элемент цепочки f Элемент образца q; Элемент цепочки q Элемент образца w; Элемент цепочки w Элемент образца e; Элемент цепочки e Элемент образца r; Элемент цепочки r Элемент образца t; Элемент цепочки t Элемент образца y; Элемент цепочки y Длины совпали - индекс начала: 6

Выводы.

В ходе выполнения лабораторной работы был реализован алгоритм КМП. Также проанализирована сложность алгоритма.

ПРИЛОЖЕНИЕ А.
ИСХОДНЫЙ КОД

#include <iostream>

```
#include <vector>
#include <string>
#include <fstream>
void prefixFunction (std::vector<int>& vectorPi, const std::string& str){
    vectorPi[0] = 0;
    for (int i=1; i<str.length(); i++) {
        int curPrefixLenght = vectorPi[i-1];
        while ((str[i] != str[curPrefixLenght]) && (curPrefixLenght > 0))
curPrefixLenght = vectorPi[curPrefixLenght-1];
        if (str[i] == str[curPrefixLenght]) curPrefixLenght++;
        vectorPi[i] = curPrefixLenght;
    }
}
void KMP(std::istream& input, std::ostream& output){
    output << "Начало работы алгоритма Кнута-Морриса-Пратта" << std::endl;
    bool is_result = false;
    std::string str;
    input >> str;
    char tmp = '1';
    std::vector<int> vectorPi(str.length());
    prefixFunction(vectorPi, str);
    size_t j = 0;
    size_t counter = 0;
    input.get();
    input.get(tmp);
    while(input.peek() != EOF && input.peek() != '\n'){
        while (str[j] == tmp) { output<<"Совпадение!!! Элемент шаблона:
"<<str[j]<<"; Элемент текста: " << tmp<< std::endl; input.get(tmp); j++;
counter++; }
        if (j == str.length()){
            output <<"\tВхождение найдено, индекс начала: " << counter - j <<
std::endl;
            j = vectorPi[j-1];
            is_result = true;
        }else{
            if(j != 0) j = vectorPi[j-1];
            else input.get(tmp);
        }
    }
    if (!is_result) output << "Вхождений не найдено" << std::endl;
    output << "Конец работы алгоритма Кнута-Морриса-Пратта" << std::endl;
}
int main() {
    char in, out;
    std::cout << "Для считывания/вывода через консоль введите - 'c', через
файл - 'f' \n";
    std::cin >> in >> out;
    std::ifstream input("input.txt");
    std::ofstream output("output.txt");
    if((out != 'c' && out != 'f') || (in != 'c' && in != 'f')) { std::cout <<
"Неверный ввод\n"; return 1; }
    if((in == 'c') && (out == 'c')) KMP(std::cin, std::cout);
```

```
    if((in == 'f') && (out == 'f')) KMP(input, output);
    if((in == 'f') && (out == 'c')) KMP(input, std::cout);
    if((in == 'c') && (out == 'f')) KMP(std::cin, output);
    return 0;

}
```

ПРИЛОЖЕНИЕ В.
ИСХОДНЫЙ КОД

```
#INCLUDE <IOSTREAM>
#include <string>
#include <vector>
#include <fstream>

void prefixFunction (std::vector<int>& vectorPi, const std::string& str){
    vectorPi[0] = 0;
    for (int i=1; i<str.length(); i++) {
        int curPrefixLenght = vectorPi[i-1];
        while ((str[i] != str[curPrefixLenght]) && (curPrefixLenght > 0))
curPrefixLenght = vectorPi[curPrefixLenght-1];
        if (str[i] == str[curPrefixLenght]) curPrefixLenght++; // равны
        vectorPi[i] = curPrefixLenght;
    }
}

void KMP(std::string& B, std::string& A, std::ostream& fout){
    fout << "Начало работы алгоритма Кнута-Морриса-Пратта" << std::endl;
    std::vector<int> vectorPi(B.length());
    prefixFunction(vectorPi, B);
    int matching = 0;
    for (int i = 0; i<A.length(); ++i){
        fout << "Элемент образца " << B[matching]
        << "; Элемент цепочки " << A[i] << std::endl;
        while ((B[matching] != A[i]) && (matching > 0)) matching =
vectorPi[matching - 1];
        if (A[i] == B[matching]) matching++;
        if (matching == B.length()){ fout<<"Длины совпали - индекс начала:
"<< i - B.length() + 1; return; }
    }
    fout<<"Не циклический сдвиг";
}

int main(){
    char in, out;
    std::cout << "Для считывания/вывода через консоль введите - 'c', через
файл - 'f' \n";
    std::cin >> in >> out;
    if((out != 'c' && out != 'f') || (in != 'c' && in != 'f')) { std::cout <<
"Неверный ввод\n"; return 1; }
    std::string A, B;
    if(in == 'c') std::cin >> A >> B;
    else{ std::ifstream input("input.txt"); input >> A >> B; }
    A += A;
    if(out == 'c') KMP(B, A, std::cout);
    else{ std::ofstream output("output.txt"); KMP(B, A, output); }
    return 0;
}
```