

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ
по лабораторной работе №4
по дисциплине «Построение и анализ алгоритмов»
Тема: Алгоритм Кнута-Морриса-Пратта

Студентка гр. 9382

Круглова В.Д.

Преподаватель

Фирсов М.А.

Санкт-Петербург

2020

Цель работы.

Ознакомится с алгоритмом Кнута-Морриса-Пратта для поиска подстроки в строке.

Постановка задачи.

Задача 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 .

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

Sample Input:

ab
abab

Sample Output:

0, 2

Задача 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 A состоит из суффикса B , склеенного с префиксом B). Например, defabc является циклическим сдвигом abcdef.

Вход:

Первая строка - A

Вторая строка - B

Выход:

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

Алгоритм решения.

Реализована префикс-функция. Для нулевого символа строки значение префикс-функции устанавливается 0. Для каждого i -го символа, кроме нулевого, берется значение $k = \text{prefix}[i-1]$. Проверяется, совпадают ли символы на позициях i и k . Если они совпадают, то для i -го элемента устанавливается значение $k + 1$. Если же они не совпадают, то k устанавливается как значение префикс-функции для $k - 1$ -ого элемента. Для обновленного значения k проверка повторяется. Это продолжается до тех пор, пока либо не будут найдены одинаковые символы на позициях i и k , либо k не станет равным 0. В последнем случае на i -ую позицию префикс-функции будет установлена либо 1, если i -ый символ совпадает с первым, либо 0, если не совпадает.

Для поиска шаблона в строке вычисляется префикс-функция шаблона. Затем для строки, в которой осуществляется поиск шаблона происходит процедура, похожая на вычисление префикс-функции, но с некоторыми отличиями: суффикс берется в строке поиска, а префикс – в строке шаблона. Для строки поиска не имеет смысла хранить значения префикс-функции,

поэтому запоминается только последнее вычисленное. Если на каком-то символе значение префикс-функции совпадет с длиной шаблона, значит, найден суффикс, совпадающий с шаблоном. Выводится индекс его начала.

Для определения сдвига используется похожий алгоритм. Для второй строки вычисляется префикс-функция. Алгоритм дважды проходит по первой строке с заикливанием. Если в какой-то момент текущее значение максимальной длины суффикса совпадет с длиной первой строки, значит, вторая строка является сдвигом первой. Выводится индекс начала вхождения второй строки в первой.

Пусть m – длина строки, для которой вычисляется префикс-функция, n – длина второй строки. В вычислении префикс-функции используются два цикла: внешний цикл `for` длиной m и внутренний цикл. Во внешнем цикле значение k может увеличиться максимум на 1 за итерацию (а может и не увеличиваться). Во внутреннем цикле оно уменьшается как минимум на 1 за итерацию, но не может опуститься ниже 0. То есть, каждый заход во внутренний цикл накладывает ограничение на максимальную длину внутренних циклов в последующих итерациях. Очевидно, что количество уменьшений во внутреннем цикле не может превышать количества увеличений во внешнем, поэтому общее число заходов во внутренний цикл не больше m . Сложность вычисления префиксфункции $O(2m)$. При $m \rightarrow \inf$, сложность $O(m)$.

Сложность поиска шаблона оценивается так же, как и префикс-функция, но внешний цикл проходит по другой строке длины n . Поэтому общая сложность алгоритма для первого задания $O(2m + 2n)$ При $m, n \rightarrow \inf$, сложность $O(m+n)$.

Сложность проверки сдвига равна $O(2m + 4n)$, так как внешний цикл там вдвое больше. При $m, n \rightarrow \inf$, сложность $O(m+n)$.

Хранение изначальных строк занимает $O(m + n)$ памяти. Для хранения префикс-функции шаблона дополнительно используется $O(m)$ памяти.

Структуры данных.

class FullString -класс хранит текст, шаблон и значения префиксной функции `std::string` используется для хранения введенных строк. `std::vector<int>` используется для хранения значений префикс-функции.

Функции.

`std::vector<int> FullString::KMP_search()` – вычисление префикс-функции для строки. Возвращает вектор результата.

`void FullString::isCyclShift()` – проверяет, является ли pattern циклическим сдвигом text. Если да, выводит индекс начала pattern в text, иначе -1.

`void FullString::fill_pf()` – заполняет массив значений префиксной функции.

`FullString::FullString()` – конструктор класса FullString. Считывает шаблон и текст, а также вызывает функцию `fill_pf`.

Тестирование.

```
qwe
rtyqweqweasdqwe
***KMP Search***
Current symbol: r
Current match:
Current symbol: t
Current match:
Current symbol: y
Current match:
Current symbol: q
Current match: q
Current symbol: w
Current match: qw
Current symbol: e
Current match: qwe
Current symbol: q
Current match: q
Current symbol: w
Current match: qw
Current symbol: e
Current match: qwe
Current symbol: a
Current match:
Current symbol: s
Current match:
Current symbol: d
Current match:
Current symbol: q
Current match: q
Current match:
Current symbol: q
Current match: q
Current symbol: w
Current match: qw
Current symbol: e
Current match: qwe
3, 6, 12
***Cycle Shift Check***
-1
```

Рис. 2. Тестирование КМП

```
qwerty
rtyqwe
***KMP Search***
Current symbol: r
Current match:
Current symbol: t
Current match:
Current symbol: y
Current match:
Current symbol: q
Current match: q
Current symbol: w
Current match: qw
Current symbol: e
Current match: qwe
-1

***Cycle Shift Check***

Current symbol: r
Current match:
Current symbol: t
Current match:
Current symbol: y
Current match:
Current symbol: q
Current match: q
Current symbol: w
Current match: qw
Current symbol: e
Current match: qwe
Current symbol: r
Current match: qwer
Current symbol: t
Current match: qwerty
3
```

Рис. 1. Тестирование функции поиска циклического переноса

Выводы.

В ходе работы был реализован алгоритм Кнута-Морриса-Пратта для поиска подстрок в строке, а также его модификация для проверки циклического сдвига двух строк.

ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <iostream>
#include <string>
#include <vector>

using
std::vector;
using
std::string;
using std::cout;
using std::cin;
using std::endl;

class FullString
//класс хранит текст, шаблон и значения префиксной функции
{
private:
    vector<int> pf; //массив значений префикс функции
    string
pattern;
string S;
public:
    FullString();
vector<int>
KMP_search();    void
isCyclShift();   void
fill_pf();
};

FullString::FullString()
{
    getline(cin, pattern);
getline(cin, S);
pf.resize(pattern.length());
    fill_pf();
}
```

```

void FullString::fill_pf()
//заполняем массив значений префиксной функции
{
    pf[0]
] = 0;

    for (int k = 0, i = 1; i < pattern.length(); ++i)
        //заполняем pf
        {
            while (k > 0 && pattern[i] != pattern[k])
                k = pf[k-1];

            if (pattern[i] == pattern[k])
                k++;
            pf[i]
= k;
        }
}

vector<int> FullString::KMP_search()
{
    vector<int> res; // массив результата

    for (int k = 0, i = 0; i < S.length(); i++)
        {
            // Если предыдущий суффикс нельзя расширить, нужно попытаться взять суффикс
            // меньшего размера
            while (k > 0 && pattern[k]
!= S[i])
                k = pf[k-1];

            // Если символы справа от префикса и суффикса совпадают, суффикс расширяется
            if (pattern[k] ==
S[i])
                k++;

            //Найдено вхождение

```

```

        if (k == pattern.length()) {
res.push_back(i - k + 1);
        }

```

```

        cout << "Current symbol: " <<
S[i] << endl;        cout << "Current
match: ";        for (int l=0; l < k; l+
+)        cout << pattern[l];
cout << endl;
    }

```

```

    if (res.empty())
{        res.push_back(-1);
    }

```

```

    return res;
}

```

```

void FullString::isCyclShift()
{
    //Если длины не равны,
сразу откат    if
(pattern.length() != S.length())
    {
        cout << -1;
return;
    }

```

```

    for (int k = 0, i = 0; i < pattern.length()*2; i++)
    {
        //j - циклический обход
строки        int j = i %
pattern.length();

```

```

        // Если предыдущий суффикс нельзя расширить, нужно попытаться взять суффикс
меньшего размера        while (k > 0 && S[k] != pattern[j])        k = pf[k-1];

```

```

        // Если символы справа от префикса и суффикса совпадают, суффикс расширяется
        if (S[k] ==
pattern[j])        k++;

        //Найдено вхождение
        if (k ==
pattern.length())    {
cout << pattern[k-1];
cout << endl;
        cout << i - k + 1 << endl;
        return;
    }
        cout << endl << "Current symbol: "
<< S[j] << endl;        cout << "Current
match: ";        for (int l=0; l < k; l++)
cout << pattern[l];
    }
    cout << -1 << endl;
}

int main()
{
    FullString str;
    cout << "***KMP Search***" << endl;
    std::vector<int> res = str.KMP_search();

    cout << res[0];    for (int i = 1; i <
res.size(); i++)        cout << ", " <<
res[i];    cout << endl << "***Cycle Shift
Check***" << endl;
    str.isCyclShift();

    return 0;
}

```