**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе №4**

**по дисциплине «Проектирование и анализ алгоритмов»**

Тема: Алгоритм Кнута-Морриса-Пратта

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Запевалов А.И. |
| Преподаватель |  | Филатов А.Ю. |

Санкт-Петербург

2019

**Цель работы.**

Разработать программу, реализующую алгоритм Кнута-Морриса-Пратта. На вход программе подаются шаблон и текст, в котором следует найти все вхождения данного шаблона.

Даны образец (строка) S и строка T. Требуется определить индекс, начиная с которого образец содержится в строке . Если S не содержится в T — вернуть индекс, который не может быть интерпретирован как позиция в строке (например, отрицательное число). При необходимости отслеживать каждое вхождение образца в текст имеет смысл завести дополнительную функцию, вызываемую при каждом обнаружении образца.

**Основные теоретические положения.**

Принципиальное отличие алгортма Кнутта-Морриса-Пратта от наивного состоит в том, что в нем используется информация о предыдущих совпадениях. Это позволяет не производить лишних сравнений, если оказывается что строки не равны на каком-либо участке. Порядок сложности наивного алгоритма составляет длина строки \* длина шаблона а при использовании КМП надо лишь один раз создать таблицу префикс-функций, что имеет количество сравнений порядка длины шаблона и произвести эффективный поиск (поряд длины строки).

**Экспериментальные результаты.**

Входные данные:

ab

ababcdoeabe

Результат:

0,2,8

**Обработка результатов эксперимента.**

Подход к выполнению задачи.

1. Используемые структуры.

Основная структура, используемая для вывода результата (queue <int> result) - очередь целочисленных индексов символов строк с соответствующими значениями префикс-функции в них. Также используется объединенная строка(P + "#" + T) из шаблона и текста с символом-разделителем между ними. Результат работы префикс-функции записывается в целочисленный массив(vector<int> p = prefix\_function(P + "#" + T)).

2. Основная функция и принцип ее работы.

Известно, что значение префикс-функции может меняться в пределах от 0 до n+1, где n — значение префикс-функции на предыдущем символе. Если текущие символы шаблона и текста совпадают, то оно увеличивается на 1. Если нет, то сравниваются текущий символ текста и символ с индексом под номером значения префикс-функции в символе под индексом n-1, если символы не равны, то это повторяется с использованием получаемых индексов из значений префикс-функции, вместо n-1 используется индекс, на 1 меньше использованного ранее. Когда очередной индекс равен 0, происходит выход из цикла и проверяется, не равны ли первый символ шаблона и текущий символ. Если они равны, то префикс-функция равна 1, иначе — 0. Вышеизложенное повторяется для следующего символа из текста и шаблона.

После вычисления значений префикс-функции для всех символов текста, выбираются индексы, в которых префикс-функция равна длине шаблона, и вносятся результирующий массив.

**Выводы.**

Был реализован эффективный поиск шаблона в тексте по алгоритму Кнутта-Морриса-Пратта. При выполнении данной лабораторной работы были решены проблемы реализации префикс-функции, а также использования ее результатов для определения позиций искомого шаблона в заданном тексте. Было найдено решение задачи нахождения циклического сдвига с использованием префикс-функции. Для каждого вхождения в тексте возвращался индекс позиции первого символа шаблона. Скорость получения результатов лабораторной работы соответствуют теоретически ожидаемой.

**ПРИЛОЖЕНИЕ А**

**Исходный код программы.**

#include <iostream>

#include <vector>

#include <string>

#include <queue>

using namespace std;

vector<int> prefix\_function (string s) {

int n = (int) s.length();

vector<int> p (n);

for (int i=1; i<n; ++i) {

int j = p[i-1];

while (j > 0 && s[i] != s[j])

j = p[j-1];

if (s[i] == s[j]) ++j;

p[i] = j;

}

return p;

}

void kmp(queue <int> &answer, string P, string T)

{

int pl = P.length();

int tl = T.length();

vector<int> p = prefix\_function(P + "#" + T);

for (int i=0; i<tl; i++){

if (p[pl+i+1] == pl)

answer.push(i - pl + 1);

}

}

int main()

{

string P, T;

queue <int> result;

cin >> P;

cin >> T;

kmp(result, P, T);

if (result.size())

{

cout << result.front();

result.pop();

while (!result.empty())

{

cout << ',' << result.front();

result.pop();

}

}

else

cout << -1;

return 0;

}