Санкт-Петербургский Национальный Исследовательский

Университет Информационных Технологий, Механики и Оптики Мегафакультет компьютерных технологий и управления

Дисциплина

«Алгоритмы и структуры данных»

Лабораторная работа №1

Выполнил:

Студент группы P3218

Рябов Сергей Витальевич Преподаватель:

Муромцев Дмитрий Ильич

Санкт-Петербург, 2018

**1. Наивный поиск подстроки в строке**

Даны строки p и t. Требуется найти все вхождения строки p в строку t в качестве подстроки.

**Формат входного файла**

Первая строка входного файла содержит p, вторая — t (1≤|p|,|t|≤104). Строки состоят из букв латинского алфавита.

**Формат выходного файла**

В первой строке выведите число вхождений строки p в строку t. Во второй строке выведите в возрастающем порядке номера символов строки t, с которых начинаются вхождения p. Символы нумеруются с единицы.

**Исходный код (C#):**

using System;

using System.IO;

using System.Collections.Generic;

using System.Globalization;

namespace ADS.Week9

{

public class SearchableString

{

private string str;

public int Length { get { return str.Length; } }

private const int x = 27;

private const long p = 223372036854775801;

public SearchableString(string s)

{

str = s;

}

public List<int> IndexesOf(string s)

{

if (s.Length > str.Length)

return new List<int>();

long hash = GetHash(s);

List<int> indexes = new List<int>();

for (int i = 0; i <= str.Length - s.Length; i++)

{

string substring = str.Substring(i, s.Length);

if (hash == GetHash(substring) && substring.Equals(s))

indexes.Add(i);

}

return indexes;

}

private long GetHash(string s)

{

long hash = 0;

for (int i = 0; i < s.Length; i++)

{

hash = (hash \* x + s[i]) % p;

}

return hash;

}

}

public class Task1

{

public static void main(string[] args)

{

using (StreamReader streamReader = new StreamReader("input.txt"))

using (StreamWriter streamWriter = new StreamWriter("output.txt"))

{

string substring = streamReader.ReadLine();

SearchableString str = new SearchableString(streamReader.ReadLine());

List<int> indexes = str.IndexesOf(substring);

streamWriter.WriteLine(indexes.Count);

for (int i = 0; i < indexes.Count; i++)

streamWriter.Write("{0} ", indexes[i] + 1);

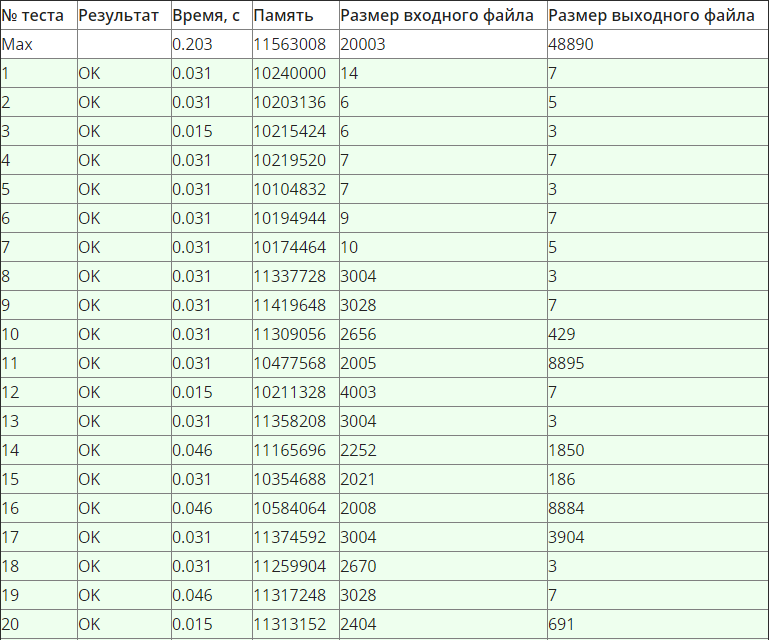
}

}

}

}

**Результат:**



**2. Карта**

В далеком 1744 году во время долгого плавания в руки капитана Александра Смоллетта попала древняя карта с указанием местонахождения сокровищ. Однако расшифровать ее содержание было не так уж и просто.

Команда Александра Смоллетта догадалась, что сокровища находятся на x шагов восточнее красного креста, однако определить значение числа она не смогла. По возвращению на материк Александр Смоллетт решил обратиться за помощью в расшифровке послания к знакомому мудрецу. Мудрец поведал, что данное послание таит за собой некоторое число. Для вычисления этого числа необходимо было удалить все пробелы между словами, а потом посчитать количество способов вычеркнуть все буквы кроме трех так, чтобы полученное слово из трех букв одинаково читалось слева направо и справа налево.

Александр Смоллетт догадывался, что число, зашифрованное в послании, и есть число x. Однако, вычислить это число у него не получилось.

После смерти капитана карта была безнадежно утеряна до тех пор, пока не оказалась в ваших руках. Вы уже знаете все секреты, осталось только вычислить число x.

**Формат входного файла**

В единственной строке входного файла дано послание, написанное на карте. Длина послания не превышает 3⋅105. Гарантируется, что послание может содержать только строчные буквы английского алфавита и пробелы. Также гарантируется, что послание не пусто. Послание не может начинаться с пробела или заканчиваться им.

**Формат выходного файла**

Выведите одно число x — число способов вычеркнуть из послания все буквы кроме трех так, чтобы оставшееся слово одинаково читалось слева направо и справа налево.

**Исходный код (C#):**

using System;

using System.IO;

using System.Collections.Generic;

using System.Globalization;

namespace ADS.Week9

{

public class Map

{

private string str;

public Map(string s)

{

str = s.Replace(" ", string.Empty);

}

public long CalculateX()

{

List<int>[] indexes = new List<int>[26];

for (int i = 0; i < indexes.Length; i++)

indexes[i] = new List<int>();

for (int i = 0; i < str.Length; i++)

indexes[str[i] - 'a'].Add(i);

long count = 0;

foreach(List<int> letterIndexes in indexes)

for (int i = 0; i < letterIndexes.Count - 1; i++)

count += (((long)letterIndexes[i + 1] - letterIndexes[i])

\* (letterIndexes.Count - i - 1) - 1) \* (i + 1);

return count;

}

}

public class Task2

{

public static void Main(string[] args)

{

using (StreamReader streamReader = new StreamReader("input.txt"))

using (StreamWriter streamWriter = new StreamWriter("output.txt"))

{

Map map = new Map(streamReader.ReadLine());

streamWriter.WriteLine(map.CalculateX());

}

}

}

}

**Результат:**

