UNIVERSITY OF WISCONSIN PARKSIDE

Department of Computer Science

CSCI 340/CIS 540 Data Structures/Algorithm Design

Homework #2: 50 points

Outcomes:

- Write programs that use Recursive algorithms.
- Use Divide-and-Conquer algorithm design paradigm.
- Use Sorting algorithms to solve computational problems.
- Generate random numbers of different distributions for experimental analysis of algorithms.

Scoring:

- If you do not submit a zip file containing your source code, your score will be zero.
- If you submit the source code that does not compile, your score will be zero.
- If you submit source code without the correct class name (`FindMedian`) or your program partially works, you will receive at most half credit for this assignment.
- Deductions will be made for not meeting the usual requirements:
    - The source code is not formatted according to the usual style guidelines, including commenting on each method to explain its purpose.

Create an IntelliJ Java Project called Program2. Write the comment lines to include the author's name(s) and complete the project following the instructions specified in Part 1 and Part 2.

Part 1 (30 points)

Create a Java class named `FindMedian` with a method

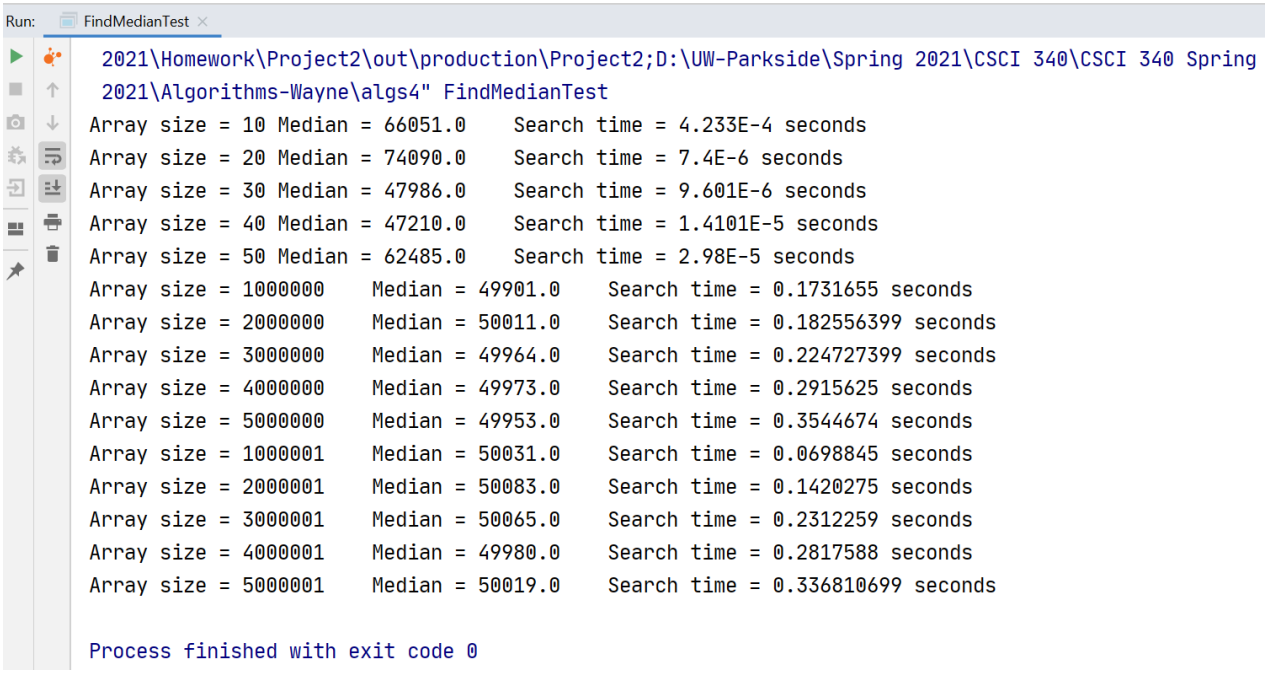
```
public static double findMedian (int [] array)
```

This method is to find the *median* of the array of integers. The median is defined as the middle element if the array is sorted. For example, the median of an array containing 7, 1 and 3 is 3. If there is an even number of elements in the array, the median is defined as the average of the middle two elements. For example, the median of an array containing 7, 1, 3 and 4 is 3.5.

A test harness is available from the projects page on the Canvas course web site. Your instructor will use this test harness to test your programs, but change the random number seed. The array data is generated by using the uniform distribution from the `StdRandom` class.

One easy approach to this problem is to sort the array and then calculate and return the median, but there are faster methods available. For example, consider using a modified ***Quicksort*** that only sorts the partitions needed to find the median. You can customize the Quicksort algorithm from algs4.jar file.

Your instructor wrote a naïve `findMedian` that uses `Arrays.sort`. Here are her results:



Using `Arrays.sort` you can find the median of 5,000,001 integers is 0.3368 second. On average, you should be able to do significantly better than this.

Only those students who can beat the times given above will receive full credit for this assignment.

Don't forget to use good programming techniques. A significant portion of your grade is still based on proper indentation, adequate comments, etc.

Part 2 (20 points)

(i) Estimate the amount of time that it would take in your computer to run the **ThreeSumFast.java** available in the algs4.jar file that we used in the class. Start at n equal to 1000, and print n, and the number of seconds. Now, use your program to find the execution times for 2000, 4000, 8000, and 16000 numbers using the following procedure:

a. Collect data and plot the data on a log-log graph, and a trend line. Find the equation for this trend line. (Excel can help with this.) Then use this to derive a power law equation from this.

b. Using the doubling technique estimate the exponent in a power law equation. Then, use the techniques described in class to estimate the constant.

c. Now, use your two estimates to predict how long it would take to solve the problem for an array of one million numbers.

Grading Rubric:

Requirement	Full credit	Partial credit
Implement the `FindMedian` Class using the random number data (30 points)	Your program implements the `FindMedian` Class as described.	Your program implements the class, but with some errors.
A document (pdf) that presents time complexity analysis, graphs, validation, and prediction (20 points)	Your document presents the details as described.	Your document is submitted, but with some errors.
Format code (follow Style Guidelines) and output as specified (Additional deduction)	Your output is formatted as specified, including proper spacing, spelling, and so on.	You did not follow some or all of the requirements for output.

What to Submit:

Electronically submit a zip file containing the following files.

- `FindMedian.java` file.
- a document (pdf) that presents:
  - graphs of your data (x-y plot)
  - your analyses of this data to show your derivations of the power law equations for both methods
  - validation of your equations by checking the prediction with actual results for lager array sizes.
  - your prediction of how long it would take to solve the problem for an array of ten million numbers.