|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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)** |

Базовые компоненты интернет технологий

Отчет по лабораторной работе №5

Студент: Пылаев Б. А.

Группа: ИУ5Ц-52Б

Преподаватель: Гапанюк Ю. Е.

2020 г.

**Лабораторная работа №5**

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.

**Диаграмма классов**



**Текст программы**

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace LAB\_5

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

using System.Diagnostics;

namespace LAB\_5

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

List<string> list = new List<string>();

private static int Dist(string s1, string s2)

{

if (s1 == s2)

{

return 0;

}

var M = s1.Length + 1;

var N = s2.Length + 1;

var dist = new int[M, N];

for (var i = 0; i < M; i++)

{

dist[i, 0] = i;

}

for (var j = 0; j < N; j++)

{

dist[0, j] = j;

}

for (var i = 1; i < M; i++)

{

for (var j = 1; j < N; j++)

{

var diff = (s1[i - 1] == s2[j - 1]) ? 0 : 1;

dist[i, j] = Math.Min(

Math.Min(

dist[i - 1, j] + 1,

dist[i, j - 1] + 1

),

dist[i - 1, j - 1] + diff

);

}

}

return dist[M - 1, N - 1];

}

private static int DistDamerau(string s1, string s2)

{

if (s1 == s2)

{

return 0;

}

var M = s1.Length + 1;

var N = s2.Length + 1;

var dist = new int[M, N];

for (var i = 0; i < M; i++)

{

dist[i, 0] = i;

}

for (var j = 0; j < N; j++)

{

dist[0, j] = j;

}

for (var i = 1; i < M; i++)

{

for (var j = 1; j < N; j++)

{

if (s1[i - 1] == s2[j - 1])

{

dist[i, j] = dist[i - 1, j - 1];

}

var diff = (s1[i - 1] == s2[j - 1]) ? 0 : 1;

dist[i, j] = Math.Min(

Math.Min(

dist[i - 1, j] + 1,

dist[i, j - 1] + 1

),

dist[i - 1, j - 1] + diff

);

if (i > 1 && j > 1 && s1[i - 2] == s2[j - 1] && s1[i - 1] == s2[j - 2])

{

dist[i, j] = Math.Min(dist[i, j], dist[i - 2, j - 2] + 1);

}

}

}

return dist[M - 1, N - 1];

}

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

{

OpenFileDialog fd = new OpenFileDialog();

fd.Filter = "текстовые файлы|\*.txt";

if (fd.ShowDialog() == DialogResult.OK)

{

Stopwatch t = new Stopwatch();

t.Start();

//Чтение файла в виде строки

string text = File.ReadAllText(fd.FileName);

//Разделительные символы для чтения из файла

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

string[] textArray = text.Split(separators);

foreach (string strTemp in textArray)

{

//Удаление пробелов в начале и конце строки

string str = strTemp.Trim();

//Добавление строки в список, если строка не содержится в списке

if (!list.Contains(str)) list.Add(str);

}

t.Stop();

this.textFileReadTime.Text = t.Elapsed.ToString();

this.textUniqueWords.Text = list.Count.ToString();

}

else

{

MessageBox.Show("Файл не выбран!");

}

}

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

{

var expectedSubstring = wordSearchTextBox.Text.Trim();

if (!string.IsNullOrWhiteSpace(expectedSubstring) && list.Count > 0)

{

listBox.Items.Clear();

var tempList = new List<string>();

var stopwatch = new Stopwatch();

var isFinded = false;

var maxDist = Int32.Parse(textMaxDistance.Text);

stopwatch.Start();

this.listBox.BeginUpdate();

foreach (string word in list)

{

if (DistDamerau(word.ToUpper(), expectedSubstring.ToUpper()) <= maxDist)

{

listBox.Items.Add(word);

isFinded = true;

}

}

this.listBox.EndUpdate();

stopwatch.Stop();

if (!isFinded)

{

listBox.Items.Add("No matches!");

}

this.textWordSearchTime.Text = stopwatch.Elapsed.ToString() + " ms";

}

}

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

{

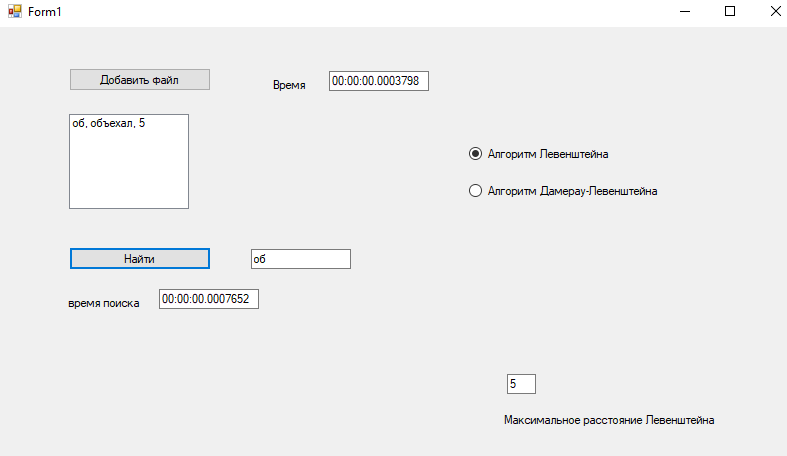
this.Close();

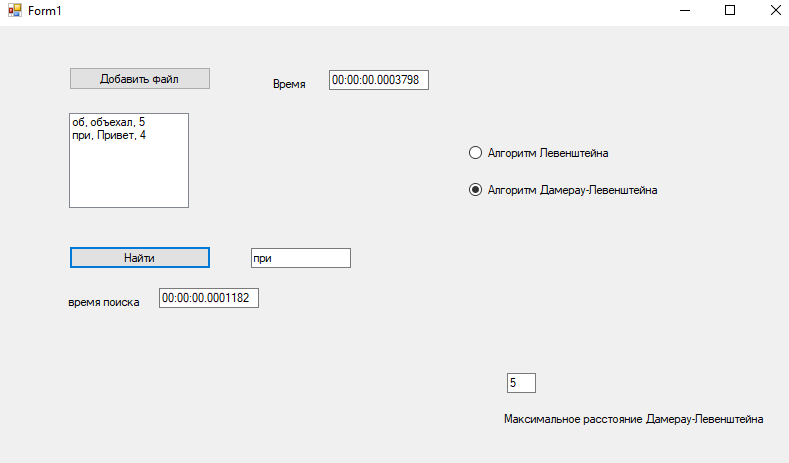
}

}

}

**Тест программы**





**Cсылка на репозиторий исходных кодов GitHub**

https://github.com/bogdanpylaev/BCIT\_Bogdan\_Pylaev