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

Студентка гр. 8382 _____ Кулачкова М.К.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2020

Цель работы.

Построить алгоритм КМП поиска всех вхождений подстроки в строку, а также алгоритм, определяющий, является ли одна строка циклическим сдвигом другой.

Задание.

Заданы две строки A ($|A| \leq 5000000$) и B ($|B| \leq 5000000$).

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

Sample Input:

defabc
abcdef

Sample Output:

3

Вариант дополнительного задания.

Вар. 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.

Описание алгоритма.

Алгоритм КМП реализован следующим образом. Чтобы не производить лишних действий, сначала сравниваются длины образца и строки поиска. Очевидно, что, если строка поиска короче образца, образец в ней встретиться

не может. Затем для образца и строки вычисляется несколько модифицированная префикс-функция, которая позволяет не создавать копию строки поиска и таким образом сэкономить память.

Сначала заполняется массив префиксов для образца. Затем осуществляется сравнение строки поиска с образцом. В переменную записывается число совпавших символов (оно же будет индексом рассматриваемого символа в образце). Если очередной символ строки не совпал с рассматриваемым символом в образце, сдвигаемся по образцу назад, причем если суффикс образца совпадает с его префиксом, мы перемещаемся в конец префикса. Иначе увеличиваем длину совпавшего фрагмента на 1. Когда длина совпавшего фрагмента равна длине образца, был найден конец вхождения образца в строку и, вычитая из индекса текущего элемента строки длину образца, можно получить индекс начала вхождения образца в строку. Все эти индексы записываются в массив и выводятся на экран по окончании работы алгоритма.

Алгоритм определения циклического сдвига основан на алгоритме КМП. Сначала, во избежание излишних запусков алгоритма, проверяются длины строк. Если длины различны, очевидно, что одна строка не может быть циклическим сдвигом другой. Затем проверяется, совпадают ли строки. Если они совпадают, то индекс вхождения второй строки в первую будет равен 0. Затем создается вспомогательная строка, которая будет равна двум строкам A, склеенным подряд. В полученной строке с помощью алгоритма КМП осуществляется поиск строки B. Если строка B является циклическим сдвигом строки A, то она гарантированно войдет в полученную вспомогательную строку. Первое вхождение строки B во вспомогательную строку и будет индексом вхождения строки B в строку A.

Оценка сложности алгоритма.

Временная сложность алгоритма КМП составляет $O(P + T)$, где P – длина образца, T – длина строки поиска. Сложность алгоритма по памяти составляет $O(P)$, так как префикс-функция вычисляется только для образца.

Временная сложность алгоритма определения циклического сдвига составляет $O(3T)$, так как сначала идет вычисление префикс-функции для строки B длины T, а затем сравнение символов вспомогательной строки длины 2T со строкой B. Сложность алгоритма по памяти будет составлять также $O(3T)$, так как создается вспомогательная строка длины 2T, однако алгоритм КМП, используемый при решении задачи, обладает пространственной сложностью $O(T)$.

Тестирование.

№	Ввод	Вывод
1	defabc abcdef	3
2	flabbergasted flabbergasted	0
3	abcdefabcdef defabcdefabc	3
4	abcdef abc	-1
5	abcdef defghi	-1

Выводы.

Были реализованы два алгоритма: алгоритм КМП поиска всех вхождений подстроки в строку и алгоритм, определяющий, является ли одна строка циклическим сдвигом другой. Для алгоритмов были вычислены временные и пространственные сложности.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл main.cpp

```
#include <iostream>
#include <vector>
#include <string>

using namespace std;

//префикс-функция
void prefixFunction(vector<int> &answ, string str, string patt) {
	cout << "Calculating preffix-function" << endl;
	vector<int> pi(patt.length());
	pi[0] = 0;
	cout << "\tPrefix(pattern[0]) = 0" << endl;
	//заполняем массив префиксов для образца
	for (int i = 1; i < patt.length(); i++) {
		int j = pi[i - 1]; //берем предыдущее значение и пытаемся его увеличить
		while ((j > 0) && (patt[j] != patt[i])) { //не выходит -
			j = pi[j - 1]; //берем максимально возможное из ранее рассчитанных
		}
		if (patt[i] == patt[j]) {
			j++;
		}
		pi[i] = j;
		cout << "\tPrefix(pattern[" << i <<"]) = " << j << endl;
	}
	int j = 0; //число совпавших символов
	//оно же индекс сравниваемого элемента в образце
	for (int i = 0; i < str.length(); i++) {
		cout << "\tCurrent symbol in main string: " << str[i] << endl;
		while ((j > 0) && (patt[j] != str[i])) { //символ не совпал
			cout << "\tCurrent symbol in pattern: " << patt[j] << endl;
			j = pi[j - 1]; //сдвигаем образец
		}
		if (str[i] == patt[j]) { //символ совпал - увеличиваем длину совпавшего
фрагмента
			cout << "\tCurrent symbol in pattern: " << patt[j] << endl;
			j++;
		}
		if (j == patt.length()) {  //обошли весь образец
			cout << "\tFound a match" << endl;
			answ.push_back(i - patt.length() + 1); //добавили индекс его вхождения
		}
	}
}

//алгоритм КМП
vector<int> kmp(string pattern, string str) {
	vector<int> answ;
	if (str.length() < pattern.length()) {
		answ.push_back(-1);
		return answ;
	}
	//вычисляем префикс-функцию
	prefixFunction(answ, str, pattern);
	if (answ.empty()) {
		answ.push_back(-1);
	}
```

```
	return answ;
}

//определение циклического сдвига
int cycle(string strA, string strB) {
	if (strA.length() != strB.length()) {
		cout << "Different string lengths!" << endl;
		return -1;
	}
	if (strA == strB) {
		cout << "Strings are equal" << endl;
		return 0;
	}
	string tmp = strA + strA;
	cout << "Transforming \"" << strA << "\" into \"" << tmp << "\"" << endl;
	cout << "Starting KMP" << endl;
	vector<int> answ = kmp(strB, tmp);
	return answ[0];
}

int main() {
	string strA, strB;
	//ввод для КМП
	//cout << "Enter pattern, then enter text: " << endl;
	//ввод для определения циклического сдвига
	cout << "Enter string A and string B: " << endl;
	cin >> strA >> strB;

	//КМП
	/*vector<int> answ = kmp(strA, strB);
	cout << "Pattern positions within a main string: " << endl;
	for (int i = 0; i < answ.size(); i++) {
		if (i == answ.size() - 1) {
			cout << answ[i] << endl;
		}
		else {
			cout << answ[i] << ",";
		}
	}*/

	//определение циклического сдвига
	int res = cycle(strA, strB);
	cout << "Starting position of string B within string A: ";
	cout << res << endl;
	return 0;
}
```