Fuat Yiğit Koçyiğit – 16429085948

28.04.2021

TED ÜNİVERSİTESİ

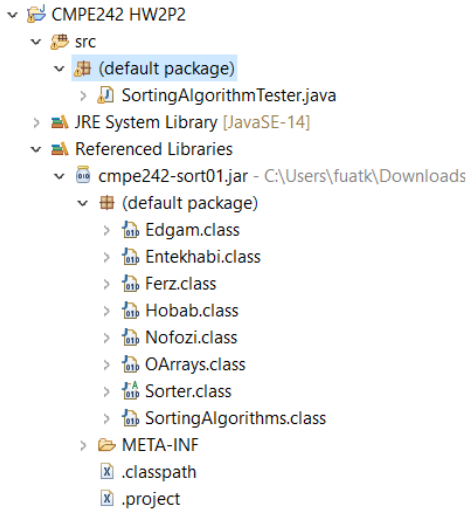
CMPE242 Homework 2 Part 2 Sort Algorithm Test REPORT

Experimental Setup

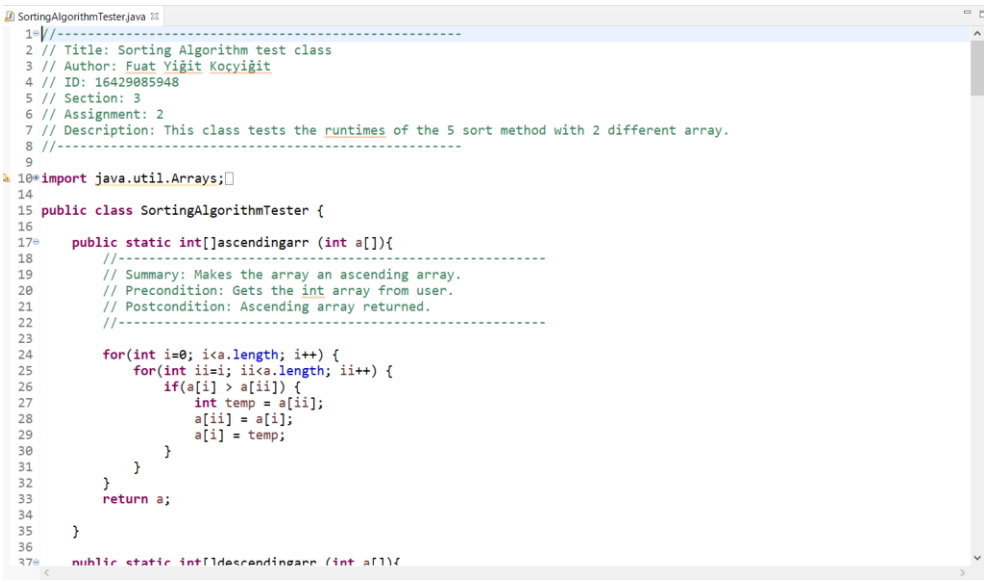
- To do the experiment, we used “Eclipse IDE for Java Developers” program.

Procedure

- Firstly we downloaded the jar file given by our instructors and configured the build path so we can use the referenced libraries.

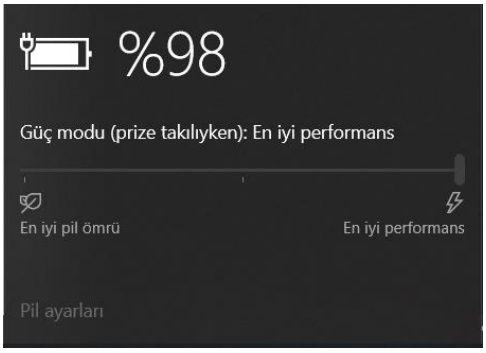


- After that we created a SortAlgorithmTester class and prepared the program that will give us the run times of 5 different sort algorithms to us.

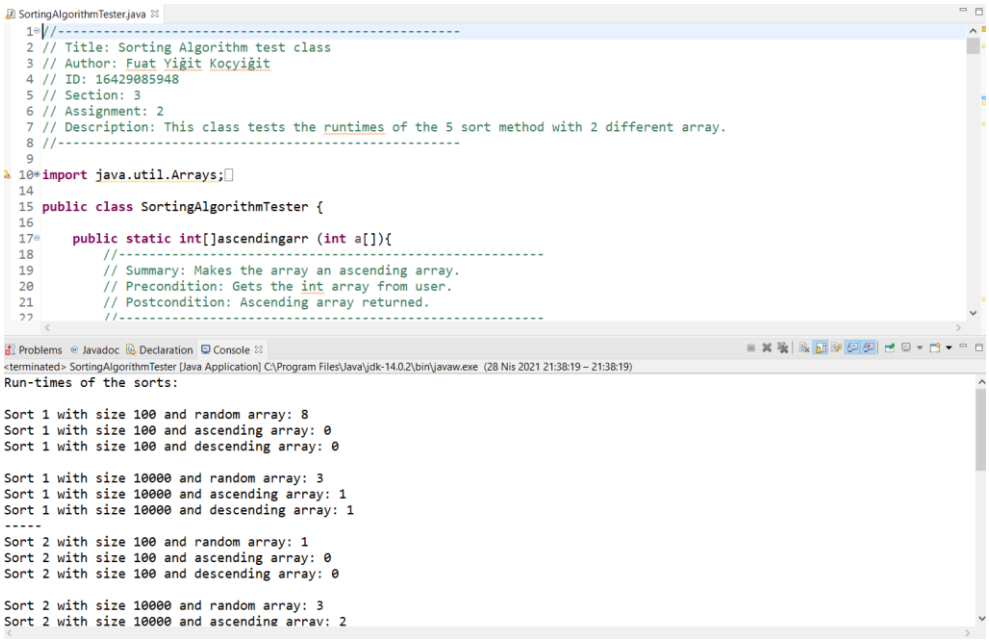


```
SortingAlgorithmTester.java
1  //-----------------------------------------------------
2  // Title: Sorting Algorithm test class
3  // Author: Fuat Yiğit Koçyiğit
4  // ID: 16429085948
5  // Section: 3
6  // Assignment: 2
7  // Description: This class tests the runtimes of the 5 sort method with 2 different array.
8  //-----------------------------------------------------
9
10 import java.util.Arrays;
14
15 public class SortingAlgorithmTester {
16
17     public static int[]ascendingarr (int a[]){
18         //-----------------------------------------------------
19         // Summary: Makes the array an ascending array.
20         // Precondition: Gets the int array from user.
21         // Postcondition: Ascending array returned.
22         //-----------------------------------------------------
23
24         for(int i=0; i<a.length; i++) {
25             for(int ii=i; ii<a.length; ii++) {
26                 if(a[i] > a[ii]) {
27                     int temp = a[ii];
28                     a[ii] = a[i];
29                     a[i] = temp;
30                 }
31             }
32         }
33         return a;
34
35     }
36
37     public static int[]descendingarr (int a[]){
```

- After our project is ready, we closed all the background apps and only left the Eclipse IDE and set computer to the max performance mode so we get the highest performance. The reason we did is that we want to get the most accurate results.



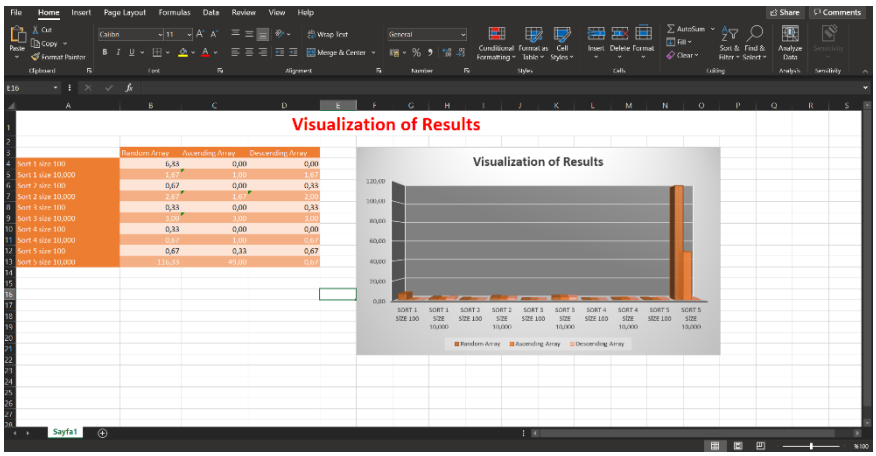
- Then we run the program 3 times and noted the results. (My run results are on the page 5, 6 and 7)



```
SortingAlgorithmTester.java
1  //----------------------------------------------------
2  // Title: Sorting Algorithm test class
3  // Author: Fuat Yiğit Koçyiğit
4  // ID: 16429085948
5  // Section: 3
6  // Assignment: 2
7  // Description: This class tests the runtimes of the 5 sort method with 2 different array.
8  //----------------------------------------------------
9
10 import java.util.Arrays;
14
15 public class SortingAlgorithmTester {
16
17     public static int[]ascendingarr (int a[]){
18         //-------------------------------------------------------
19         // Summary: Makes the array an ascending array.
20         // Precondition: Gets the int array from user.
21         // Postcondition: Ascending array returned.
22         //-------------------------------------------------------
```

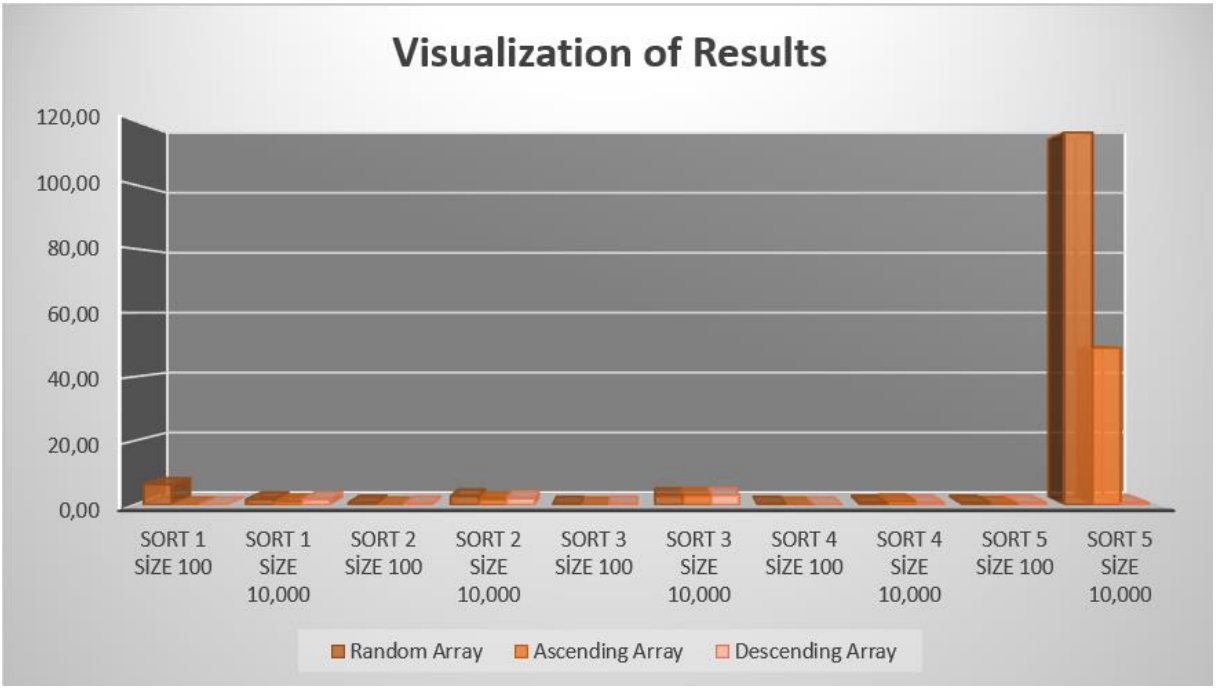
```
Problems  Javadoc  Declaration  Console
<terminated> SortingAlgorithmTester [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe  (28 Nis 2021 21:38:19 – 21:38:19)
Run-times of the sorts:

Sort 1 with size 100 and random array: 8
Sort 1 with size 100 and ascending array: 0
Sort 1 with size 100 and descending array: 0

Sort 1 with size 10000 and random array: 3
Sort 1 with size 10000 and ascending array: 1
Sort 1 with size 10000 and descending array: 1
-----
Sort 2 with size 100 and random array: 1
Sort 2 with size 100 and ascending array: 0
Sort 2 with size 100 and descending array: 0

Sort 2 with size 10000 and random array: 3
Sort 2 with size 10000 and ascending array: 2
```

- Finally, we made the visualization of our results on Microsoft Excel.



Experimental Results

	Random Array	Ascending Array	Descending Array
Sort 1 size 100	6,33	0,00	0,00
Sort 1 size 10,000	1,67	1,00	1,67
Sort 2 size 100	0,67	0,00	0,33
Sort 2 size 10,000	2,67	1,67	2,00
Sort 3 size 100	0,33	0,00	0,33
Sort 3 size 10,000	3,00	3,00	3,00
Sort 4 size 100	0,33	0,00	0,00
Sort 4 size 10,000	0,67	1,00	0,67
Sort 5 size 100	0,67	0,33	0,67
Sort 5 size 10,000	116,33	49,00	0,67



From our 3 java program run's average run-times, we observed that sort1 algorithm is not good with small sized random arrays it is good at big sized arrays. Sort2 algorithm is very efficient at small sizes in every type of array but average efficient at the big sizes. Sort3 algorithm is also very efficient at small sizes in every type but not that efficient at big sized arrays. Sort4 algorithm is very efficient in both small and big sized arrays and also random, ascending and descending arrays. Sort5 is so efficient at small size but very inefficient at the big sizes as we can also see from the run-time graph.

I think sort2 and sort4 may be "bubble sort" or "insertion sort" since they have the most efficient results and did not affected from the size. Sort5 may be "selection sort" because it is the worst efficient results in that 5 algorithm like $O(n^2)$. Sort3 may be "quick sort" because it is efficient at small size but has a bad efficiency at big sizes.

Program Run Results

First run:

```
Run-times of the sorts:

Sort 1 with size 1000 and random array: 6
Sort 1 with size 1000 and ascending array: 0
Sort 1 with size 1000 and descending array: 0

Sort 1 with size 100000 and random array: 2
Sort 1 with size 100000 and ascending array: 1
Sort 1 with size 100000 and descending array: 2
-----
Sort 2 with size 1000 and random array: 0
Sort 2 with size 1000 and ascending array: 0
Sort 2 with size 1000 and random array: 1

Sort 2 with size 100000 and random array: 3
Sort 2 with size 100000 and ascending array: 1
Sort 2 with size 100000 and descending array: 2
-----
Sort 3 with size 1000 and random array: 0
Sort 3 with size 1000 and ascending array: 0
Sort 3 with size 1000 and descending array: 1

Sort 3 with size 100000 and random array: 3
Sort 3 with size 100000 and ascending array: 3
Sort 3 with size 100000 and descending array: 3
-----
Sort 4 with size 1000 and random array: 0
Sort 4 with size 1000 and ascending array: 0
Sort 4 with size 1000 and random array: 0

Sort 4 with size 100000 and random array: 1
Sort 4 with size 100000 and ascending array: 1
Sort 4 with size 100000 and descending array: 0
-----
Sort 5 with size 1000 and random array: 1
Sort 5 with size 1000 and ascending array: 0
Sort 5 with size 1000 and random array: 0

Sort 5 with size 10000 and random array: 117
Sort 5 with size 100000 and ascending array: 47
Sort 5 with size 100000 and descending array: 1
-----
```

Second run:

```
Run-times of the sorts:

Sort 1 with size 1000 and random array: 6
Sort 1 with size 1000 and ascending array: 0
Sort 1 with size 1000 and descending array: 0

Sort 1 with size 100000 and random array: 1
Sort 1 with size 100000 and ascending array: 1
Sort 1 with size 100000 and descending array: 1
-----
Sort 2 with size 1000 and random array: 1
Sort 2 with size 1000 and ascending array: 0
Sort 2 with size 1000 and random array: 0

Sort 2 with size 100000 and random array: 2
Sort 2 with size 100000 and ascending array: 2
Sort 2 with size 100000 and descending array: 2
-----
Sort 3 with size 1000 and random array: 0
Sort 3 with size 1000 and ascending array: 0
Sort 3 with size 1000 and descending array: 0

Sort 3 with size 100000 and random array: 2
Sort 3 with size 100000 and ascending array: 3
Sort 3 with size 100000 and descending array: 3
-----
Sort 4 with size 1000 and random array: 1
Sort 4 with size 1000 and ascending array: 0
Sort 4 with size 1000 and random array: 0

Sort 4 with size 100000 and random array: 0
Sort 4 with size 100000 and ascending array: 1
Sort 4 with size 100000 and descending array: 1
-----
Sort 5 with size 1000 and random array: 0
Sort 5 with size 1000 and ascending array: 1
Sort 5 with size 1000 and random array: 0

Sort 5 with size 10000 and random array: 112
Sort 5 with size 100000 and ascending array: 51
Sort 5 with size 100000 and descending array: 0
-----
```

Third run:

```
Run-times of the sorts:

Sort 1 with size 1000 and random array: 7
Sort 1 with size 1000 and ascending array: 0
Sort 1 with size 1000 and descending array: 0

Sort 1 with size 100000 and random array: 2
Sort 1 with size 100000 and ascending array: 1
Sort 1 with size 100000 and descending array: 2
-----
Sort 2 with size 1000 and random array: 1
Sort 2 with size 1000 and ascending array: 0
Sort 2 with size 1000 and random array: 0

Sort 2 with size 100000 and random array: 3
Sort 2 with size 100000 and ascending array: 2
Sort 2 with size 100000 and descending array: 2
-----
Sort 3 with size 1000 and random array: 1
Sort 3 with size 1000 and ascending array: 0
Sort 3 with size 1000 and descending array: 0

Sort 3 with size 100000 and random array: 4
Sort 3 with size 100000 and ascending array: 3
Sort 3 with size 100000 and descending array: 3
-----
Sort 4 with size 1000 and random array: 0
Sort 4 with size 1000 and ascending array: 0
Sort 4 with size 1000 and random array: 0

Sort 4 with size 100000 and random array: 1
Sort 4 with size 100000 and ascending array: 1
Sort 4 with size 100000 and descending array: 1
-----
Sort 5 with size 1000 and random array: 1
Sort 5 with size 1000 and ascending array: 0
Sort 5 with size 1000 and random array: 0

Sort 5 with size 10000 and random array: 120
Sort 5 with size 100000 and ascending array: 49
Sort 5 with size 100000 and descending array: 1
-----
```