**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: **Кнут-Моррис-Пратт**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9303 |  | Павлов Д.Р. |
| Преподаватель |  | Шевская Н.В. |

Санкт-Петербург

2021

## Цель работы.

Изучить алгоритм Кнута-Морриса-Пратта поиска подстроки в строке.

## Задание.

1. Реализуйте алгоритм КМП и с его помощью для заданных шаблона *P* (∣P∣≤15000∣*P*∣≤15000) и текста *T* (∣T∣≤5000000∣*T*∣≤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

2) Заданы две строки *A*(|*A*|*≤*500000) и *B*(|*B*|*≤* 500000).  
 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* ). Например, defabc является циклическим сдвигом abcdef.

Вход:  
Первая строка – *A*Вторая строка – *B*Выход:  
Если является *A* циклическим сдвигом *B*, индекс начала строки *B* в *A*,

иначе вывести -1. Если возможно несколько сдвигов вывести первый индекс.

## Выполнение работы.

lps\_func(string txt, vector<int>&lps) – Префикс.

KMP(string pattern, string text, list<size\_t>&result) – Алгоритм Кнута-Морриса-Пратта.

kmp\_cycle(std::string& pat, std::string& text) – Алгоритм, который находит циклический сдвиг.

Разработанный программный код см. в приложении А.

Результаты тестирования см. в приложении Б.

**Описание алгоритма.**

Для нахождения всех вхождений шаблона P в текст T, надо посчитать значения префикс-функции для строки P#T. Если функция содержит значения, равные длине P, то P входит в T.

Для определения, является ли строка A циклическим сдвигом строки B, надо посчитать значение префикс-функции для строки B#AA.

**Анализ алгоритма.**

Пусть m — длина строки, для которой вычисляется префикс-функция, n — длина текста. Тогда сложность алгоритма будет равняться *O*(*n*+*m*) .

## Выводы.

Был изучен и реализован алгоритм Кнута-Морриса-Пратта для поиска всех вхождений подстроки в строке. Также на основе данного алгоритма был реализован алгоритм, который определяет является ли заданная строка циклическим сдвигом другой строки.

# Приложение А Исходный код программы

Название файла: KMP.hpp

#include <iostream>

#include <vector>

#include <list>

using namespace std;

void lps\_func(string txt, vector<int>&Lps){

Lps[0] = 0;

int len = 0;

int i=1;

while (i<txt.length()){

if(txt[i]==txt[len]){

len++;

Lps[i] = len;

i++;

continue;

}

else{

if(len==0){

Lps[i] = 0;

i++;

continue;

}

else{

len = Lps[len-1];

continue;

}

}

}

}

void KMP(string pattern,string text, list<size\_t>&result){

size\_t n = text.length();

size\_t m = pattern.length();

vector<int>Lps(m);

lps\_func(pattern,Lps);

int i=0,j=0;

while(i<n){

if(pattern[j]==text[i]){i++;j++;}

if (j == m) {

result.push\_back(i-m);

j = Lps[j - 1];

}

else if (i < n && pattern[j] != text[i]) {

if (j == 0)

i++;

else

j = Lps[j - 1];

}

}

}

Название файла: cycle.hpp

#include <string>

#include <vector>

#include <iostream>

auto kmp\_cycle(std::string& pat, std::string& text){

auto res = -1;

if (pat.length() != text.length()){

return res;

}

std::string s = text + "#" + pat + pat;

size\_t len\_text = text.length();

size\_t len\_s = s.length();

int j = 0;

for (size\_t i = len\_text + 1; i < len\_s; i++){

size\_t k = i;

while (s[j] == s[k]){

j++;

k++;

if (j == len\_text){

res = k - 2\*len\_text - 1;

return res;

}

}

j = 0;

}

return res;

}

Название файла: main.cpp

#include <iostream>

#include <list>

#include "cycle.hpp"

#include "KMP.hpp"

#include <fstream>

int main(){

string pattern;

string text;

list<size\_t> res = {};

int consoleOrFile;

cout << "1 -- Console" << endl << "2 -- File" << endl;

cin >> consoleOrFile;

if (consoleOrFile == 1){

cout << "Input pattern : ";

cin >> pattern;

cout << "Input text : ";

cin >> text;

int choose = 0;

cout << "1 -- index\n2 -- cycle\n";

cin >> choose;

if (choose == 1) {

KMP(pattern, text, res);

if (res.empty()) {cout << "-1"<<endl; return 0;}

for(auto iter = res.begin(); iter != res.end(); iter++){

if(iter != res.begin()) cout <<",";

cout<<\*iter;

}

cout << endl;

}else if (choose == 2){

cout << kmp\_cycle(pattern, text) << '\n';

}

}else if (consoleOrFile == 2){

string file;

cout << "Input filename(.txt): ";

cin >> file;

ifstream fin(file + ".txt");

if(!fin) {cout << "Can't open this file!";return 0;}

getline(fin, pattern);

getline(fin, text);

int choose = 0;

cout << "1 -- index\n2 -- cycle\n";

cin >> choose;

if (choose == 1) {

KMP(pattern, text, res);

if (res.empty()) {cout << "-1"<<endl; return 0;}

for(auto iter = res.begin(); iter != res.end(); iter++){

if(iter != res.begin()) cout <<",";

cout<<\*iter;

}

cout << endl;

}else if (choose == 2){

cout << kmp\_cycle(pattern, text) << '\n';

}

}

}

# Приложение Б Тестирование

Таблица 1 - Примеры тестовых случаев для нахождения индексов.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | defabc  abcdef | -1 | Программа работает корректно |
|  | ab  abab | 0,2 | Программа работает корректно |

Таблица 2 - Примеры тестовых случаев для нахождения циклического сдвига.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | defabc  abcdef | 3 | Программа работает корректно |
|  | ab  abab | -1 | Программа работает корректно |