

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. 8382

Чирков С.А.

Преподаватель

Фирсов М.А.

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.

Задание 1.

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 .

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.

Sample Input:

ab

abab

Sample Output:

0, 2

Задание 2.

Заданы две строки A ($|A| \leq 5000000$) и B ($|B| \leq 5000000$). Определить, является ли A циклическим сдвигом B (это значит, что A и B имеют одинаковую длину и AA состоит из суффикса BB , склеенного с префиксом BB). Например, defabc является циклическим сдвигом abcdef.

Входные данные:

Первая строка – A

Вторая строка – B

Выходные данные:

Если A является циклическим сдвигом B , индекс начала строки B в A , иначе вывести -1 . Если возможно несколько сдвигов вывести первый индекс.

Sample Input:

defabc

abcdef

Sample Output:

3

Вариант дополнительного задания.

Вар. 1. Подготовка к распараллеливанию: работа по поиску разделяется на k равных частей, пригодных для обработки k потоками (при этом длина образца гораздо меньше длины строки поиска).

Описание алгоритма

Алгоритм Кнута — Морриса — Пратта (КМП-алгоритм) — эффективный алгоритм, осуществляющий поиск подстроки в строке. Время работы алгоритма линейно зависит от объёма входных данных, то есть разработать асимптотически более эффективный алгоритм невозможно. На вход алгоритма передается строка-образ, вхождения которой нужно найти, и строка-текст, в которой нужно найти вхождения. Алгоритм сначала вычисляет префикс-функцию строки-образа, с учетом запрашиваемого количества потоков. Далее сравнивается элемент строки-текста и элемент строки-образа. В случае их равенства, происходит увеличение индексов, указывающих на символ в строке-тексте и строке-образе. Затем после того как выявилось совпадение символов, происходит проверка равенства обрабатываемого индекса и длины строки-образа, если это верно, то значит, что вхождение найдено и происходит запись индекса начала вхождения в массив с ответами ans.

В случае, когда элемент строки-текста и элемент строки-образа не совпали, то происходит проверка, не равен ли сейчас нулю индекс, указывающий на текущий элемент строки-образа. Если это верно, увеличиваем на единицу индекс, который указывает на символ в строке-тексте. Иначе, если индекс не равен 0, то происходит перемещение позиции индекса при помощи префикс-функции. Алгоритм завершает работу по окончании строки-текста.

Описание функций и структур данных.

vector <int> ans – динамический массив целых чисел для хранения ответа.

vector <int> pp – динамический массив целых чисел для хранения префикс-функции.

Используются встроенные функции языка.

Тестирование

Ввод	Вывод
Вхождение подстроки	
ab abab 1	0,2
abc abcsabcsacs 3	0,3,6
aba ababababa 2	0,2,4,6
aba abbba 5	слишком много потоков
Циклический сдвиг	
aba baa 2	2
abc accc 2	длины строк не равны
ab ba 3	слишком много потоков
abc def 2	-1

Тестирование с промежуточными выводами

Вхождение подстроки

Test input:

```
abababa  
ababababababa
```

Test output:

```
текущий поток 1  
найдено новое значение  $p(2) = 1$   
 $0, 0, 1$   
  
текущий поток 2  
найдено новое значение  $p(3) = 2$   
найдено новое значение  $p(4) = 3$   
найдено новое значение  $p(5) = 4$   
найдено новое значение  $p(6) = 5$   
  
префикс-функция создана:  
 $0, 0, 1, 2, 3, 4, 5$   
  
текущий поток 1  
  
текущий поток 2  
найдено новое решение 0  
найдено новое решение 2  
найдено новое решение 4  
найдено новое решение 6  
  
ответ  
 $0, 2, 4, 6$ 
```

Циклический сдвиг

```
Test input:
abcbabc
bcbabca

Test output:

текущий поток 1
0, 0, 0

текущий поток 2
найдено новое значение  $p(4) = 1$ 
найдено новое значение  $p(5) = 2$ 
найдено новое значение  $p(6) = 3$ 

префикс-функция создана:
0, 0, 0, 0, 1, 2, 3

текущий поток 1
текущий поток 2
найдено новое решение 6

ответ
6
```

Сложность алгоритма

Сложность алгоритма по времени и по памяти: $O(m + n)$, m – длина образа, n – длина текста.

Выводы.

В ходе работы был построен и анализирован алгоритм КМП на примере программ, решающих следующие задачи: нахождение индексов вхождения образца в строке и индекс циклического смещения одной строки в другой.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД.

Вхождение образца в строку

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <string>
#include <fstream>
#include <map>
#include <set>
#include <math.h>
using namespace std;

int main()
{
    string p, t;
    int i,j,fl=0;
    vector <int> pp;
    vector <int> ans;
    cout << "введите первую строку" << endl;
    getline(cin,p);
    cout << "введите вторую строку" << endl;
    getline(cin,t);
    int thread;
    cout << "введите количество потоков" << endl;
    cin >> thread;
    int n=p.length();
    int m=t.length();
    pp.push_back(0);
    i=1;
    j=0;
    if(thread>n)
    {
        cout<<"слишком много потоков"<<endl;
        return 0;
    }
    for(int c=0;c<thread;c++)
    {
        cout<<"текущий поток "<<c+1<<endl;
        while(i!=(c+1)*round(n/thread))
        {
            if(p[i]==p[j])
            {
                pp.push_back(j+1);
                cout<<"найдено новое значение p("<<i<<" = "<<j+1<<endl;
                j++;
                i++;
            }
            else
            {
                if(j==0)
                {
                    pp.push_back(0);
                    i++;
                }
                else
                {
                    j=pp[j-1];
                }
            }
        }
    }
}
```



```

    }
}
}
if(c!=thread-1){
cout<<endl<<pp[0];
for(int l=1;l<pp.size();l++)
{
    cout<<" "<<pp[l];
}
cout<<endl<<endl;
}
}
while(p[i]!='\0')
{
    if(p[i]==p[j])
    {
        pp.push_back(j+1);
        cout<<"найденно новое значение p("<<i<<" ) = "<< j+1<<endl;
        j++;
        i++;
    }
    else
    {
        if(j==0)
        {
            pp.push_back(0);
            i++;
        }
        else
        {
            j=pp[j-1];
        }
    }
}
cout<<endl<<"префикс-функция создана:"<<endl<<endl<<pp[0];
for(int l=1;l<n;l++)
{
    cout<<" "<<pp[l];
}
cout<<endl;
i=0;
j=0;
for(int c=0;c<thread;c++)
{
    cout<<endl<<"текущий поток "<<c+1<<endl;
    while(i!=(c+1)*round(m/thread))
    {
        if(p[j]==t[i])
        {
            if(pp.size()-1==j)
            {
                ans.push_back(i-j);
                cout<<"найденно новое решение "<<i-j<<endl;
                fl=1;
                if(j!=0)
                    j=pp[j-1];
            }
            else
                i++;
        }
    }
}

```

```

        else
        {
            i++;
            if(p[j+1]!='\0')
                j++;
        }
    }
    else
    {
        if(j!=0)
        {
            j=pp[j-1];
        }
        else
        {
            i++;
        }
    }
}
}
while(t[i]!='\0')
{
    if(p[j]==t[i])
    {
        if(pp.size()-1==j)
        {
            ans.push_back(i-j);
            cout<<"найденно новое решение "<<i-j<<endl;
            fl=1;
            if(j!=0)
                j=pp[j-1];
            else
                i++;
        }
        else
        {
            i++;
            if(p[j+1]!='\0')
                j++;
        }
    }
    else
    {
        if(j!=0)
        {
            j=pp[j-1];
        }
        else
        {
            i++;
        }
    }
}
if(fl==0)
{
    cout<<"-1";
    return 0;
}

```

```

    }
    cout<<endl<<"ответ"<<endl<<ans[0];
    for(int u=1;u<ans.size();u++)
    {
        cout<<","<<ans[u];
    }
    return 0;
}

```

Циклический сдвиг

```

#include <iostream>
#include <vector>
#include <algorithm>
#include <string>
#include <fstream>
#include <map>
#include <set>
#include <math.h>
using namespace std;

int main()
{
    string p, t;
    int i,j,fl=0;
    vector <int> pp;
    vector <int> ans;
    cout << "введите первую строку" << endl;
    getline(cin,p);
    cout << "введите вторую строку" << endl;
    getline(cin,t);
    int thread;
    cout << "введите количество потоков" << endl;
    cin >> thread;
    int n=p.length();
    int m=t.length();
    t=t+t;
    pp.push_back(0);
    i=1;
    j=0;
    if(thread>n)
    {
        cout<<"слишком много потоков"<<endl;
        return 0;
    }
    for(int c=0;c<thread;c++)
    {
        cout<<"текущий поток "<<c+1<<endl;
        while(i!=(c+1)*round(n/thread))
        {
            if(p[i]==p[j])
            {
                pp.push_back(j+1);
                cout<<"найденно новое значение p("<<i<<" = "<<j+1<<endl;
                j++;
                i++;
            }
        }
    }
}

```

```

else
{
    if(j==0)
    {
        pp.push_back(0);
        i++;
    }
    else
    {
        j=pp[j-1];
    }
}
}
if(c!=thread-1){
cout<<endl<<pp[0];
for(int l=1;l<pp.size();l++)
{
    cout<<","<<pp[l];
}
cout<<endl<<endl;
}
}
while(p[i]!='\0')
{
    if(p[i]==p[j])
    {
        pp.push_back(j+1);
        cout<<"найдено новое значение p("<<i<<") = "<< j+1<<endl;
        j++;
        i++;
    }
    else
    {
        if(j==0)
        {
            pp.push_back(0);
            i++;
        }
        else
        {
            j=pp[j-1];
        }
    }
}
cout<<endl<<"префикс-функция создана:"<<endl<<endl<<pp[0];
for(int l=1;l<n;l++)
{
    cout<<","<<pp[l];
}
cout<<endl;
i=0;
j=0;
for(int c=0;c<thread;c++)
{
    cout<<endl<<"текущий поток "<<c+1<<endl;
    while(i!=(c+1)*round(m/thread))
    {
        if(p[j]==t[i])
        {

```

```

        if(pp.size()-1==j)
        {
            ans.push_back(i-j);
            cout<<"найденo новое решение "<<i-j<<endl;
            fl=1;
            if(j!=0)
                j=pp[j-1];
            else
                i++;
        }
        else
        {
            i++;
            if(p[j+1]!='\0')
                j++;
        }
    }
    else
    {
        if(j!=0)
        {
            j=pp[j-1];
        }
        else
        {
            i++;
        }
    }
}
}
while(t[i]!='\0')
{
    if(p[j]==t[i])
    {
        if(pp.size()-1==j)
        {
            ans.push_back(i-j);
            cout<<"найденo новое решение "<<i-j<<endl;
            fl=1;
            if(j!=0)
                j=pp[j-1];
            else
                i++;
        }
        else
        {
            i++;
            if(p[j+1]!='\0')
                j++;
        }
    }
    else
    {
        if(j!=0)
        {
            j=pp[j-1];
        }
        else
    }
}

```

```

        {
            i++;
        }
    }

}

if(fl==0 || n!=m)
{
    cout<<"-1";
    return 0;
}
cout<<endl<<"ответ"<<endl<<ans[0];
for(int u=1;u<ans.size();u++)
{
    cout<<","<<ans[u];
}
return 0;
}

```