**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

**КНІТ**

Кафедра **ПЗ**

### ЗВІТ

До лабораторної роботи № 5

**З дисципліни:** *“Алгоритми та структури даних”*

**На тему:** *“Метод сортування злиттям”*

**Лектор:**

доц. каф. ПЗ

Коротєєва Т.О.

**Виконала:**

ст. групи ПЗ-26

Матолінець Л. А.

**Прийняв:**

асистент каф. ПЗ

Симець І.І.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2022 р.

∑= \_\_\_\_\_ .

Львів – 2022

**Тема роботи:** Метод сортування злиттям**.**

**Мета роботи**:  Вивчити [алгоритм](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=472&eid=19102&displayformat=dictionary" \o "Глосарій: Алгоритм) сортування злиттям. Здійснити програмну реалізацію [алгоритм](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=472&eid=19102&displayformat=dictionary" \o "Глосарій: Алгоритм)у сортування злиттям. Дослідити швидкодію [алгоритм](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=472&eid=19102&displayformat=dictionary" \o "Глосарій: Алгоритм)у сортування злиттям.

**TЕОРЕТИЧНІ ВІДОМОСТІ**

***Покроковий опис роботи алгоритму сортування злиттям.***

**Алгоритм MS** Задано одновимірний масив елементів *R1 ,R2 ,…,Rn.* Даний алгоритм реорганізує масив у висхідному порядку таким чином, що ділить навпіл заданий масив доти, поки підмасиви не булуть мати 1 елемент.

MS1. Початок рекурсії. Оголошення масиву, крайнього лівого та правого елемента, медіани.

MS2. Розбиття масиву навпіл. Перевірка, чи крайній лівий елемент менший за крайній правий.

MS3. Розбиття та впорядкування підмасивів, довжина яких >= 1.

MS4. Повторювання кроку MS2, поки довжина підмасиву <= 1.

MS5. Злиття двох ківнцевих посортованих підмасивів у 1 результуючий масив.

MS6. Кінець рекурсії.

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

**Варіант 5**

Задано одномірний масив дійсних чисел. Виключити з нього моду (елемент, який повторюється найчастіше). Отриманий масив посортувати в порядку зростання.

**ХІД ВИКОНАННЯ**

Метод сортування злиттям:

using System;

using System.Diagnostics;

using System.Threading;

class SortProgram {

public static void merge(int[] arr, int leftElem, int middleElem, int rightElem)

{

int subArray1Length = middleElem - leftElem + 1;

int subArray2Length = rightElem - middleElem;

int[] leftSubarray = new int[subArray1Length];

int[] rightSubarray = new int[subArray2Length];

int i, j;

for (i = 0; i < subArray1Length; i++)

leftSubarray[i] = arr[leftElem + i];

for (j = 0; j < subArray2Length; ++j)

rightSubarray[j] = arr[middleElem + 1 + j];

i = 0;

j = 0;

int k = leftElem;

while (i < subArray1Length && j < subArray2Length) {

if (leftSubarray[i] <= rightSubarray[j]) {

arr[k] = leftSubarray[i];

i++;

}

else {

arr[k] = rightSubarray[j];

j++;

}

k++;

}

while (i < subArray1Length) {

arr[k] = leftSubarray[i];

i++;

k++;

}

while (j < subArray2Length) {

arr[k] = rightSubarray[j];

j++;

k++;

}

}

public static void mergeSort(int[] arr, int leftElem, int rightElem)

{

if (leftElem < rightElem) {

int middleElem = leftElem+ (rightElem-leftElem)/2;

mergeSort (arr, leftElem,middleElem );

mergeSort (arr, middleElem + 1, rightElem);

merge(arr, leftElem, middleElem, rightElem);

}

}

public static void Main(String[] args)

{

int Min = -50000;

int Max = 50000;

Console.WriteLine("Enter the size of your array: ");

int length = Convert.ToInt32(Console.ReadLine());

int[] arr = new int[length];

Random randNum = new Random();

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

{

arr[i] = randNum.Next(Min, Max);

}

Stopwatch stopwatch = new Stopwatch();

stopwatch.Start();

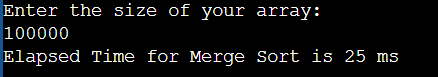
mergeSort (arr, 0, arr.Length - 1);

stopwatch.Stop();

Console.WriteLine("Elapsed Time for Merge Sort is {0} ms", stopwatch.ElapsedMilliseconds);

}

}



*Рис.1 Результат виконання програми*

Вихідний код виконання індивідуального завдання:

//myArray.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Documents;

namespace SortingApp

{

public class MyArray

{

protected double[] array = new double[0];

public MyArray() {

}

public double[] getList() { return array; }

public void addElement(double item)

{

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

array[array.Length - 1] = item;

}

public void setArray(double[] arr)

{

this.array = arr;

}

public double Mode(double[] list)

{

double mode = default(double);

if (list != null && list.Count() > 0)

{

Dictionary<double, double> counts = new Dictionary<double, double>();

foreach (double element in list)

{

if (counts.ContainsKey(element))

counts[element]++;

else

counts.Add(element, 1);

}

double max = 0;

foreach (KeyValuePair<double, double> count in counts)

{

if (count.Value > max)

{

// Update the mode

mode = count.Key;

max = count.Value;

}

}

}

return mode;

}

}

}

//MainWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Numerics;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

namespace SortingApp

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

///

public class MyArrayOfArrays

{

protected MyArray[] array = new MyArray[0];

public MyArrayOfArrays() { }

public MyArray[] getList() { return array; }

public void addElement(MyArray item)

{

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

array[array.Length - 1] = item;

}

public void setArray(MyArray[] arr)

{

array = arr;

}

}

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private static void merge(double[] arr, int leftElem, int middleElem, int rightElem)

{

int subArray1Length = middleElem - leftElem + 1;

int subArray2Length = rightElem - middleElem;

double[] leftSubarray = new double[subArray1Length];

double[] rightSubarray = new double[subArray2Length];

int i, j;

for (i = 0; i < subArray1Length; ++i)

leftSubarray[i] = arr[leftElem + i];

for (j = 0; j < subArray2Length; ++j)

rightSubarray[j] = arr[middleElem + 1 + j];

i = 0;

j = 0;

int k = leftElem;

while (i < subArray1Length && j < subArray2Length)

{

if (leftSubarray[i] <= rightSubarray[j])

{

arr[k] = leftSubarray[i];

i++;

}

else

{

arr[k] = rightSubarray[j];

j++;

}

k++;

}

while (i < subArray1Length)

{

arr[k] = leftSubarray[i];

i++;

k++;

}

while (j < subArray2Length)

{

arr[k] = rightSubarray[j];

j++;

k++;

}

}

private static double[] sort(double[] arr, int leftElem, int rightElem)

{

if (leftElem < rightElem)

{

int middleElem = leftElem + (rightElem - leftElem) / 2;

sort(arr, leftElem, middleElem);

sort(arr, middleElem + 1, rightElem);

merge(arr, leftElem, middleElem, rightElem);

}

return arr;

}

MyArray array = new MyArray();

private void addArray\_Click(object sender, RoutedEventArgs e)

{

String arrField = column\_field.Text;

String[] strings = arrField.Split(" ; ");

foreach (var item in strings)

{

double x = Convert.ToDouble(item);

array.addElement(x);

}

double[] array1 = array.getList();

double elementToDelete = array.Mode(array.getList());

double [] numbers = array.getList();

numbers = numbers.Where(val => val != elementToDelete).ToArray();

array.setArray(numbers);

}

private void showArray\_Click(object sender, RoutedEventArgs e)

{

double[] sortedArray = array.getList();

String result = "";

foreach (var item in sortedArray)

{

result += Convert.ToString(Math.Round(item, 1));

result += " ; ";

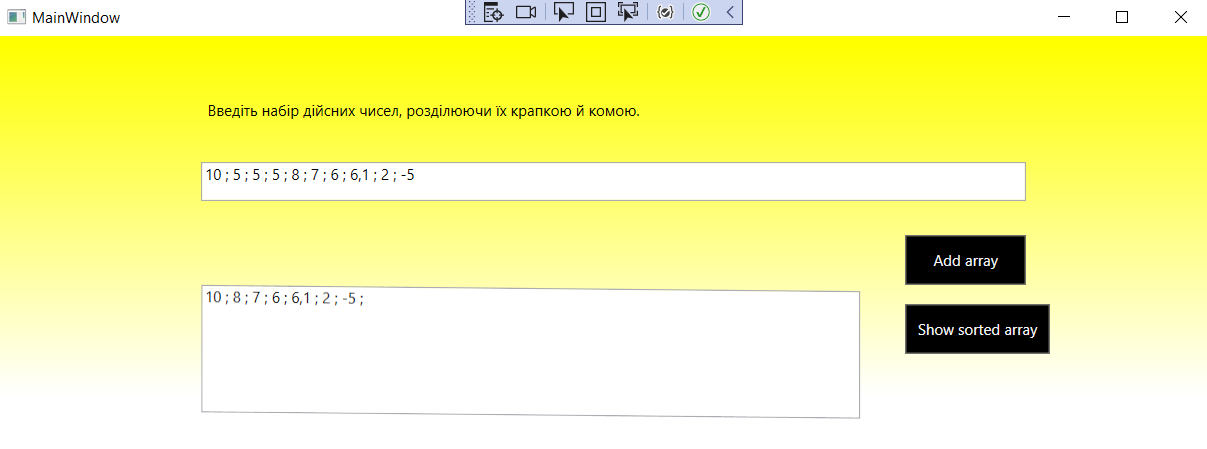
}

showedArray.Text = result;

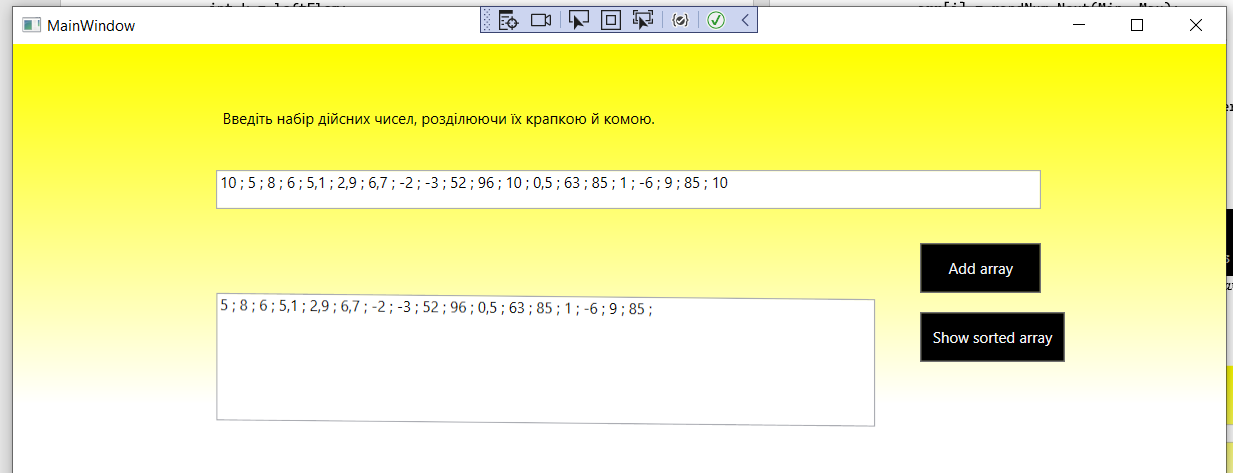
}

}

}



*Рис. 1 Результат сортування масиву з 10ти елементів*



*Рис. 2 Результат сортування масиву з 20ти елементів*

**Висновок**

На цій лабораторній роботі я познайомилася з алгоритмом сортування злиттям та виконала індивідуальне завдання – сортування дійсних чисел, видаляючи моду з заданого масиву. Також я реалізувала функції для виконання сортування, де O(n\*log(n)).