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: Алгоритм Кнута-Морриса-Пратта

Студентка гр. 7383 _____ Прокопенко Н.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург

2019

Цель работы

Исследовать и реализовывать задачу поиска вхождения подстроки в строке, используя алгоритм Кнута – Морриса – Пратта.

Формулировка задачи: необходимо разработать программу, которая реализует алгоритм КМП и с его помощью для заданных шаблона P ($|P|\leq15000$) и текста T ($|T|\leq5000000$) найдите все вхождения P в T. Если P не входит в T, то вывести −1. Также следует разработать программу для решения следующей задачи: заданы две строки A ($|A|\leq5000000$) и B ($|B|\leq5000000$).

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. Если A является циклическим сдвигом B, индекс начала строки B в A, иначе вывести −1.

Вариант 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.

Входные данные: в первой строке указывается строка шаблона P, а во второй текст, в котором ищем подстроки.

Выходные данные: индексы вхождения подстроки в строку.

Реализация задачи

В данной работе для решения поставленной цели были написаны главная функция `main()` и дополнительные `void prefix(vector<int> &pi, string P)` и `void KMP(vector<int> &pi, vector<int> &result, string P, string T)`.

Функция для `lab4_1.cpp` `void KMP(vector<int> &pi, vector<int> &result, string P, string T)`, которая непосредственно реализует поиск подстроки в строке. В качестве аргументов функция принимает две строки: строку – образ и строку, в которой необходимо производить поиск, а также вектор в который будут записаны индексы начал вхождения `P в T` и вектор `pi`. Функция работает следующим образом: пока не был достигнут конец строки, выполняется сравнение символов строки и подстроки. Если символы равны, индексы увеличиваются, и функция

переходит к сравнению следующих символов. Если символы оказались не равны и индекс образа не указывает на его начало, то новый индекс образа вычисляется с использованием префикс-функции. Иначе индекс строки увеличивается на 1.

Функция `void prefix(vector<int> &pi, string P)` префикс-функция для подстроки, которая определяет наибольшую длину префикса, который одновременно является суффиксом для данной подстроки. Функция заполняет вектор типа int, который хранит максимальные длины суффиксов, которые одновременно являются суффиксами подстроки.

Функция для `lab4_2.cpp` `void KMP(vector<int> &pi, int &res, string P, string T)`, которая проверяет на цикличность две строки. Если длины исходных строк не равны или не удалость найти вхождение подстроки в строку, то вектор `res` остается с значением -1. Если строки, переданные данной функции, равны, то функция возвращает значение равное 0. Иначе выполняется поиск полстроки в строке, с условием, что, если был достигнут конец строки, в которой выполняется поиск, то индекс обнуляется, и поиск продолжается дальше. В результате функция меняет значение `res` на индекс начала одной строки в другой строке.

В главной функции `main()` считывается две строки вызывается префикс-функция. Далее вызывается функция поиска вхождения подстроки в строке, используя алгоритм Кнута – Морриса – Пратта.

Если считанные строки равны, то выводится значение равное 0.

Тестирование

Программа собрана в операционной системе Ubuntu 17.04 с использованием компилятора g++. В других ОС и компиляторах тестирование не проводилось. Результаты тестирования показали, что поставленная цель выполнена. Результаты тестирования представлены в Приложении Б.

Так же было проведено исследование алгоритма. В начале работы программа вычисляет значения префикс функции для каждого символа

первой строки. Пусть P – длина первой строки, тогда T – длина второй. В этом случае сложность алгоритма по времени будет составлять O(P + T).

По памяти сложность алгоритма составляет O(P).

Выводы

В ходе выполнения данной лабораторной работы были изучен и реализован на языке программирования c++ алгоритм Кнута – Морриса – Пратта. Был написан код на языке программирования C++, который применял этот метод для поставленной задачи. Полученный алгоритм имеет линейную сложность как по времени, так и по памяти. Код программы представлен в приложении А.

ПРИЛОЖЕНИЕ А.
КОД ПРОГРАММЫ

Lab4_1.cpp

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

void prefix(vector<int> &pi, string P){
    int i = 1, j = 0;
    pi[0] = 0;
    while(i < P.length()){
        if(P[i] == P[j]){
            pi[i] = j + 1;
            i++, j++;
        } else if(j == 0){
            pi[i] = 0;
            i++;
        } else j = pi[j-1];
    }
}

void KMP(vector<int> &pi, vector<int> &result, string P, string T){
    int k = 0, l = 0;
    while(k < T.length()){
        if(T[k] == P[l]){
            k++, l++;
            if(l == P.length())
                result.push_back(k - l);
        } else if(l == 0){
            k++;
        } else l = pi[l -1];
    }
}

int main()
{
    string P, T;
    getline(cin, P);
     getline(cin, T);
     vector<int> pi(P.size()), result;
     prefix(pi, P);
     KMP(pi, result, P, T);
     if(result.size() != 0){
         for(int i = 0; i < result.size(); i++){
```

```
            cout << result[i];
            if(i != result.size() - 1)
                cout <<",";
        }
     } else cout << "-1";
    return 0;
}
```

Lab4_2.cpp

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

void prefix(vector<int> &pi, string P){
    int i = 1, j = 0;
    pi[0] = 0;
    while(i < P.length()){
        if(P[i] == P[j]){
            pi[i] = j + 1;
            i++, j++;
        } else if(j == 0){
            pi[i] = 0;
            i++;
        } else j = pi[j-1];
    }
}

void KMP(vector<int> &pi, int &res, string P, string T){
    int k = 0, l = 0;
    if(P.length() == T.length())
        while(true){
            if(k == T.length())
                k = 0;
            if(T[k] == P[l]){
                k++, l++;
                if(l == P.length()){
                    res = (l - k);
                    break;
                }
            } else if(l == 0){
                        k++;
                        if(k == T.length())
                            break;
            } else l = pi[l -1];
```

```
    }
}

int main()
{
    string P, T;
    getline(cin, P);
      getline(cin, T);
      vector<int> pi(P.size());
      int res = -1;
      prefix(pi, P);
      KMP(pi, res, P, T);
      cout << res;
      pi.clear(), P.clear(), T.clear();
    return 0;
}
```

ПРИЛОЖЕНИЕ Б.

ТЕСТОВЫЕ СЛУЧАИ

Результаты тестов представлены в табл. 1.

Входные данные	Выходные данные
defabc abcdef	3
defabcdfgh abcdef	-1
asdfgh asdfgh	0