ÇANKAYA UNIVERSITY

SENG 201 Data and Game Structures

Programming Assignment 2

This assignment is designed to help your understanding of the concepts of sorting algorithms.

PART I

In this part of the assignment, you are asked to make some modifications on merge sort algorithms that we have discussed in class.

IMPORTANT: The homework consists of steps and you should follow them sequentially, i.e. do not go to the next step before finishing the current step.

Step 1: create a **Person** class. Each person should have three fields: a String attribute, called name, a second String attribute called surname, a third attribute of type long, called id. The Person class should implement the "Comparable" interface to be able to compare it with respect to their "ids".

Step 2: create a Person array having 10 Person objects, each with a different name, surname, id (you can randomly set different name, surname and ids)

Step 3: modify the Merge Sort algorithm using the implementation from the lecture slides to sort a Comparable array in descending order. Sort the Person array in descending order to check.

Step 4: overload the Merge Sort method which will take a second parameter with the type "Comparator". This time the sorting should be done using this Comparator object.

Step 5: create two Comparators for the Person object. First comparator should compare Person objects according to the "name" field. The second comparator should compare Person objects according to the "surname" field. Now sort the Person array in descending order with respect to "name" and "surname" using comparators.

Step 6: now repeat the steps 3, 4, 5 for Quick Sort algorithm using the codes provided on lecture slides (modify for descending order, overload with Comparator, you can use the same comparators you created to sort the Person array in descending order using Quick sort with respect to name and surname fields)

ÇANKAYA UNIVERSITY

PART II

In this part of the assignment, you are given a package named **`sorter.jar`** that is available at the course Webonline site. This jar file contains a class that has implemented 5 sorting algorithms: QuickSort, MergeSort, Insertion Sort, Bubble Sort, and Selection Sort. These sorting algorithms have been implemented using almost similar source code as in the course slides (and lab for bubble sort). Note that the QuickSort algorithm does not use shuffle in this implementation.

`sorter.jar` package includes the following class.

Class Name: `sorter`

Methods:

- `public static void sort1(int [] arr)`
- `public static void sort2(int [] arr)`
- `public static void sort3(int [] arr)`
- `public static void sort4(int [] arr)`
- `public static void sort5(int [] arr)`

In this part, you are expected to guess the sorting algorithm that the class uses for each sort method, by looking and comparing execution time of the program to sort different sizes of ascending/descending and random ordered integer arrays.

1. To use the jar package, which is including the five sorting algorithms, you should follow the following steps in Eclipse:
   - right-click on the Project → Build Path → Configure Build Path
   - under “Libraries” tab select “Classpath”
   - click “Add External JARs” and give the Jar
   - click “Apply and Close”
   - now in your main method you can call the sorting method as following:

```
sorter.sort1( array );
sorter.sort2( array );
…
```

2. As you test the algorithms, collect time measures to make a guess about which sorting algorithm was running.

3. For this part, you should submit your Tester.java where you implemented the tests and your at most 3-page report in PDF format. The report should give a detailed explanation of your experimental setup, procedure, and experimental results justifying your answer. You should also write the individual steps that you took to complete the homework.

   You are expected to add visualization (plot, graph etc.) showing the time complexity of the sorting methods with different type of input (e.g. ascending, descending, random input). If you couldn't differentiate the corresponding sorting algorithms by examining the experimental results, you should list all sorting algorithms that may correspond to the given case, together with your explanations why you came up with this conclusion.

4. Hint: for especially large arrays, you will likely get a stack overflow error and your program will not run. This is due to the limited area that JVM reserves for its own call stack. The call stack is used for storing local variables of the methods, method call arguments, etc.

   To increase the size of the call stack size, try to use "-Xss" command line argument when you run your program (e.g. use "-Xss8m" to increase the stack size to 8 Mbytes, of course you can increase it more). Do research on how to add this option in your Java development environment (e.g. For Eclipse, follow the tutorial in the following link: https://www.youtube.com/watch?v=OU0H3d1rhfw).

IMPORTANT

IMPORTANT NOTES: Do not start your homework before reading these notes!!!

1. This assignment is due by 23:59 on Saturday, December 2nd.

2. You should submit your homework as **a single zip file** including your java files and pdf report. You should submit to the course webonline page before the deadline.

3. The rules about late homework submissions apply. Please see the course syllabus for further discussion of the late homework policy as well as academic integrity.

4. The submissions that do not obey these rules will not be graded.

5. To increase the efficiency of the grading process as well as the readability of your code, you have to follow the following instructions about the format and general layout of your program.

   - Indentation, indentation, indentation... The format of your code makes it readable!

   - This homework will be graded by your TA, Naz Dündar. Thus, you may ask them your homework related questions. You are also welcome to ask your course instructor Bora Çelikkale for help.