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|≤5000000) и B|B|≤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.

Префикс-функция - это такая функция от строки, которая для каждого элемента строки с номером i показывает наибольшую длину собственного суффикса подстроки с 0 до i элемента включительно, совпадающий с ее префиксом. Префикс — это подстрока начинающиеся с начала строки. Суффикс — это подстрока, заканчивающиеся в конце строки. Собственный — это значит не совпадающий со всей строкой. Для её вычисления инициализируются два индекса элементов j = 0, i = 1. Пока алгоритм не дойдет до конца строки по индексу i, происходит сравнение элементов с индексами i и j, если они равны – префикс-функция элемента i равна j+1 и оба индекса инкрементируются, в противном случае если j указывает на первый элемент строки – префикс-функция элемента i равна 0 и инкрементируется i, если j не указывает на начало строки – j переходит к элементу с индексом, равным значению префикс-функции предыдущего элемента.

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 abcabcabcac 3	0,3,6
aba ababababa 2	0,2,4,6
aba abbba 5	слишком много потоков
Циклический сдвиг	
aba baa 2	1

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:

```
ababa
abababababa
```

Test output:

```
построение префикс-функции ababa
текущий поток 1
текущие p(i) = p(1) = b, p(j) = p(0) = a
текущие p(i) = p(2) = a, p(j) = p(0) = a
найдено новое значение префикс-функции pp(2) = 1,так как p(i) = p(j)
текущие p(i) = p(3) = b, p(j) = p(1) = b
найдено новое значение префикс-функции pp(3) = 2,так как p(i) = p(j)
текущие p(i) = p(4) = a, p(j) = p(2) = a
найдено новое значение префикс-функции pp(4) = 3,так как p(i) = p(j)
префикс-функция создана:
0,0,1,2,3
алгоритм кмп
текущий поток 1
текущие p(j) = p(0) = a, t(i) = t(0) = a
текущие p(j) = p(1) = b, t(i) = t(1) = b
текущие p(j) = p(2) = a, t(i) = t(2) = a
текущие p(j) = p(3) = b, t(i) = t(3) = b
текущие p(j) = p(4) = a, t(i) = t(4) = a
найдено новое решение 0, так как дошли до конца шаблона
индекс j перешёл в pp[j-1] = 2
текущие p(j) = p(2) = a, t(i) = t(4) = a
текущие p(j) = p(3) = b, t(i) = t(5) = b
текущие p(j) = p(4) = a, t(i) = t(6) = a
найдено новое решение 2, так как дошли до конца шаблона
индекс j перешёл в pp[j-1] = 2
текущие p(j) = p(2) = a, t(i) = t(6) = a
текущие p(j) = p(3) = b, t(i) = t(7) = b
текущие p(j) = p(4) = a, t(i) = t(8) = a
```

```
текущие p(j) = p(3) = b, t(i) = t(9) =
индекс j перешёл в pp[j-1] = 1
текущие p(j) = p(1) = b, t(i) = t(9) =
индекс j перешёл в pp[j-1] = 0
текущие p(j) = p(0) = a, t(i) = t(9) =
текущие p(j) = p(0) = a, t(i) = t(10) =
текущие p(j) = p(0) = a, t(i) = t(11) =
текущие p(j) = p(0) = a, t(i) = t(12) =
текущие p(j) = p(0) = a, t(i) = t(13) =
текущие p(j) = p(0) = a, t(i) = t(14) =
текущие p(j) = p(0) = a, t(i) = t(15) =
текущие p(j) = p(0) = a, t(i) = t(16) =
текущие p(j) = p(0) = a, t(i) = t(17) =
0,2,4
```

Циклический сдвиг

Test input:

```
abcab
bcaba
1
```

Test output:

```
построение префикс-функции bcaba
текущий поток 1
текущие p(i) = p(1) = c, p(j) = p(0) = b
текущие p(i) = p(2) = a, p(j) = p(0) = b
текущие p(i) = p(3) = b, p(j) = p(0) = b
найдено новое значение префикс-функции pp(3) = 1,так как p(i) = p(j)
текущие p(i) = p(4) = a, p(j) = p(1) = c
индекс j перешёл в pp[j-1] = 0
текущие p(i) = p(4) = a, p(j) = p(0) = b
префикс-функция создана:
0,0,0,1,0
алгоритм кмп
текущий поток 1
текущие p(j) = p(0) = b, t(i) = t(0) = a
текущие p(j) = p(0) = b, t(i) = t(1) = b
текущие p(j) = p(1) = c, t(i) = t(2) = c
текущие p(j) = p(2) = a, t(i) = t(3) = a
текущие p(j) = p(3) = b, t(i) = t(4) = b
текущие p(j) = p(4) = a, t(i) = t(5) = a
найдено новое решение 1, так как дошли до конца шаблона
индекс j перешёл в pp[j-1] = 1
текущие p(j) = p(1) = c, t(i) = t(5) = a
индекс j перешёл в pp[j-1] = 0
текущие p(j) = p(0) = b, t(i) = t(5) = a
текущие p(j) = p(0) = b, t(i) = t(6) = b
текущие p(j) = p(1) = c, t(i) = t(7) = c
текущие p(j) = p(2) = a, t(i) = t(8) = a
текущие p(j) = p(3) = b, t(i) = t(9) = b
1
```

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;
   }
   cout<<"построение префикс-функции "<<p<<endl<<endl;
   for(int c=0;c<thread;c++) //цикл по шаблону, разделенный на thread потоков
   {
      cout<<"текущий поток "<<c+1<<endl<<endl;
      while(i!=(c+1)*round(n/thread)) //инициализация префикс-функции
      {
         cout<<"текущие p(i) = p("<<i<<") = "<< p[i] <<", p(j) = p("<<j<<") = "<< p[j] <<endl<<endl;
         if(p[i]==p[j])
         {
            pp.push_back(j+1);
            cout<<"найдено новое значение префикс-функции pp("<<i<<") = "<< j+1<<",так как p(i) =
p(j)"<<endl<<endl;
            j++;
            i++;
         }
         else
            {
            if(j==0)
            {
               pp.push_back(0);
               i++;
            }
```

```
            else
            {
               j=pp[j-1];
               cout<<"индекс j перешёл в pp[j-1] = "<<pp[j-1]<<endl<<endl;
            }
         }
      }
      if(c!=thread-1){
      cout<<pp[0];
      for(int l=1;l<pp.size();l++)
      {
         cout<<","<<pp[l];
      }
      cout<<endl<<endl;
      }
   }
   while(p[i]!='\0') //обработка оставшейся части строки (если не делится на равные части)
   {
      if(p[i]==p[j])
      {
         pp.push_back(j+1);
         cout<<"найдено новое значение pp("<<i<<") = "<< j+1<<", так как p(i) = p(j)"<<endl<<endl;
         j++;
         i++;
      }
      else
      {
         if(j==0)
         {
            pp.push_back(0);
            i++;
         }
         else
         {
            j=pp[j-1];
         }
      }
   }
   cout<<"префикс-функция создана:"<<endl<<endl<<pp[0];
   for(int l=1;l<n;l++)
   {
      cout<<","<<pp[l];
   }
   cout<<endl<<endl;

   cout<<"алгоритм кмп"<<endl<<endl;
   i=0;
   j=0;
   for(int c=0;c<thread;c++) //цикл по тексту, разделенный на thread потоков
   {
      cout<<"текущий поток "<<c+1<<endl<<endl;
      while(i!=(c+1)*round((m*2)/thread)) //идем по round(len(b)/thread) символов каждый раз
      {
         cout<<"текущие p(j) = p("<<j<<") = "<< p[j] <<", t(i) = t("<<i<<") = "<< t[i] <<endl<<endl;
         if(p[j]==t[i]) //совпал ли символ шаблон?
         {
            if(pp.size()-1==j)
            {
               ans.push_back(i-j); //сохраняем, если шаблон пройден
```

```
            cout<<"найдено новое решение "<<i-j<<", так как дошли до конца шаблона"<<endl<<endl;
            fl=1;
            if(j!=0)
            {
               cout<<"индекс j перешёл в pp[j-1] = "<<pp[j-1]<<endl<<endl;
               j=pp[j-1];
            }
            else
               i++;
         }
         else
         {
            i++;
            if(p[j+1]!='\0')
               j++;
         }
      }
      else
      {
         if(j!=0) //иначе - переход по префикс-функции
         {
            cout<<"индекс j перешёл в pp[j-1] = "<<pp[j-1]<<endl<<endl;
            j=pp[j-1];
         }
         else //если в начале шаблона - следующий символ
         {
            i++;
         }
      }

   }
}
while(t[i]!='\0') //обработка оставшейся части строки (если не делится на равные части)
{
   if(p[j]==t[i])
   {
   if(pp.size()-1==j)
   {
      ans.push_back(i-j);
      cout<<"найдено новое решение "<<i-j<<endl<<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<<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,t);
   cout << "введите вторую строку" << endl;
   getline(cin,p);
   int thread;
   cout << "введите количество потоков" << endl;
   cin >> thread;
   int n=p.length();
   int m=t.length();
   t=t+t; // циклический сдвиг проверяется при помощи строки, которая конкатенировна с собой же
   pp.push_back(0);
   i=1;
   j=0;
   if(thread>n)
   {
      cout<<"слишком много потоков"<<endl;
      return 0;
   }
   cout<<"построение префикс-функции "<<p<<endl<<endl;
   for(int c=0;c<thread;c++) //цикл по шаблону, разделенный на thread потоков
```

```
        {
          cout<<"текущий поток "<<c+1<<endl<<endl;
          while(i!=(c+1)*round(n/thread))  //инициализация префикс-функции
          {
            cout<<"текущие p(i) = p("<<i<<") = "<< p[i] <<", p(j) = p("<<j<<") = "<< p[j] <<endl<<endl;
            if(p[i]==p[j])
            {
              pp.push_back(j+1);
              cout<<"найдено новое значение префикс-функции pp("<<i<<") = "<< j+1<<",так как p(i) =
p(j)"<<endl<<endl;
              j++;
              i++;
            }
            else
              {
              if(j==0)
              {
                pp.push_back(0);
                i++;
              }
              else
              {
                j=pp[j-1];
                cout<<"индекс j перешёл в pp[j-1] = "<<pp[j-1]<<endl<<endl;
              }
            }
          }
          if(c!=thread-1){
          cout<<pp[0];
          for(int l=1;l<pp.size();l++)
          {
            cout<<","<<pp[l];
          }
          cout<<endl<<endl;
          }
        }
        while(p[i]!='\0') //обработка оставшейся части строки (если не делится на равные части)
        {
          if(p[i]==p[j])
          {
            pp.push_back(j+1);
            cout<<"найдено новое значение pp("<<i<<") = "<< j+1<<", так как p(i) = p(j)"<<endl<<endl;
            j++;
            i++;
          }
          else
          {
            if(j==0)
            {
              pp.push_back(0);
              i++;
            }
            else
            {
              j=pp[j-1];
            }
          }
        }
        cout<<"префикс-функция создана:"<<endl<<endl<<pp[0];
```

```
for(int l=1;l<n;l++)
{
   cout<<","<<pp[l];
}
cout<<endl<<endl;

cout<<"алгоритм кмп"<<endl<<endl;
i=0;
j=0;
for(int c=0;c<thread;c++) //цикл по тексту, разделенный на thread потоков
{
   cout<<"текущий поток "<<c+1<<endl<<endl;
   while(i!=(c+1)*round((m*2)/thread)) //идем по round(len(b)/thread) символов каждый раз
   {
      cout<<"текущие p(j) = p("<<j<<") = "<< p[j] <<", t(i) = t("<<i<<") = "<< t[i] <<endl<<endl;
      if(p[j]==t[i]) //совпал ли символ шаблон?
      {
         if(pp.size()-1==j)
         {
            ans.push_back(i-j); //сохраняем, если шаблон пройден
            cout<<"найдено новое решение "<<i-j<<", так как дошли до конца шаблона"<<endl<<endl;
            fl=1;
            if(j!=0)
            {
               cout<<"индекс j перешёл в pp[j-1] = "<<pp[j-1]<<endl<<endl;
               j=pp[j-1];
            }
            else
               i++;
         }
         else
         {
            i++;
            if(p[j+1]!='\0')
               j++;
         }
      }
      else
      {
         if(j!=0)  //иначе - переход по префикс-функции
         {
            cout<<"индекс j перешёл в pp[j-1] = "<<pp[j-1]<<endl<<endl;
            j=pp[j-1];
         }
         else //если в начале шаблона - следующий символ
         {
            i++;
         }
      }

   }
}
while(t[i]!='\0') //обработка оставшейся части строки (если не делится на равные части)
{
   if(p[j]==t[i])
   {
   if(pp.size()-1==j)
   {
      ans.push_back(i-j);
```

```
            cout<<"найдено новое решение "<<i-j<<endl<<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<<ans[0];
      return 0;
}
```