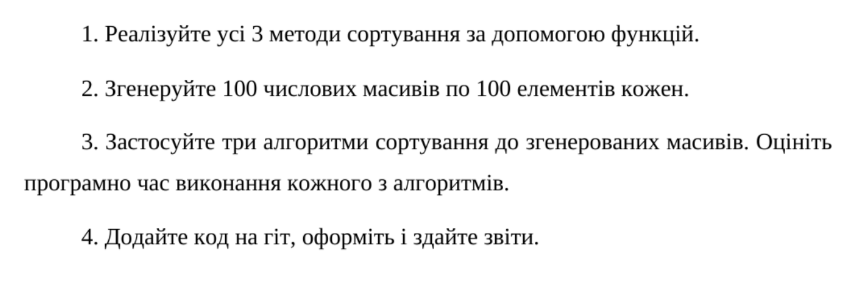
**ЗВІТ**

про виконання лабораторної роботи 4-5

«**Задачі на сортування**»  
з дисципліни  
«Методика розв'язування олімпіадних задач» студента(ки) групи ІН-2327Б  
Чухрая Олександра Васильовича

**Умова завдання:**

****

**Виконання:**

using System;

using System.Diagnostics;

using System.Linq;

class Program

{

static void Main()

{

int arrayCount = 100;

int arraySize = 100;

Random random = new Random();

int[][] arrays = new int[arrayCount][];

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

{

arrays[i] = Enumerable.Range(0, arraySize).Select(\_ => random.Next(1000)).ToArray();

}

MeasureSortingTime("Bubble Sort", BubbleSort, arrays);

MeasureSortingTime("Quick Sort", QuickSort, arrays);

MeasureSortingTime("Tree Sort", TreeSort, arrays);

Console.ReadLine(); // Затримка перед закриттям консолі

}

static void MeasureSortingTime(string name, Action<int[]> sortMethod, int[][] arrays)

{

Stopwatch stopwatch = new Stopwatch();

int[][] arraysCopy = arrays.Select(a => a.ToArray()).ToArray();

Console.WriteLine($"Starting {name}...");

stopwatch.Start();

foreach (var array in arraysCopy)

{

sortMethod(array);

}

stopwatch.Stop();

Console.WriteLine($"{name} took {stopwatch.ElapsedMilliseconds} ms");

}

static void BubbleSort(int[] array)

{

int n = array.Length;

for (int i = 0; i < n - 1; i++)

{

for (int j = 0; j < n - i - 1; j++)

{

if (array[j] > array[j + 1])

{

(array[j], array[j + 1]) = (array[j + 1], array[j]);

}

}

}

}

static void QuickSort(int[] array) => QuickSortHelper(array, 0, array.Length - 1);

static void QuickSortHelper(int[] array, int left, int right)

{

if (left < right)

{

int pivot = Partition(array, left, right);

QuickSortHelper(array, left, pivot - 1);

QuickSortHelper(array, pivot + 1, right);

}

}

static int Partition(int[] array, int left, int right)

{

int pivot = array[right];

int i = left - 1;

for (int j = left; j < right; j++)

{

if (array[j] < pivot)

{

i++;

(array[i], array[j]) = (array[j], array[i]);

}

}

(array[i + 1], array[right]) = (array[right], array[i + 1]);

return i + 1;

}

static void TreeSort(int[] array)

{

BinaryTree tree = new BinaryTree();

foreach (var item in array)

{

tree.Insert(item);

}

int index = 0;

tree.InOrderTraversal(array, ref index);

}

}

class TreeNode

{

public int Value;

public TreeNode Left, Right;

public TreeNode(int value) { Value = value; }

}

class BinaryTree

{

private TreeNode root;

public void Insert(int value)

{

root = InsertRec(root, value);

}

private TreeNode InsertRec(TreeNode root, int value)

{

if (root == null)

{

return new TreeNode(value);

}

if (value < root.Value) root.Left = InsertRec(root.Left, value);

else root.Right = InsertRec(root.Right, value);

return root;

}

public void InOrderTraversal(int[] array, ref int index)

{

InOrderTraversalRec(root, array, ref index);

}

private void InOrderTraversalRec(TreeNode root, int[] array, ref int index)

{

if (root != null)

{

InOrderTraversalRec(root.Left, array, ref index);

array[index++] = root.Value;

InOrderTraversalRec(root.Right, array, ref index);

}

}

}

Результат:

