23-09-2014

Centre for Information Technology

DTU Diplom
Lautrupvang 15, 2750 Ballerup

Sorting Algorithms

Data Structures and Algorithms

COMP3

Supervisor(s): Roger Munck-Fairwood

Student name	**Student number**
Aimo Suikkanen	s082577
Lukáš Janočko	s135272

1.Comments

We've created Java methods for Selection, Insertion and System(Quick) sorting.

System sorting was the fastest one as we expected.
Selection sorting was the slowest one as we expected.

In Selection and Insertion sort we expected following time results.
The ascending array should be sorted with the fastest time.
The descending array should be sorted with the slowest time.
The random array should be sorted with time which is between ascending and descending time.

We got the proper results in Insertion sorting.
On the other hand in Selection sorting we always got the Random sorting as the fastest.
We tried to edit the Java code and also tried to run it on different computer but it didn't help.

The time values in our table are all in seconds.

2.TABLES

Algorithm: Selection Sorting, **Input array**: (arrayAsc)

	N_1= 10000 $T(N_1)$	N_2= 30000 $T(N_2)$	N_3= 50000 $T(N_3)$	N_4= 100000 $T(N_4)$	Time-relation for N_1 & N_2	O-relation for N_1 & N_2	Time-relation for N_2 & N_3	O-relation for N_2 & N_3	Time-relation for N_3 & N_4	O-relation for N_3 & N_4
Run 1	0.11	0.921	2.578	10.077	8.373	9	2.799	2.77	3.909	4
Run 2	0.093	0.953	2.563	10.063	10.247		2.689		3.926	
Run 3	0.109	0.890	2.547	10.188	8.165		2.862		4.000	
Run 4	0.141	0.891	2.563	10.098	6.319		2.877		3.940	

Algorithm: Selection Sorting, **Input array**: (arrayDes)

	N_1= 10000 $T(N_1)$	N_2= 30000 $T(N_2)$	N_3= 50000 $T(N_3)$	N_4= 100000 $T(N_4)$	Time-relation for N_1 & N_2	O-relation for N_1 & N_2	Time-relation for N_2 & N_3	O-relation for N_2 & N_3	Time-relation for N_3 & N_4	O-relation for N_3 & N_4
Run 1	0.113	0.925	2.509	9.874	8.186	9	2.712	2.77	3.935	4
Run 2	0.107	1.089	2,477	9.888	10.178		2.275		3.992	
Run 3	0.109	0.917	2.529	9.919	8.413		2.758		3.922	
Run 4	0.514	0.897	2.583	9.881	5.825		2.880		3.825	

Algorithm: Selection Sorting, **Input array**: (arrayRan)

	N_1= 10000 $T(N_1)$	N_2= 30000 $T(N_2)$	N_3= 50000 $T(N_3)$	N_4= 10000 $T(N_4)$	Time-relation for N_1 & N_2	O-relation for N_1 & N_2	Time-relation for N_2 & N_3	O-relation for N_2 & N_3	Time-relation for N_3 & N_4	O-relation for N_3 & N_4
Run 1	0.122	0.897	2.535	9.889	7.352	9	2.826	2.77	3.900	4
Run 2	0.116	0.902	2.53	9.869	7.776		2.804		3.900	
Run 3	0.11	0.904	2.474	9.853	8.218		2.737		3.983	
Run 4	0.109	0.898	2.473	9.837	8.239		2.753		3.977	

Algorithm: Insertion Sort, **Input array**: (arrayAsc)

	N_1= 10000 $T(N_1)$	N_2= 30000 $T(N_2)$	N_3= 50000 $T(N_3)$	N_4= 100000 $T(N_4)$	Time-relation for N_1 & N_2	O-relation for N_1 & N_2	Time-relation for N_2 & N_3	O-relation for N_2 & N_3	Time-relation for N_3 & N_4	O-relation for N_3 & N_4
Run 1	0.001	0.002	0.003	0.005	2	9	1.5	2.77	1.667	4
Run 2	0.001	0.003	0.003	0.005	3		1		1.667	
Run 3	0.001	0.002	0.003	0.005	2		1.5		1.667	
Run 4	0.001	0.001	0.003	0.004	1		3		1.333	

Algorithm: Insertion Sort, **Input array**: (arrayDes)

	N_1= 10000	N_2= 30000	N_3= 50000	N_4= 100000	Time-relation for N_1 & N_2	O-relation for N_1 & N_2	Time-relation for N_2 & N_3	O-relation for N_2 & N_3	Time-relation for N_3 & N_4	O-relation for N_3 & N_4
	$T(N_1)$	$T(N_2)$	$T(N_3)$	$T(N_4)$						
Run 1	0.039	0.274	0.682	2.705	7.026	9	2.489	2.77	3.966	4
Run 2	0.038	0.254	0.679	2.693	6.684		2.673		3.966	
Run 3	0.039	0.253	0.672	2.680	6.487		2.656		3.988	
Run 4	0.039	0.256	0.676	2.686	6.564		2.641		3.973	

Algorithm: Insertion Sort, **Input array**: (arrayRan)

	N_1= 10000	N_2= 30000	N_3= 50000	N_4= 100000	Time-relation for N_1 & N_2	O-relation for N_1 & N_2	Time-relation for N_2 & N_3	O-relation for N_2 & N_3	Time-relation for N_3 & N_4	O-relation for N_3 & N_4
	$T(N_1)$	$T(N_2)$	$T(N_3)$	$T(N_4)$						
Run 1	0.026	0.134	0.351	1.365	5.154	9	2.619	2.77	3.889	4
Run 2	0.027	0.135	0.349	1.369	5.000		2.585		3.923	
Run 3	0.031	0.133	0.353	1.380	4.290		4.290		3.909	
Run 4	0.024	0.132	0.351	1.383	5.500		5.500		3.940	

Algorithm: System Sort, **Input array**: (arrayAsc)

	N_1= 10000	N_2= 30000	N_3= 50000	N_4= 100000	Time-relation for N_1 & N_2	O-relation for N_1 & N_2	Time-relation for N_2 & N_3	O-relation for N_2 & N_3	Time-relation for N_3 & N_4	O-relation for N_3 & N_4
	$T(N_1)$	$T(N_2)$	$T(N_3)$	$T(N_4)$						
Run 1	0.001	0.001	0.002	0.003	1	3.35	2	1.75	1.5	2.13
Run 2	0.001	0.002	0.002	0.004	2		1		2	
Run 3	0.001	0.001	0.002	0.004	1		2		2	
Run 4	0.001	0.001	0.002	0.004	1		2		2	

Algorithm: System Sort, **Input array**: (arrayDes)

	N_1= 10000	N_2= 30000	N_3= 50000	N_4= 100000	Time-relation for N_1 & N_2	O-relation for N_1 & N_2	Time-relation for N_2 & N_3	O-relation for N_2 & N_3	Time-relation for N_3 & N_4	O-relation for N_3 & N_4
	$T(N_1)$	$T(N_2)$	$T(N_3)$	$T(N_4)$						
Run 1	0.001	0.002	0.003	0.007	2	3.35	1.5	1.75	2.333	2.13
Run 2	0.001	0.003	0.003	0.006	3		1		2	
Run 3	0.001	0.002	0.003	0.006	2		1.5		2	
Run 4	0.001	0.002	0.003	0.006	2		1.5		2	

Algorithm: System Sort, **Input array**: (arrayRan)

	N_1= 10000	N_2= 30000	N_3= 50000	N_4= 100000	Time-relation for N_1 & N_2	O-relation for N_1 & N_2	Time-relation for N_2 & N_3	O-relation for N_2 & N_3	Time-relation for N_3 & N_4	O-relation for N_3 & N_4
	$T(N_1)$	$T(N_2)$	$T(N_3)$	$T(N_4)$						
Run 1	0.006	0.01	0.028	0.024	1.667	3.35	2.800	1.75	0.857	2.13
Run 2	0.007	0.01	0.016	0.024	1.429		1.600		1.500	
Run 3	0.006	0.011	0.028	0.022	1.833		2.545		0.786	
Run 4	0.006	0.011	0.019	0.024	1.833		0.173		1.263	

3.CODE

```
import java.util.Arrays;

/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author Aimo & Lukas
 */
public class Assignment1_SortingAlgorithms {

    public static void main(String[] args) {

        Assignment1_SortingAlgorithms ass1 = new
Assignment1_SortingAlgorithms();

        System.out.println("Selection sorting\n");
        ass1.createArray();

        System.out.println(Arrays.toString(ass1.arrayAsc));
        System.out.println(Arrays.toString(ass1.arrayDes));
        System.out.println(Arrays.toString(ass1.arrayRan));

        ass1.selectionSort(ass1.arrayAsc);
        ass1.selectionSort(ass1.arrayDes);
        ass1.selectionSort(ass1.arrayRan);

        System.out.println(Arrays.toString(ass1.arrayAsc));
        System.out.println(Arrays.toString(ass1.arrayDes));
        System.out.println(Arrays.toString(ass1.arrayRan));

        System.out.println("Insertion sorting\n");

        ass1.createArray();

        System.out.println(Arrays.toString(ass1.arrayAsc));
        System.out.println(Arrays.toString(ass1.arrayDes));
        System.out.println(Arrays.toString(ass1.arrayRan));

        ass1.insertionSort(ass1.arrayAsc);
        ass1.insertionSort(ass1.arrayDes);
        ass1.insertionSort(ass1.arrayRan);

        System.out.println(Arrays.toString(ass1.arrayAsc));
        System.out.println(Arrays.toString(ass1.arrayDes));
        System.out.println(Arrays.toString(ass1.arrayRan));

        System.out.println("System quick sorting\n");
        ass1.createArray();

        System.out.println(Arrays.toString(ass1.arrayAsc));
```

```
        System.out.println(Arrays.toString(ass1.arrayDes));
        System.out.println(Arrays.toString(ass1.arrayRan));

        ass1.systemSort(ass1.arrayAsc);
        ass1.systemSort(ass1.arrayDes);
        ass1.systemSort(ass1.arrayRan);

        System.out.println(Arrays.toString(ass1.arrayAsc));
        System.out.println(Arrays.toString(ass1.arrayDes));
        System.out.println(Arrays.toString(ass1.arrayRan));



    }

    public int arrayLength = 10000;
    long start;
    long stop;
    long waste;
    int[] arrayAsc = new int[arrayLength];
    int[] arrayDes = new int[arrayLength];
    int[] arrayRan = new int[arrayLength];

    public void createArray() {
        for (int i = 0; i < this.arrayLength; i++) {
            arrayRan[i] = (int) (Math.random() * this.arrayLength);
            arrayAsc[i] = i;
            arrayDes[i] = this.arrayLength - i - 1;
        }
    }

    public void selectionSort(int[] array) {
        start = System.currentTimeMillis();
        for (int i = 0; i < array.length - 1; i++) {
            int index = i;
            for (int j = index + 1; j < array.length; j++) {
                if (array[index] > array[j]) {
                    index = j;
                }
            }
            int lowestNumber = array[index];
            array[index] = array[i];
            array[i] = lowestNumber;

        }
        stop = System.currentTimeMillis();
        waste = stop - start;
        System.out.println("Time wasted: " + waste / 1000f);
    }

    public void insertionSort(int[] array) {
        start = System.currentTimeMillis();
        for (int i = 1; i < array.length; i++) {
            int sort = array[i];
            int j;
            for (j = i - 1; j >= 0 && sort < array[j]; j--) {
```

```
                array[j + 1] = array[j];
            }
            array[j + 1] = sort;

        }
        stop = System.currentTimeMillis();
        waste = stop - start;
        System.out.println("Time wasted: " + waste / 1000f);
    }

    public void systemSort(int[] array){
        start = System.currentTimeMillis();
        Arrays.sort(array);
        stop = System.currentTimeMillis();
        waste = stop - start;
        System.out.println("Time wasted: " + waste / 1000f);

    }

     }
```