**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. 8304 |  | Рыжиков А.В. |
| Преподаватель |  | Размочаева Н.В. |

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.

**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.

**Задание.**

Реализуйте алгоритм КМП и с его помощью для заданных шаблона P и текста T 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

## Описание алгоритма.

На вход алгоритма передается строка-образец, 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-образца.

Далее посимвольно считывается строка-текст. Переменная-счетчик изначально k = 0. При каждом совпадении k-го символа образца и i-го символа текста счетчик увеличивается на 1. Если k = размер образца, значит вхождение найдено. Если очередной символ текста не совпал с k-ым символом образца, то сдвигаем образец, причем точно знаем, что первые k символов образца совпали с символами строки и надо сравнить k+1-й символ образца (его индекс k) с i-м символом строки.

Сложность алгоритма по операциям: O (m + n), m – длина образца, n – длина текста.

Сложность алгоритма по памяти: O (m), m – длина образца.

**Вывод промежуточной информации.**

Во время основной части работы алгоритма происходит вывод промежуточной информации, а именно, значения префикс функции и проверка идентичности префикса и суффикса первой и второй строки соответственно.

**Тестирование.**

Таблица 1 – Результаты тестирование

|  |  |
| --- | --- |
| **Ввод** | **Вывод** |
| ab  abab | 0,2 |
| ababab  ababab | 0 |
| alex  alexalexalex | 0,4,8 |
| aba  ababababa | 0,2,4,6 |

# Вывод.

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

**ПРИЛОЖЕНИЕ А.  
ИСХОДНЫЙ КОД**

#include <iostream>  
#include <vector>  
  
void createPiArray(std::vector<int> \*vector, std::string \*string, int length) {  
  
 int j = 0;  
 int i = 1;  
  
 vector->emplace\_back(0);  
  
 while (length > i) {  
 if (string->at(i) == string->at(j)) {  
 vector->emplace\_back(j + 1);  
 i++;  
 j++;  
 } else {  
 if (j == 0) {  
 vector->emplace\_back(0);  
 i++;  
 } else {  
 j = vector->at(j - 1);  
 }  
 }  
 }  
}  
  
int main() {  
 std::string string;  
  
 std::cin >> string;  
  
 std::vector<int> vector;  
 vector.reserve(0);  
  
 std::vector<int> answer;  
 vector.reserve(0);  
  
 createPiArray(&vector, &string, string.length());  
  
 char c;  
 std::cin.get(c);  
  
 int l = 0;  
 int n = string.size();  
  
 int count = 0;  
 std::cin.get(c);  
 while (true) {  
 bool isCinActive = true;  
  
 if (c == string[l]) {  
 l++;  
 count++;  
 if (l == n) {  
 answer.emplace\_back(count - n);  
 }  
 } else {  
 if (l == 0) {  
 count++;  
 } else {  
 l = vector.at(l - 1);  
 isCinActive = false;  
 }  
 }  
  
 if (isCinActive) {  
 std::cin.get(c);  
 }  
  
 if (c == '\n') {  
 break;  
 }  
 }  
  
 if (!answer.empty()) {  
 for (size\_t m = 0; m < answer.size(); ++m) {  
 std::cout << answer[m];  
 if (m != answer.size() - 1) std::cout << ",";  
 }  
 } else {  
 std::cout << -1;  
 }  
  
  
 return 0;  
}