**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**

Студент гр.8304 \_\_\_\_\_\_\_\_\_\_\_\_\_ Холковский К.В.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_ Размочаева Н.В.

Санкт-Петербург

2020

**Задание.**

Вариант 2

Оптимизация по памяти: программа должна требовать O(m) памяти, где m - длина образца.

Заданы две строки *A* (∣*A*∣≤5000000) и *B* (∣*B*∣≤5000000).

Определить, является ли А циклическим сдвигом В (это значит, что А и В имеют одинаковую длину и А состоит из суффикса В, склеенного с префиксом В). Например, defabc является циклическим сдвигом abcdef.

**Цель работы.**

Разработать программу, определяющую, является ли А циклическим сдвигом В.

**Описание алгоритма.**

К строке B прибавляем строку B и в полученной строке ищем на вхождение строку A алгоритмом Кнута-Морриса-Пратта.

При сдвиге вполне можно ожидать, что префикс образца {\displaystyle \displaystyle S}S сойдется с каким-нибудь суффиксом текста {\displaystyle \displaystyle P}T. Тогда, пусть p[j] {\displaystyle \displaystyle {\rm {{\pi }[j]}}}ppзppp— значение префикс-функции от строки S {\displaystyle \displaystyle S[0,m-1]}Sдля индекса j{\displaystyle \displaystyle j}. Тогда после сдвига мы можем возобновить сравнения с места T[i + j] {\displaystyle \displaystyle T[i+j]} и S[p[j]] {\displaystyle \displaystyle S[{\rm {{\pi }[j]]}}}без потери возможного местонахождения образца. Сложность алгоритма O(m + n), где m – длинна строки которую мы ищем, а n – длинна строки в которой мы ищем.

**Описание функций и структур данных.**

Функция, находящая префикс строки S за линейное время:

**void** Prefix(std::string **const**& S, std::vector<**int**>& A);

Функция, возвращающая индекс с которого S находится в T:

**int** KMP(std::string & S, std::string & T);

**Тестирование**

Таблица 1 – результаты тестирования

|  |  |
| --- | --- |
| **Input** | **Output** |
| defabc  abcdef | 3 |
| adasd  zxcv | -1 |
| aaaaaaab  aaaaabaa | 2 |

**Выводы.**

В ходе выполнения работы, была написана программа, определяющая: является ли одна строка циклическим сдвигом другой строки.

**ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД**

#include **<iostream>**#include **<vector>**#include **<fstream>  
  
constexpr const char**\* PATH\_IN = **"D:/test.txt"**;  
**constexpr const char**\* PATH\_OUT = **"D:/result.txt"**;  
  
**void** Prefix(std::string **const**& S, std::vector<**int**>& A){  
 A[0]=0;  
 **for**(**int** i = 1; i<S.size();++i) {  
 **int** k = A[i - 1];  
 **while** (k > 0 && S[i] != S[k])  
 k = A[k - 1];  
 **if** (S[k] == S[i - k])  
 A[i] = k;  
 **if**(S[i] == S[k])  
 ++k;  
 A[i]=k;  
 }  
}  
  
**int** KMP(std::string & S, std::string & T) {  
 **int** size = S.size();  
 **int** pos = -1;  
 **int** k = 0;  
 std::vector<**int**> P(size);  
 Prefix(S,P);  
 **for**(**int** i = 0; i < T.size();++i) {  
 **while**(k>0 && T[i] != S[k])  
 k = P[k-1];  
 **if**(T[i] == S[k])  
 ++k;  
 **if**(k == size) {  
 pos = i - size + 1;  
 **return** pos;  
 }  
 }  
 **return** pos;  
}  
  
**int** main() {  
 std::string S, T;  
  
 **int** choseIn, choseOut;  
 std::cout << **"Input: 1 - console, 0 - file"** << std::endl;  
 std::cin >> choseIn;  
 **if**(choseIn!=0 && choseIn!=1) {  
 std::cout << **"Wrong chose Input"**;  
 **return** 0;  
 }  
 std::cout << **"Output: 1 - console, 0 - file"** << std::endl;  
 std::cin >> choseOut;  
 **if**(choseOut!=0 && choseOut!=1) {  
 std::cout << **"Wrong chose Output"**;  
 **return** 0;  
 }  
 **if**(choseIn == 1)  
 std::cin >> T >> S;  
 **else**{  
 std::ifstream file;  
 file.open(PATH\_IN);  
  
 **if** (!file.is\_open()) {  
 std::cout << **"Can't open file!\n"**;  
 **return** 0;  
 }  
 file >> T >> S;  
 }  
  
 T+=T;  
 **if**(choseOut ==1)  
 std::cout << KMP(S,T);  
 **else**{  
 std::ofstream file;  
 file.open(PATH\_OUT);  
  
 **if** (!file.is\_open()) {  
 std::cout << **"Can't open file!\n"**;  
 **return** 0;  
 }  
 file << KMP(S,T);  
 }  
 **return** 0;  
}