|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет имени Н.Э. Баумана**  **(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ                              Информатика и системы управления

КАФЕДРА                  Системы обработки информации и управления

**Лабораторная работа №5**

**По курсу «Базовые компоненты интернет-технологий»**

Подготовила:

Студентка группы ИУ5-35Б.

Очеретная С.В.

18.11.2020

Проверил:

Преподаватель кафедры ИУ5  
Гапанюк Ю.Е.

Москва, 2020 г.

Задание.

Разработать программу, реализующую вычисление расстояния Левенштейна с использованием алгоритма Вагнера-Фишера.

1. Программа должна быть разработана в виде библиотеки классов на языке C#.
2. Использовать самый простой вариант алгоритма без оптимизации.
3. Дополнительно возможно реализовать вычисление расстояния Дамерау-Левенштейна (с учетом перестановок соседних символов).
4. Модифицировать предыдущую лабораторную работу, вместо поиска подстроки используется вычисление расстояния Левенштейна.
5. Предусмотреть отдельное поле ввода для максимального расстояния. Если расстояние Левенштейна между двумя строками больше максимального, то строки считаются несовпадающими и не выводятся в список результатов.

Текст программы

Dameray\_Levenshtayna.cs

using System;

using System.Data.Common;

namespace LABA5\_

{

//M - символ тот же

//I - добавление (insert)

//D - удаление (delete)

//R - замена (replace)

//T - транспонирование (transposition)

public class T9

{

public static int Dameray\_Levenshteyna(string str1, string str2) // c учетом транспонирования

{

if ((str1 == null) || (str2 == null)) { // проверка на наличие строк

return -1;

}

if ((str1.Length == 0) && (str2.Length == 0)) { // если обе строки пустые - они совпадают

return 0;

}

if (str1.Length == 0) { // одна строка - пустая, значит кол-во несовпадающих символов = кол-ву символов 2 строки

return str2.Length;

}

if (str2.Length == 0) {

return str1.Length;

}

int[,] D = new int[str1.Length + 1, str2.Length + 1];

string s1 = str1.ToLower(); // для сравнения приводим строки в одному регистру

string s2 = str2.ToLower();

for (int i = 0; i <= s1.Length; i++) { // иницициализация при i=0 или j=0 (по формуле для Левенштейна)

D[i, 0] = i;

}

for (int j = 0; j <= s2.Length; j++) {

D[0, j] = j;

}

int M = 0;

for (int i = 1; i <= s1.Length; i++){

for (int j = 1; j <= s2.Length; j++){

if (s1.Substring(i - 1, 1) == s2.Substring(j - 1, 1)) { // возвращаем строку i-1 (j-1) длины 1, т.е. сам символ

M = 0;

}

else { // если символы разные, увеличиваем расстояния для случая с заменой и транспонированием (или нет)

M = 1;

}

int Del = D[i - 1, j] + 1; // распишем 3 случая по формуле - удаление, добавление, вставка и + транспонирование (доп. Дамерау) и найдем минимальный

int Ins = D[i, j - 1] + 1;

int Rep = D[i - 1, j - 1] + M;

int Trans = Rep;

if (i > 1 && j > 1 && (s1.Substring(i - 1, 1) == s2.Substring(j - 2, 1)) && (s1.Substring(i - 2, 1) == s2.Substring(j - 1, 1))) {

Trans = D[i - 2, j - 2] + M;

}

D[i, j] = Math.Min(Math.Min(Math.Min(Ins, Del), Rep), Trans); // минимальный среди 4 операций

}

}

return D[s1.Length, s2.Length];

}

public static void PrintD\_Levenshteyna(string s1, string s2)

{

Console.WriteLine("D\_Levenshteyna of " + s1 + " and " + s2 + " = " + Dameray\_Levenshteyna(s1, s2));

}

}

}

Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using LABA5\_;

namespace LABA4\_

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

List<string> list = new List<string>(); //список слов

bool SearchDone = false;

private void buttonReadFile\_Click(object sender, EventArgs e) //кнопка, реализующая чтение файла в список слов (2)

{

OpenFileDialog SelectFile = new OpenFileDialog(); //открывает диалоговое окно для выбора файла(3)

SelectFile.Filter = "Только текстовые файлы|\*.txt"; //(3)

if (SelectFile.ShowDialog() == DialogResult.OK) //если файл был выбран

{

Stopwatch time = new Stopwatch(); //вычисление времени загрузки и сохранения(6)

time.Start();

string file = File.ReadAllText(SelectFile.FileName); //считывается всё содержимое файла в одну строку(4)

char[] separators = new char[] { ' ', '.', ',', '!', '?', '/', '\t', '\n' };

string [] fileArr = file.Split(separators); //содержимое файла разбивается на массив слов используя разделители(4)

foreach(string word in fileArr)

{

if (!list.Contains(word)) list.Add(word.Trim()); //сохранение слова в список(4) без повторения(5)

}

time.Stop();

this.textBoxLoadReadTime.Text = time.Elapsed.ToString(); //вывод времени загрузки и чтения файла(6)

}

else { MessageBox.Show("Файл не выбран!"); }

}

private void buttonSearchWord\_Click(object sender, EventArgs e)

{

string word = this.textBoxSearchWord.Text.Trim(); //считывание слова с текстовой панели

if (!string.IsNullOrWhiteSpace(word) && list.Count > 0)//если слово введено и список не пуст

{

Stopwatch search\_time = new Stopwatch(); // (9) время поиска

search\_time.Start();

List<string> flist = new List<string>(); //список со словами, которые содержат искомое слово (элемент) (7)

this.listBoxResult.BeginUpdate(); //(8)

this.listBoxResult.Items.Clear();

foreach (string el in list)

{

if (el.ToUpper().Contains(word.ToUpper())) //если слово входит в подстроку с учетом регистра букв (7)

{

flist.Add(el);

this.listBoxResult.Items.Add(el); //(8)

}

}

this.listBoxResult.EndUpdate(); //(8)

search\_time.Stop();

this.textBoxSearchTime.Text = search\_time.Elapsed.ToString(); // запись времени поиска в текстовое поле

}

else

{

MessageBox.Show("Введите слово для поиска и выберите файл!");

}

SearchDone = true;

}

private void buttonDLevenshteyna\_Click(object sender, EventArgs e) { // кнопка, с которой выводятся только слова, соответствующие расстоянию Д-Левенштейна

int D\_Lev\_dist = 0;

string str = this.textBoxSearchWord.Text.Trim(); // считывание слова для поиска

// MessageBox.Show("кол во слов: " + list.Count);

if (list.Count <= 0 || !SearchDone) { // если не введено (или неверно введно) макс. расстояние и нет слов в списке, среди которых ищем, то ошибка

MessageBox.Show("Не нашлись слова для поиска");

return;

}

else if (string.IsNullOrWhiteSpace(str)) {

MessageBox.Show("Не введено макс. расстояние Д-Левентштейна");

return;

} else if (!int.TryParse(this.MaxLentghTextBox.Text.Trim(), out D\_Lev\_dist)) {

MessageBox.Show("Не верно введено макс. расстояние Д-Левентштейна");

return;

} else {

Stopwatch time\_D\_Lev = new Stopwatch();

time\_D\_Lev.Start();

// Dictionary<string, int> search\_words = new Dictionary<string, int>();

this.listBoxResult.BeginUpdate(); // обновить поле для вывода слов

this.listBoxResult.Items.Clear(); // очистить его

foreach (string el in list) {

int dist = T9.Dameray\_Levenshteyna(str, el);

if (dist <= D\_Lev\_dist) {

// search\_words.Add(str, dist);

this.listBoxResult.Items.Add(el);

}

}

this.listBoxResult.EndUpdate();

time\_D\_Lev.Stop();

this.textBoxLoadReadTime.Text = time\_D\_Lev.Elapsed.ToString();

}

}

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace LABA4\_

{

static class Program

{

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

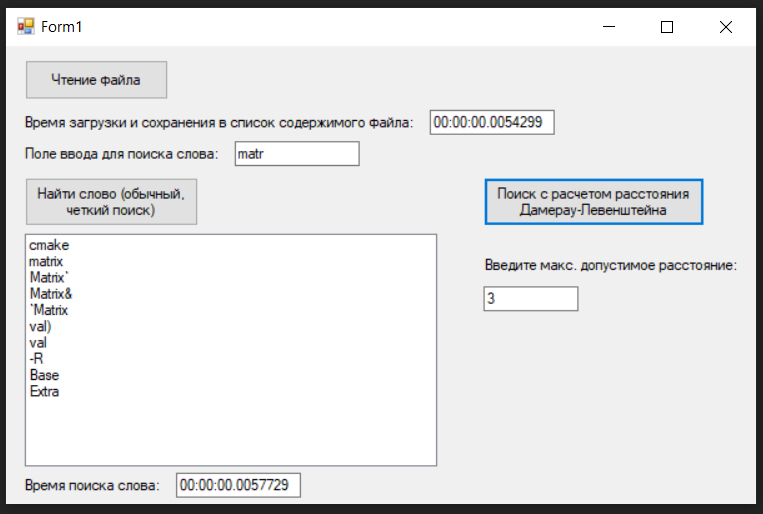
}

}

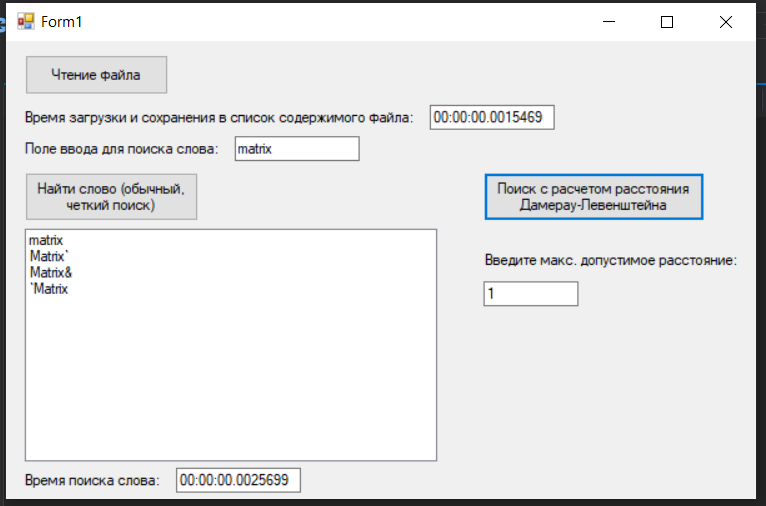
}

Результаты

Пример1:



Пример2:



Обработки ошибок:

