

**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

\_\_\_\_\_

Щеглов А.С.

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.**

Реализуйте алгоритм КМП и с его помощью для заданных шаблона  $P(|| \leq 15000)$  и текста  $T(|| \leq 5000000)$  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.

#### **Пример входных данных**

aba

ababa

#### **Пример выходных данных**

0, 2

### **Вариант дополнительного задания.**

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.

### **Описание структуры данных.**

В данной программе используются векторы.

Вектор `vector<char> pp` – данный контейнер является аналогом массива  $\pi[i]$ , он хранит в себе максимальную длину совпадающих префикса и суффикса подстроки в образе, которая заканчивается  $i$ -м символом.

Вектор `vector<int> ans` – это вектор для записи в него ответа.

### **Описание функций**

Функций, кроме `main()`, нет.

### **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.

### **Ход работы.**

На вход алгоритма передается строка-образ, вхождения которой нужно найти, и строка-текст, в которой нужно найти вхождения. Оптимизация – строка-текст считывается посимвольно, в памяти хранится текущий символ. Алгоритм сначала вычисляет префикс-функцию строки-образа. Далее посимвольно считывается строка-текст.

изначально сравнивается рассматриваемый

текущий символ(`char x`) строки-текста и текущий  $l$ -й элемент строки-образа.

В случае их равенства, происходит сдвиг  $l$ -го элемента строки-образа и также увеличивается переменная(`int i`), которая олицетворяет индекс, указывающий на символ в строке-тексте. Затем после того как выявилось совпадение символов, происходит проверка на равенство  $j$  номера строки образа и длины строки образа, если это верно, то значит, что вхождение найдено и происходит запись индекса начала вхождения вектор ответа(`vector<int> ans`).

В случае, когда текущий символ(`char x`) строки-текста и текущий  $j$ -й элемент строки-образа не равны, то происходит проверка, не находится ли сейчас в

начале(в нуле) индекс, указывающий на текущий элемент строки-образа. Если это верно, увеличиваем на единицу индекс, который указывает на символ в строке текста. Иначе, если индекс не равен 0, то происходит перемещение позиции индекса 1 из одной позиции в другую. Алгоритм завершает работу по окончании строки-текста.

### Тестирование.

```
ab
abab
0,2
```

```
a
aaaaaaaaa
0,1,2,3,4,5,6,7
```

```
ab
ababababababasdvasdbasdbabababab
0,2,4,6,8,22,27
```

```
eeeeeee
fffffffffff
-1
```

### Тестирование с промежуточным выводом

Промежуточный вывод – это вывод получившейся префикс функции.

```
aaaaaaa
The Prefix function is: 0123456
aaaaaaaaaaaaaaaaaaaaaaaaaaaaa
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18
Для продолжения нажмите любую клавишу . . . _
```

```
ababa
The Prefix function is: 00123
ababababababababfbasdasbaabaabababab
0,2,4,6,8,10,28,30
Для продолжения нажмите любую клавишу . . .
```

```
abbadsfab
The Prefix function is: 000100012
abbadsfabfasdfwasadsaabbadsfabdfsfe
0.22
Для продолжения нажмите любую клавишу . . . _
```

### **Сложность алгоритма.**

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

Сложность алгоритма по памяти:  $O(m)$ ,  $m$  – длина образа, так как программа хранит только строку-образ, которая считывается в самом начале.

### **Выводы.**

В ходе работы был построен и анализирован алгоритм КМП. Код программы представлен в приложении А.

## ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД.

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

int main()
{
    string p;
    int i,j,fl=0;
    vector<int> pp;//Prefix-function for P
    vector<int> ans;//Vector for answers
    getline(cin,p);//Input line P
    pp.push_back(0);
    i=1;
    j=0;
    while(p[i]!='\0')//start deploying values into Prefix-function
    {
        if(p[i]==p[j])
        {
            pp.push_back(j+1);

            j++;
            i++;
        }
        else
        {
            if(j==0)
            {
                pp.push_back(0);
                i++;
            }
            else
            {
                j=pp[j-1];
            }
        }
    }

    cout<<"The Prefix function is: ";
    for(int u=0;u<pp.size();u++)
    {
        cout<<pp[u];
    }
    cout<<endl;
    i=0;
    j=0;
    char x=getchar();
    while(x!='\n')//Starting go on T , to find entry
    {
        if(p[j]==x)
        {
            if(pp.size()-1==j)//if whole string P was found in T
            {
```

```

        ans.push_back(i-j);
        fl=1;
        if(j!=0)//To not (Seg fault)
            j=pp[j-1];
        else
        {
            i++;
            x=getchar();
        }
    }
    else
    {
        x=getchar();
        i++;
        if(p[j+1]!='\0')
            j++;
    }
}
else
{
    if(j!=0)
    {
        j=pp[j-1]; //Back to nearest entry
    }
    else
    {
        i++; //Step in
        x=getchar();
    }
}

}
if(fl==0)
{
    cout<<"-1";
    return 0;
}
cout<<ans[0];
for(int u=1;u<ans.size();u++)
{
    cout<<","<<ans[u];
}
cout<<endl;
//cout<<" String P = "<<p<<endl;
//      cout<<" String T = "<<t<<endl;
system("pause");
return 0;
}

```