Trabalho 1 - Ordenacao

```
package algoritmos.SelectionSort;

import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Random;
import java.util.Scanner;

public class CinquentaAgora {
    private static int comparisons = 0;
    private static int trocas_quick = 0;
    public static void main(String[] args) {
        int[] lista_500 = generateRandomArray(500, 1, 500);
        int[] lista_5000 = generateRandomArray(5000, 1, 100000);
        int[] lista_com_50000 = generateRandomArray(50000, 1, 100000);

        Scanner sc = new Scanner(System.in);
        System.out.println("VAMOS COMECAR COM OS ALGORITMOS ");
        String n = "";
        int numero, quantidadeListas, descAsc;
        long startTime;
        long endTime;
        long executionTime;
        double bigO;



        while (!n.equals("n")){
            System.out.println("Qual ALGORITMO voce quer ?");
            System.out.println("1. Selection Sort");
            System.out.println("2. Insertion Sort");
            System.out.println("3. Bubble Sort");
            System.out.println("4. Quick Sort");
            numero = sc.nextInt();

            switch (numero){
                case 1:
                    System.out.println("Escolhe a quantidade de numeros aleatorios que desejas");
                    System.out.println("1. 500");
                    System.out.println("2. 5000");
                    System.out.println("3. 50_000");
                    quantidadeListas = sc.nextInt();

                    System.out.println("Gostaria de ordenar por Ordem Crescente ou decrescente?");
                    System.out.println("1. Crescente");
                    System.out.println("2. Decrescente");
                    descAsc= sc.nextInt();

                    switch (quantidadeListas){
                        case 1:
                            startTime = System.nanoTime();
                            int num_trocas[] = selectionSort(lista_500,descAsc == 1 ? "asc" : "desc");
                            endTime = System.nanoTime();

                            executionTime = endTime - startTime;
                            bigO = Math.pow(lista_500.length,2);
                            ArrayList<String> dados = new ArrayList<String>();
                            System.out.println("ALGORITMO SELECTION SORT ");
                            dados.add("ALGORITMO SELECTION SORT ");
                            dados.add("O tempo em complexidade do Algorimto " + bigO);
                            dados.add("Tempo de execução em Segundos " + executionTime / 1e9);
                            dados.add("Numeros de Trocas " + num_trocas[0]);
```

```
                dados.add("Numeros de comparacoes " + num_trocas[1]);
                WriteToFileExample(dados);

                System.out.println("O tempo em complexidade do Algorimto " + bigO);
                System.out.println("Tempo de execução em Segundos " + executionTime / 1e9);
                System.out.println("Numeros de Trocas " + num_trocas[0]);
                System.out.println("Numeros de comparacoes " + num_trocas[1]);
                break;
            case 2:
                startTime = System.nanoTime();
                int num_trocas_2[] = selectionSort(lista_5000,descAsc == 1 ? "asc" : "desc");
                endTime = System.nanoTime();

                executionTime = endTime - startTime;
                bigO = Math.pow(lista_5000.length,2);
                ArrayList<String> dados_2 = new ArrayList<String>();
                dados_2.add("ALGORITMO SELECTION SORT ");
                dados_2.add("O tempo em complexidade do Algorimto " + bigO);
                dados_2.add("Tempo de execução em Segundos " + executionTime / 1e9);
                dados_2.add("Numeros de Trocas " + num_trocas_2[0]);
                dados_2.add("Numeros de comparacoes " + num_trocas_2[1]);
                WriteToFileExample(dados_2);

                System.out.println("O tempo em complexidade do Algorimto " + bigO);
                System.out.println("Tempo de execução em Segundos " + executionTime / 1e9);
                System.out.println("Numeros de Trocas " + num_trocas_2[0]);
                System.out.println("Numeros de comparacoes " + num_trocas_2[1]);
                break;
            case 3:
                startTime = System.nanoTime();
                int num_trocas_3[] = selectionSort(lista_com_50000,descAsc == 1 ? "asc" : "desc");
                endTime = System.nanoTime();

                executionTime = endTime - startTime;
                bigO = Math.pow(lista_com_50000.length,2);
                ArrayList<String> dados_3 = new ArrayList<String>();
                dados_3.add("ALGORITMO SELECTION SORT ");
                dados_3.add("O tempo em complexidade do Algorimto " + bigO);
                dados_3.add("Tempo de execução em Segundos " + executionTime / 1e9);
                dados_3.add("Numeros de Trocas " + num_trocas_3[0]);
                dados_3.add("Numeros de comparacoes " + num_trocas_3[1]);
                WriteToFileExample(dados_3);

                System.out.println("O tempo em complexidade do Algorimto " + bigO);
                System.out.println("Tempo de execução em Segundos " + executionTime / 1e9);
                System.out.println("Numeros de Trocas " + num_trocas_3[0]);
                System.out.println("Numeros de comparacoes " + num_trocas_3[1]);
                break;
        }

        break;
    case 2:
        System.out.println("Escolhe a quantidade de numeros aleatorios que desejas");
        System.out.println("1. 500");
        System.out.println("2. 5000");
        System.out.println("3. 50_000");
        quantidadeListas = sc.nextInt();

        System.out.println("Gostaria de ordenar por Ordem Crescente ou decrescente?");
        System.out.println("1. Crescente");
        System.out.println("2. Decrescente");
        descAsc= sc.nextInt();

        switch (quantidadeListas){
            case 1:
                startTime = System.nanoTime();
```

```
                startTime = System.nanoTime();
                int num_trocas[] = insertionSort(lista_500,descAsc == 1 ? "asc" : "desc");
                endTime = System.nanoTime();

                executionTime = endTime - startTime;
                bigO = Math.pow(lista_500.length,2);
                ArrayList<String> dados = new ArrayList<String>();
                dados.add("ALGORITMO INSERTIONSORT");
                dados.add("O tempo em complexidade do Algorimto " + bigO);
                dados.add("Tempo de execução em Segundos " + executionTime / 1e9);
                dados.add("Numeros de Trocas " + num_trocas[0]);
                dados.add("Numeros de comparacoes " + num_trocas[1]);
                WriteToFileExample(dados);

                System.out.println("O tempo em complexidade do Algorimto " + bigO);
                System.out.println("Tempo de execução em Segundos " + executionTime / 1e9);
                System.out.println("Numeros de Trocas " + num_trocas[0]);
                System.out.println("Numeros de comparacoes " + num_trocas[1]);
                break;
            case 2:
                startTime = System.nanoTime();
                int num_trocas_2[] = insertionSort(lista_5000,descAsc == 1 ? "asc" : "desc");
                endTime = System.nanoTime();

                executionTime = endTime - startTime;
                bigO = Math.pow(lista_5000.length,2);
                ArrayList<String> dados_2 = new ArrayList<String>();
                dados_2.add("ALGORITMO INSERTIONSORT");
                dados_2.add("O tempo em complexidade do Algorimto " + bigO);
                dados_2.add("Tempo de execução em Segundos " + executionTime / 1e9);
                dados_2.add("Numeros de Trocas " + num_trocas_2[0]);
                dados_2.add("Numeros de comparacoes " + num_trocas_2[1]);
                WriteToFileExample(dados_2);

                System.out.println("O tempo em complexidade do Algorimto " + bigO);
                System.out.println("Tempo de execução em Segundos " + executionTime / 1e9);
                System.out.println("Numeros de Trocas " + num_trocas_2[0]);
                System.out.println("Numeros de comparacoes " + num_trocas_2[1]);
                break;
            case 3:
                startTime = System.nanoTime();
                int num_trocas_3[] = insertionSort(lista_com_50000,descAsc == 1 ? "asc" : "desc");
                endTime = System.nanoTime();

                executionTime = endTime - startTime;
                bigO = Math.pow(lista_com_50000.length,2);
                ArrayList<String> dados_3 = new ArrayList<String>();
                dados_3.add("ALGORITMO INSERTIONSORT");
                dados_3.add("O tempo em complexidade do Algorimto " + bigO);
                dados_3.add("Tempo de execução em Segundos " + executionTime / 1e9);
                dados_3.add("Numeros de Trocas " + num_trocas_3[0]);
                dados_3.add("Numeros de comparacoes " + num_trocas_3[1]);
                WriteToFileExample(dados_3);

                System.out.println("O tempo em complexidade do Algorimto " + bigO);
                System.out.println("Tempo de execução em Segundos " + executionTime / 1e9);
                System.out.println("Numeros de Trocas " + num_trocas_3[0]);
                System.out.println("Numeros de comparacoes " + num_trocas_3[1]);
                break;
        }
        break;

    case 3:
        System.out.println("Escolhe a quantidade de numeros aleatorios que desejas");
        System.out.println("1. 500");
        System.out.println("2. 5000");
```

```
System.out.println("3. 50_000");
quantidadeListas = sc.nextInt();

System.out.println("Gostaria de ordenar por Ordem Crescente ou decrescente?");
System.out.println("1. Crescente");
System.out.println("2. Decrescente");
descAsc= sc.nextInt();

switch (quantidadeListas){
    case 1:
        startTime = System.nanoTime();
        int num_trocas[] = bubbleSort(lista_500,descAsc == 1 ? "asc" : "desc");
        endTime = System.nanoTime();

        executionTime = endTime - startTime;
        bigO = Math.pow(lista_500.length,2);
        ArrayList<String> dados = new ArrayList<String>();
        dados.add("ALGORITMO BubbleSort");
        dados.add("O tempo em complexidade do Algorimto " + bigO);
        dados.add("Tempo de execução em Segundos " + executionTime / 1e9);
        dados.add("Numeros de Trocas " + num_trocas[0]);
        dados.add("Numeros de comparacoes " + num_trocas[1]);
        WriteToFileExample(dados);

        System.out.println("O tempo em complexidade do Algorimto " + bigO);
        System.out.println("Tempo de execução em Segundos " + executionTime / 1e9);
        System.out.println("Numeros de Trocas " + num_trocas[0]);
        System.out.println("Numeros de comparacoes " + num_trocas[1]);
        break;
    case 2:
        startTime = System.nanoTime();
        int num_trocas_2[] = bubbleSort(lista_5000,descAsc == 1 ? "asc" : "desc");
        endTime = System.nanoTime();

        executionTime = endTime - startTime;
        bigO = Math.pow(lista_5000.length,2);
        ArrayList<String> dados_2 = new ArrayList<String>();
        dados_2.add("ALGORITMO BubbleSort");
        dados_2.add("O tempo em complexidade do Algorimto " + bigO);
        dados_2.add("Tempo de execução em Segundos " + executionTime / 1e9);
        dados_2.add("Numeros de Trocas " + num_trocas_2[0]);
        dados_2.add("Numeros de comparacoes " + num_trocas_2[1]);
        WriteToFileExample(dados_2);

        System.out.println("O tempo em complexidade do Algorimto " + bigO);
        System.out.println("Tempo de execução em Segundos " + executionTime / 1e9);
        System.out.println("Numeros de Trocas " + num_trocas_2[0]);
        System.out.println("Numeros de comparacoes " + num_trocas_2[1]);
        break;
    case 3:
        startTime = System.nanoTime();
        int num_trocas_3[] = bubbleSort(lista_com_50000,descAsc == 1 ? "asc" : "desc");
        endTime = System.nanoTime();

        executionTime = endTime - startTime;
        bigO = Math.pow(lista_com_50000.length,2);
        ArrayList<String> dados_3 = new ArrayList<String>();
        dados_3.add("ALGORITMO BubbleSort");
        dados_3.add("O tempo em complexidade do Algorimto " + bigO);
        dados_3.add("Tempo de execução em Segundos " + executionTime / 1e9);
        dados_3.add("Numeros de Trocas " + num_trocas_3[0]);
        dados_3.add("Numeros de comparacoes " + num_trocas_3[1]);
        WriteToFileExample(dados_3);

        System.out.println("O tempo em complexidade do Algorimto " + bigO);
```

```
                System.out.println("Tempo de execuçao em Segundos " + executionTime / 1e9);
                System.out.println("Numeros de Trocas " + num_trocas_3[0]);
                System.out.println("Numeros de comparacoes " + num_trocas_3[1]);
                break;
            }
            break;

        case 4:
            System.out.println("Escolhe a quantidade de numeros aleatorios que desejas");
            System.out.println("1. 500");
            System.out.println("2. 5000");
            System.out.println("3. 50_000");
            quantidadeListas = sc.nextInt();

            switch (quantidadeListas){
                case 1:
                    startTime = System.nanoTime();
                    quickSort(lista_500,0,lista_500.length-1);
                    endTime = System.nanoTime();

                    executionTime = endTime - startTime;
                    bigO = Math.pow(lista_500.length,2);
                    ArrayList<String> dados = new ArrayList<String>();
                    dados.add("ALGORITMO QUICKSORT");
                    dados.add("O tempo em complexidade do Algorimto " + bigO);
                    dados.add("Tempo de execução em Segundos " + executionTime / 1e9);
                    dados.add("Numeros de Trocas " + trocas_quick);
                    dados.add("Numeros de comparacoes " + comparisons);
                    WriteToFileExample(dados);

                    System.out.println("O tempo em complexidade do Algorimto " + bigO);
                    System.out.println("Tempo de execução em Segundos " + executionTime / 1e9);
                    System.out.println("Numeros de Trocas " + trocas_quick);
                    System.out.println("Numeros de comparacoes " + comparisons);
                    break;
                case 2:
                    startTime = System.nanoTime();
                    quickSort(lista_5000,0,lista_5000.length-1);
                    endTime = System.nanoTime();

                    executionTime = endTime - startTime;
                    bigO = Math.pow(lista_5000.length,2);
                    ArrayList<String> dados_2 = new ArrayList<String>();
                    dados_2.add("ALGORITMO QUICKSORT");
                    dados_2.add("O tempo em complexidade do Algorimto " + bigO);
                    dados_2.add("Tempo de execução em Segundos " + executionTime / 1e9);
                    dados_2.add("Numeros de Trocas " + trocas_quick);
                    dados_2.add("Numeros de comparacoes " + comparisons);
                    WriteToFileExample(dados_2);

                    System.out.println("O tempo em complexidade do Algorimto " + bigO);
                    System.out.println("Tempo de execução em Segundos " + executionTime / 1e9);
                    System.out.println("Numeros de Trocas " + trocas_quick);
                    System.out.println("Numeros de comparacoes " + comparisons);
                    break;
                case 3:
                    startTime = System.nanoTime();
                    quickSort(lista_com_50000,0,lista_com_50000.length-1);
                    endTime = System.nanoTime();

                    executionTime = endTime - startTime;
                    bigO = Math.pow(lista_com_50000.length,2);
                    ArrayList<String> dados_3 = new ArrayList<String>();
                    dados_3.add("ALGORITMO QUICKSORT");
                    dados_3.add("O tempo em complexidade do Algorimto " + bigO);
                    dados_3.add("Tempo de execução em Segundos " + executionTime / 1e9);
```

```
                        dados_3.add("Numeros de Trocas " + trocas_quick);
                        dados_3.add("Numeros de comparacoes " + comparisons);
                        WriteToFileExample(dados_3);

                        System.out.println("O tempo em complexidade do Algorimto " + bigO);
                        System.out.println("Tempo de execução em Segundos " + executionTime / 1e9);
                        System.out.println("Numeros de Trocas " +trocas_quick);
                        System.out.println("Numeros de comparacoes " + comparisons);
                        break;
                }
                break;
        }

        System.out.println("Deseja continuar: digite 's' se sim, 'n' senao: ");
        n = sc.next();
    }

}
public  static  int[] bubbleSort(int[] LITO, String flug){
    int comparacoes=0;
    int[] lista = LITO.clone();
    System.out.println(Arrays.toString(lista));
    if(flug.equals("asc")) {
        boolean troca;
        for (int i = 0; i < lista.length; i++) {
            troca = false;
            for (int j = 0; j < lista.length - 1; j++) {
                if (lista[j] > lista[j + 1]) {
                    int temp = lista[j];
                    lista[j] = lista[j + 1];
                    lista[j + 1] = temp;
                    troca = true;
                    comparacoes++;
                }
            }
        }
    } else {
        int n = lista.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (lista[j] < lista[j + 1]) {
                    int temp = lista[j];
                    lista[j] = lista[j + 1];
                    lista[j + 1] = temp;
                    comparacoes++;
                }
            }
        }
    }
    System.out.println(Arrays.toString(lista));
    int[] troca_comparacoes = new int[2];
    troca_comparacoes[0] = comparacoes;
    troca_comparacoes[1] = comparacoes;
    return  troca_comparacoes;
}
public  static int[] insertionSort(int[] lista,String flug){
    System.out.println(Arrays.toString(lista));
    int key, j;
    int trocas=0, comparacoes=0;
    for (int i = 0; i < lista.length; i++) {
        key = lista[i];
        j = i - 1;

        if(flug=="asc"){
            while (j >= 0 && lista[j] > key){
```

```
                lista[j+1] = lista[j];
                j--;
                trocas++;
                comparacoes++;
            }
        }else {
            while (j >= 0 && lista[j] < key){
                lista[j+1] = lista[j];
                j--;
                trocas++;
                comparacoes++;
            }
        }

        lista[j+1] = key;
    }
    System.out.println(Arrays.toString(lista));
    int[] troca_comparacoes = new int[2];
    troca_comparacoes[0] = trocas;
    troca_comparacoes[1] = comparacoes;
    return  troca_comparacoes;
}
public  static int[] selectionSort(int[] lista, String flug){
    int min;
    int trocas = 0;
    int comparacoes =0;
    System.out.println(Arrays.toString(lista));

    for (int i = 0; i < lista.length; i++) {
        min = i;
        for (int j = i + 1 ; j < lista.length; j++) {
            comparacoes++;
            if(flug == "asc"){
                if(lista[min] > lista[j]){
                    min = j;
                }
            } else {
                if(lista[min] < lista[j]){
                    min = j;
                }
            }
        }
        if(i != min) {
            int temp = lista[i];
            lista[i] = lista[min];
            lista[min] = temp;
            trocas++;
        }
    }
    System.out.println(Arrays.toString(lista));
    int[] troca_comparacoes = new int[2];
    troca_comparacoes[0] = trocas;
    troca_comparacoes[1] = comparacoes;
    return  troca_comparacoes;
}
public static void quickSort(int[] arr, int low, int high) {
    System.out.println(Arrays.toString(arr));
    if (low < high) {
        int pivotIndex = partition(arr, low, high);
        quickSort(arr, low, pivotIndex - 1);
        quickSort(arr, pivotIndex + 1, high);
    }
    System.out.println(Arrays.toString(arr));
}

public static int partition(int[] arr, int low, int high) {
```

```
    public static int partition(int[] arr, int low, int high) {
        int pivot = arr[high];  // Escolha do pivô (último elemento)
        int i = low - 1;

        for (int j = low; j < high; j++) {
            comparisons++;
            if (arr[j] < pivot) {
                i++;
                trocas_quick++;
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }

        trocas_quick++;
        int temp = arr[i + 1];
        arr[i + 1] = arr[high];
        arr[high] = temp;

        return i + 1;
    }
    public static int[] generateRandomArray(int size, int min, int max) {
        int[] array = new int[size];
        Random random = new Random();

        for (int i = 0; i < size; i++) {
            array[i] = random.nextInt(max - min + 1) + min;
        }

        return array;
    }
    public static void WriteToFileExample(ArrayList<String> data) {
            try {
                FileWriter fileWriter = new FileWriter("out.txt");
                for (int i = 0; i < data.size(); i++) {
                    fileWriter.write(data.get(i)+"\n");
                }
                fileWriter.close();
                System.out.println("Dados Escritos no ficheiro out.txt, se continuar perdera os dados");
            } catch (IOException e) {
                System.out.println("An error occurred: " + e.getMessage());
            }
    }

}
```