**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*∣≤15000) и текста *T* (∣*T*∣≤5000000) найти все вхождения *P* в *T*;
* Определить, является ли А циклическим сдвигом В (это значит, что А и В имеют одинаковую длину и А состоит из суффикса В, склеенного с префиксом В);

**Описание алгоритма.**

Оптимизация — строка-текст считывается посимвольно.

Сложность алгоритма О(|P| + |T|).

1. Считать значения префикс-функции  будем по очереди: от  к  (значение  просто присвоим равным нулю).
2. Для подсчёта текущего значения  мы заводим переменную , обозначающую длину текущего рассматриваемого образца. Изначально .
3. Тестируем образец длины , для чего сравниваем символы  и . Если они совпадают — то полагаем  и переходим к следующему индексу . Если же символы отличаются, то уменьшаем длину , полагая её равной , и повторяем этот шаг алгоритма с начала.
4. Если мы дошли до длины  и так и не нашли совпадения, то останавливаем процесс перебора образцов и полагаем  и переходим к следующему индексу .

**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 A.  
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 B.  
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;

}