Державний вищий навчальний заклад

«Прикарпатський національний університет імені Василя Стефаника» Кафедра комп’ютерних наук та інформаційних систем

**ЛАБОРАТОРНА РОБОТА №4**

з предмету «Алгоритми і структури даних»

Тема: «Алгоритми внутрішнього сортування»

Виконав:

студент групи КН-2

Гриньків В.І.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2020р.

к.т.н., доц. Никорак Я.Я.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2020р.

Івано-Франківськ – 2020

Мета: Вивчення основних методів сортування

Код виконання програми:

using System;

using System.ComponentModel;

using System.Windows.Forms;

using System.Diagnostics;

namespace Laba4

{

public partial class Form1 : Form

{

Sort[] sort = new Sort[0];

public bool firstIteraton = false;

public int SpaceInArraySort = 0;

public int ArrayLenght;

public int MinForRandom;

public int MaxForRandom;

public int NumberIteration;

public double SerTimeForBuble;

public double TimeBuble;

public double SerTimeForMinElems;

public double TimeMinEL;

public double SerTimeForInsert;

public double TimeInsert;

public double SerTimeForShell;

public double TimeShell;

public double SerTimeForXoar;

public double TimeXoar;

public double SerTimeForMerger;

public double TimeMerger;

class Sort

{

public string Тип { get; set; }

public int Розмір { get; set; }

public double Час { get; set; }

public double Середній\_час { get; set; }

public int Повторення { get; set; }

public Sort(double sertimesort, string typesort, int capacity, int numberiteration, double timesort)

{

Тип = typesort;

Розмір = capacity;

Середній\_час = sertimesort;

Повторення = numberiteration;

Час = timesort;

}

}

public Form1()

{

InitializeComponent();

}

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

{

}

private void button1\_Click(object sender, EventArgs e)

{

ArrayLenght = Convert.ToInt32(textBox1.Text);

MinForRandom = Convert.ToInt32(textBox2.Text);

MaxForRandom = Convert.ToInt32(textBox3.Text);

NumberIteration = Convert.ToInt32(textBox4.Text);

Array.Resize(ref sort, 0);

SpaceInArraySort = 0;

dataGridView1.Visible = true;

dataGridView1.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells;

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

int[] array = new int[ArrayLenght];

int[] arrayForSwap = new int[ArrayLenght];

dataGridView1.Rows.Clear();

RandomArray(array, arrayForSwap);

for (int i = 0; i < NumberIteration; i++)

{

SwapArray(array, arrayForSwap);

SortByBuble(array);

if (i == NumberIteration - 1)

{

Array.Resize(ref sort, sort.Length + 1);

SerTimeForBuble /= NumberIteration;

Sort SortbyBuble = new Sort(SerTimeForBuble, "Метод обміну", array.Length, NumberIteration, TimeBuble);

SpaceInArraySort += 1;

sort[SpaceInArraySort - 1] = SortbyBuble;

InTable(sort);

}

}

firstIteraton = false;

for (int i = 0; i < NumberIteration; i++)

{

SwapArray(array, arrayForSwap);

SortByInsertion(array);

if (i == NumberIteration - 1)

{

Array.Resize(ref sort, sort.Length + 1);

SerTimeForInsert /= NumberIteration;

SpaceInArraySort += 1;

Sort SortbyInsertion = new Sort(SerTimeForInsert, "Метод вставок", array.Length, NumberIteration, TimeInsert);

sort[SpaceInArraySort - 1] = SortbyInsertion;

InTable(sort);

}

}

firstIteraton = false;

for (int i = 0; i < NumberIteration; i++)

{

SwapArray(array, arrayForSwap);

SortByShell(array);

if (i == NumberIteration - 1)

{

Array.Resize(ref sort, sort.Length + 1);

SerTimeForShell /= NumberIteration;

SpaceInArraySort += 1;

Sort SortbyShell = new Sort(SerTimeForShell, "Метод Шелла", array.Length, NumberIteration, TimeShell);

sort[SpaceInArraySort - 1] = SortbyShell;

InTable(sort);

}

}

firstIteraton = false;

for (int i = 0; i < NumberIteration; i++)

{

SwapArray(array, arrayForSwap);

SortByMinElems(array);

if (i == NumberIteration - 1)

{

Array.Resize(ref sort, sort.Length + 1);

SerTimeForMinElems /= NumberIteration;

SpaceInArraySort += 1;

Sort SortByMinElems = new Sort(SerTimeForMinElems, "Метод мінімальних елементів", array.Length, NumberIteration, TimeMinEL);

sort[SpaceInArraySort - 1] = SortByMinElems;

InTable(sort);

}

}

firstIteraton = false;

for (int i = 0; i < NumberIteration; i++)

{

SwapArray(array, arrayForSwap);

SortByXoar(array);

if (i == NumberIteration - 1)

{

Array.Resize(ref sort, sort.Length + 1);

SerTimeForXoar /= NumberIteration;

SpaceInArraySort += 1;

Sort SortbyXoar = new Sort(SerTimeForXoar, "Метод Хоара", array.Length, NumberIteration, TimeXoar);

sort[SpaceInArraySort - 1] = SortbyXoar;

InTable(sort);

}

}

firstIteraton = false;

for (int i = 0; i < NumberIteration; i++)

{

SwapArray(array, arrayForSwap);

SortByMerger(array);

if (i == NumberIteration - 1)

{

Array.Resize(ref sort, sort.Length + 1);

SerTimeForMerger /= NumberIteration;

SpaceInArraySort += 1;

Sort SortByMerger = new Sort(SerTimeForMerger, "Метод злиття", array.Length, NumberIteration, TimeMerger);

sort[SpaceInArraySort - 1] = SortByMerger;

InTable(sort);

}

}

}

private void RandomArray(int[] array, int[] arrayForSwap)

{

Random random = new Random();

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

{

array[i] = random.Next(MinForRandom, MaxForRandom);

arrayForSwap[i] = array[i];

}

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

{

richTextBox1.Text += array[i].ToString() + " ";

}

richTextBox1.Text += "|||||||||";

}

private void SwapArray(int[] array, int[] arrayForSwap)

{

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

{

array[i] = arrayForSwap[i];

}

}

private void InTable(Sort[] sort)

{

BindingList<Sort> data = new BindingList<Sort>();

dataGridView1.DataSource = data;

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

{

data.Add(sort[i]);

}

}

private void SortByBuble(int[] array)

{

string TIMESORT;

double timesort;

var stopWatch = Stopwatch.StartNew();

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

{

for (int j = i + 1; j < array.Length; j++)

{

if (array[i] > array[j])

{

Swap(ref array[i], ref array[j]);

}

}

}

stopWatch.Stop();

TIMESORT = stopWatch.ElapsedTicks.ToString();

timesort = double.Parse(TIMESORT);

if (firstIteraton == false) { TimeBuble = timesort; firstIteraton = true; }

SerTimeForBuble += timesort;

}

private void SortByMinElems(int[] array)

{

string TIMESORT;

double timesort;

int k;

var stopWatch = Stopwatch.StartNew();

for (int i = 0; i < array.Length - 1; i++)

{

double min = array[i + 1];

int k1 = i + 1;

for (int first = (i + 1) + 1; first < array.Length; first++)

{

if (array[first] < min)

{

min = array[first];

k1 = first;

}

}

k = k1;

if (array[i] > array[k])

{

Swap(ref array[i], ref array[k]);

}

}

stopWatch.Stop();

TIMESORT = stopWatch.ElapsedTicks.ToString();

timesort = double.Parse(TIMESORT);

if (firstIteraton == false) { TimeMinEL = timesort; firstIteraton = true; }

SerTimeForMinElems += timesort;

}

private void SortByInsertion(int[] array)

{

int newElement, location;

string TIMESORT;

double timesort;

var stopWatch = Stopwatch.StartNew();

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

{

newElement = array[i];

location = i - 1;

while (location >= 0 && array[location] > newElement)

{

array[location + 1] = array[location];

location = location - 1;

}

array[location + 1] = newElement;

}

stopWatch.Stop();

TIMESORT = stopWatch.ElapsedTicks.ToString();

timesort = double.Parse(TIMESORT);

if (firstIteraton == false) { TimeInsert = timesort; firstIteraton = true; }

SerTimeForInsert += timesort;

}

private void SortByShell(int[] array)

{

string TIMESORT;

double timesort;

int i, j, step;

int tmp;

var stopWatch = Stopwatch.StartNew();

for (step = array.Length / 2; step > 0; step /= 2)

for (i = step; i < array.Length; i++)

{

tmp = array[i];

for (j = i; j >= step; j -= step)

{

if (tmp < array[j - step])

array[j] = array[j - step];

else

break;

}

array[j] = tmp;

}

stopWatch.Stop();

TIMESORT = stopWatch.ElapsedTicks.ToString();

timesort = double.Parse(TIMESORT);

if (firstIteraton == false) { TimeShell = timesort; firstIteraton = true; }

SerTimeForShell += timesort;

}

private void SortByXoar(int[] array)

{

string TIMESORT;

double timesort;

var stopWatch = Stopwatch.StartNew();

QuickSort(array, 0, array.Length - 1);

stopWatch.Stop();

TIMESORT = stopWatch.ElapsedTicks.ToString();

timesort = double.Parse(TIMESORT);

if (firstIteraton == false) { TimeXoar = timesort; firstIteraton = true; }

SerTimeForXoar += timesort;

}

private void QuickSort(int[] array, int first, int last)

{

if (first < last)

{

int left = first, right = last, middle = array[(left + right) / 2];

do

{

while (array[left] < middle) left++;

while (array[right] > middle) right--;

if (left <= right)

{

Swap(ref array[left], ref array[right]);

left++;

right--;

}

} while (left <= right);

QuickSort(array, first, right);

QuickSort(array, left, last);

}

}

private void SortByMerger(int[] array)

{

string TIMESORT;

double timesort;

var stopWatch = Stopwatch.StartNew();

Merger(array, 0, array.Length - 1);

stopWatch.Stop();

TIMESORT = stopWatch.ElapsedTicks.ToString();

timesort = double.Parse(TIMESORT);

if (firstIteraton == false) { TimeMerger = timesort; firstIteraton = true; }

SerTimeForMerger += timesort;

}

private void Merger(int[] array, int first, int last)

{

if (last == first)

return;

if (last - first == 1)

{

if (array[last] < array[first])

Swap(ref array[last], ref array[first]);

return;

}

int m = (last + first) / 2;

Merger(array, first, m);

Merger(array, m + 1, last);

int[] arrayClon = new int[array.Length];

int xl = first;

int xr = m + 1;

int cur = 0;

while (last - first + 1 != cur)

{

if (xl > m)

arrayClon[cur++] = array[xr++];

else if (xr > last)

arrayClon[cur++] = array[xl++];

else if (array[xl] > array[xr])

arrayClon[cur++] = array[xr++];

else arrayClon[cur++] = array[xl++];

}

for (int i = 0; i < cur; i++)

array[i + first] = arrayClon[i];

}

static void Swap<T>(ref T a, ref T b)

{

T c = a;

a = b;

b = c;

}

}

}

Результат виконання:

