МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №2**

**з дисципліни «Алгоритми та структури даних»**

**на тему: «Аналіз алгоритмів»**

Виконав: студент гр. ПЗ1911

Сіньков Г.О.

Прийняла: ас. каф. КІТ

Куроп’ятник О. С.

Дніпро, 2020

**Лабораторна робота №2**

**Тема.** Аналіз алгоритмів.

**Мета.** Ознайомитися з поняттям з методом безпосередньої оцінки обчислювальної складності алгоритмів. Отримати практичні навички визначення та аналізу оцінки обчислювальної складності алгоритмів.

**Постановка задачі**

Напишіть програму мовою java для визначення обчислювальної складності алгоритмів сортування бульбашкою, швидкого сортування та методу відповідно до індивідуального завдання. Проведіть експеримент за схемою, описаною в теоретичних відомостях. В ході роботи програма має записувати середні значення обчислювальної складності у файл. За отриманими значеннями побудуйте графіки залежності обчислювальної складності в залежності від кількості (розміру) вхідних даних. Визначте порядок обчислювальної складності.

Індивідуальне завдання видається викладачем на занятті:

7. Сортування вибором

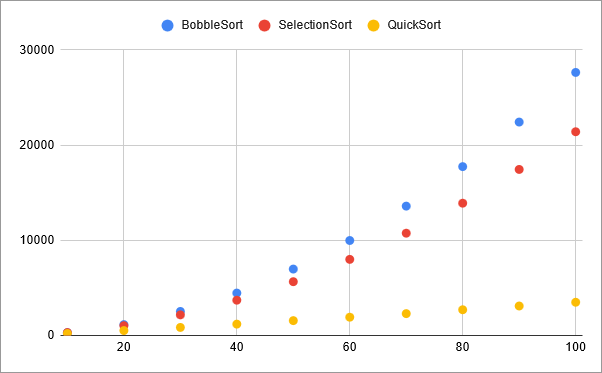
**Текст програми**

ArraySort.java

package com.company;  
  
import java.util.Random;  
import java.util.Arrays;  
  
public class ArraySort {  
  
 private int[] array;  
 private int size;  
 private final Random RANDOM = new Random();  
  
 public ArraySort(int size) {  
 this.size = size;  
 array = new int[this.size];  
 }  
   
 public void randFilling(int min, int max) {  
 for (int i = 0; i < size; i++) {  
 array[i] = RANDOM.nextInt(max - min) + min;  
 }  
  
 }  
  
 //cmp jmp mov  
 public int bubbleSort() {  
 int total = 0;  
 total++;//=  
 for (int i = 0; i < size - 1; i++) {  
 total++;//<  
 total++;//=  
 for (int j = 0; j < size - 1 - i; j++) {  
 total++;//<  
 total++;//if  
 if (array[j] > array[j + 1]) {  
 int tmp = array[j];  
 array[j] = array[j + 1];  
 array[j + 1] = tmp;  
 total += 3;  
 }  
 total++;//++  
 total++;//jmp  
 }  
 total++;//<  
  
 total++;//++  
 total++;//jmp  
 }  
 total++;//<  
 return total;  
 }  
  
 public int quickSort(int total\_tmp,int low, int high){  
 //int total = 0;  
 total\_tmp++;//=  
 if(array.length == 0)  
 return total\_tmp;  
  
 total\_tmp++;  
 if(low >= high)  
 return total\_tmp;  
  
 // выбрать опорный элемент  
 int middle = low + (high - low) / 2;  
 total\_tmp++;  
 int opora = array[middle];  
 total\_tmp++;  
  
 // разделить на подмассивы, который больше и меньше опорного элемента  
 int i = low, j = high;  
 total\_tmp += 2;  
 while (i <= j) {  
 total\_tmp++;  
 while (array[i] < opora) {  
 total\_tmp++;  
 i++;  
 total\_tmp++;  
 }  
  
 total\_tmp++;  
 while (array[j] > opora) {  
 total\_tmp++;  
 j--;  
 total\_tmp++;  
 }  
 total\_tmp++;  
  
 total\_tmp++;  
 if (i <= j) {//меняем местами  
 int temp = array[i];  
 array[i] = array[j];  
 array[j] = temp;  
 i++;  
 j--;  
 total\_tmp += 5;  
 }  
 }  
 total\_tmp++;  
  
  
 // вызов рекурсии для сортировки левой и правой части  
 total\_tmp++;  
 if (low < j) {  
 total\_tmp++;  
 total\_tmp = quickSort(total\_tmp, low, j);  
 }  
  
 total\_tmp++;  
 if (high > i) {  
 total\_tmp++;  
 total\_tmp = quickSort(total\_tmp, i, high);  
 }  
  
 return total\_tmp;  
 }  
  
 public int selectionSort(){  
 int total = 1;  
 for (int i = 0; i < array.length; i++) { // i - номер текущего шага  
 total += 2;  
 int pos = i;  
 total++;  
 int min = array[i];  
 total += 2;  
 for (int j = i + 1; j < array.length; j++) {// цикл выбора наименьшего элемента  
 total += 2;  
 if (array[j] < min) {  
 pos = j; // pos - индекс наименьшего элемента  
 min = array[j];  
 total += 2;  
 }  
 total += 2;  
 }  
 array[pos] = array[i];  
 array[i] = min; // меняем местами наименьший с array[i]  
 total += 5;  
 }  
 total++;  
  
 return total;  
 }  
}

Main.java

package com.company;  
  
import java.io.FileWriter;  
import java.io.IOException;  
import java.util.Scanner;  
  
public class Main {  
  
 public static void main(String[] args) throws IOException {  
 // write your code here  
  
 try (FileWriter writer = new FileWriter("bubbleSortResult.txt", true)) {  
 final int MINSIZE = 10;  
 final int MAXSIZE = 100;  
 final int AMOUNT = 100;  
 int total = 0;  
 for (int i = MINSIZE; i <= MAXSIZE; i += 10) {  
 ArraySort arrayBubbleSort = new ArraySort(i);  
 for (int j = 0; j < AMOUNT; j++) {  
 arrayBubbleSort.randFilling(1, 100);  
 total = total + arrayBubbleSort.bubbleSort();  
 }  
 int avg = total / AMOUNT;  
 System.*out*.println(i + ": " + avg);  
 writer.write(avg + "\n");  
 total = 0;  
  
 }  
 }  
  
 try (FileWriter writer = new FileWriter("QuickSortResult.txt", true)) {  
 final int MINSIZE = 10;  
 final int MAXSIZE = 100;  
 final int AMOUNT = 100;  
 int total = 0;  
 int total\_tmp = 0;  
 int low = 0;  
 for (int i = MINSIZE; i <= MAXSIZE; i += 10) {  
 int high = i - 1;  
 ArraySort arrayQuickSort = new ArraySort(i);  
 for (int j = 0; j < AMOUNT; j++) {  
 arrayQuickSort.randFilling(1, 100);  
 total = total + arrayQuickSort.quickSort(total\_tmp, low, high);  
 }  
 int avg = total / AMOUNT;  
 System.*out*.println(i + ": " + avg);  
 writer.write(avg + "\n");  
 total = 0;  
  
 }  
 }  
  
 try (FileWriter writer = new FileWriter("SelectionSortResult.txt", true)) {  
 final int MINSIZE = 10;  
 final int MAXSIZE = 100;  
 final int AMOUNT = 100;  
 int total = 0;  
 for (int i = MINSIZE; i <= MAXSIZE; i += 10) {  
 ArraySort arraySelectionSort = new ArraySort(i);  
 for (int j = 0; j < AMOUNT; j++) {  
 arraySelectionSort.randFilling(1, 100);  
 total = total + arraySelectionSort.selectionSort();  
 }  
 int avg = total / AMOUNT;  
 System.*out*.println(i + ": " + avg);  
 writer.write(avg + "\n");  
 total = 0;  
  
 }  
 }  
  
 }  
}



Висновок: виконавши 2 лабораторну роботу я з'ясував, яке з трьох сортувань(Bobble, Selection, Quick) найшвидше. З діаграми ми можемо побачити на скільки QuickSort сортує найшвидше за два інші сортування. BobbleSort є найпростішим сортування, який підходить лише для маленьких масивів. Він є найдовшим за всі інші сортування. SelectionSort більш швидшим за BobbleSort, хоча все одно досить повільний за QuickSort.

Порядок складності:

-BobbleSort (О(*n*2))

-SelectionSort (О(*n*2))

-QuickSort (O(n log n))