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. 1384 _____ Феопентов А.Ю

Преподаватель _____ Шевелева А.М.

Санкт-Петербург
2023

Цель работы.

Изучить и реализовать алгоритм КМП. Реализовать поиск всех вхождений строки в подстроку и возможность циклического сдвига.

Задание.

Задача 1.

Реализуйте алгоритм КМП и с его помощью для заданных шаблона P (|P|≤15000|P|≤15000) и текста TT (|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

Задача 2.

Заданы две строки A (|A|≤5000000|A|≤5000000) и B (|B|≤5000000|B|≤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 вляется циклическим сдвигом B, индекс начала строки B в A, иначе вывести −1. Если возможно несколько сдвигов вывести первый индекс.

Выполнение работы.

Для выполнения двух заданий была написана префиксная функция prefix_function, которая возвращает вектор результатов для каждого символа строки. В эту функцию передаётся строка. Будем считать символы строки с 0. Идем циклом от i = 1 до последнего символа строки. К j присваиваем предыдущее решение из вектора. Далее пока j>0 и i-й символ строки не равен j-му, то j присваиваем решение функции предыдущего от j символа. Далее если i-й символ строки равен j-му, то значение функции от i = j+1, иначе функция i = j.

Алгоритм KMP – передаём в prefix_function (P+#+T)=pi. И далее перебором проходим по pi и смотрим где значение функции равно длине P. Заносим индекс в вектор результатов. Далее при помощи функции Print выводим результат на экран.

Алгоритм Shift(сдвиг) – Проверяем на то, чтобы длина строки A была равна длине строки B. Если это так, то передаём в prefix_function (B+#+A+A)=pi, далее идем перебором по pi, если находим значение функции равно длине B, то выводим индекс и завершаем программу. Если перебор прошёл и мы ничего не нашли или строки не равны, то выводим -1.

Вывод:

В ходе лабораторной работы были написаны две программы, для удобства они представлены в одном main.cpp, одна из которых решает задачу поиска всех вхождений подстроки в строку, а другая определяет является ли одна строка циклическим сдвигом другой. Основу обеих программ составила префиксная функция. С помощью этой функции определяются все вхождения подстроки в строку (Алгоритм Кнута-Морриса-Пратта) и является ли строка циклическим сдвигом. В КМП составляем строку из подстроки, разделителя и строки. Отправляем её в префиксную функцию и далее идём циклом по нашей составленой строке и ищем позицию, у которой результат совпадает с длиной подстроки – этот номер будет индексом вхождения. Для циклического сдвига тоже самое, только вначале проверяется размер строк, он должен совпадать. И после разделителя строка дублируется 2 раза. Две программы прошли все тесты на Stepik.

ПРИЛОЖЕНИЕ

ИСХОДНЫЙ КОД ПРОГРАММЫ

main.cpp

```
#include <bits/stdc++.h>

using namespace std;
// Префиксная функция, передаём строку, возвращает вектор результатов
vector<int> prefixFunction(const string &string) {
    vector<int> prefixArray(string.length(), 0);

    for (int i = 1; i < string.length(); i++) {
        int j = prefixArray[i - 1];

        while (j > 0 && string[i] != string[j]) {
            j = prefixArray[j - 1];
        }

        if (string[i] == string[j]) {
            prefixArray[i] = j + 1;
        } else {
            prefixArray[i] = j;
        }
    }

    return prefixArray;
}
// Вывод результатов для KMP
void print(vector<int> &arrayEntryPosition) {
    for (int i = 0; i < arrayEntryPosition.size(); i++) {
        cout << arrayEntryPosition[i];
        if (i != arrayEntryPosition.size() - 1)
            cout << ',';
    }
    if (arrayEntryPosition.size() == 0) {
        cout << "-1";
    }
}
// Алгоритм KMP. Считаем префиксную функцию от строки состоящий из подстроки разделителя # и строки. А дальше просто
// сравниваем каждую ячейку префиксной функции с длиной подстроки.
void KMP(string &pattern, string &string) {
    vector<int> prefixArray = prefixFunction(pattern + '#' + string);
    int lengthPattern = pattern.length();
    vector<int> result;
    for (int i = 0; i < string.length(); i++) {
        if (prefixArray[lengthPattern + 1 + i] == lengthPattern) {
            result.push_back(i - lengthPattern + 1);
        }
    }
    print(result);
}
// алгоритм определения сдвига. Похож на KMP только строку удваиваем и работаем с равными.
```

```
    // И если встречаем в префиксе равную длине строки, то выводим индекс, иначе -
1.
    void cyclicShift(string &stringA, string &stringB) {
        if (stringB.size() == stringA.size()) {
            vector<int> prefixArray = prefixFunction(stringB + '#' + stringA +
stringA);
            int lengthB = stringB.length();
            for (int i = 0; i < stringA.length() * 2; i++) {
                if (prefixArray[lengthB + 1 + i] == lengthB) {
                    cout<< i - lengthB + 1;
                    exit(0);
                }
            }
        }
        cout<<"-1";
    }

    int main() {
        string pattern, string;
        cin >> pattern >> string;
        KMP(pattern, string);
        cyclicShift(pattern, string);
    }
```