UNIVERSITY OF VIRGINIA, DEPARTMENT OF COMPUTER SCIENCE

Advanced Sorts - Implementation

Nada Basit and Mark Floryan

March 7, 2022

1 SUMMARY

For this homework, you will be implementing two more sorting algorithms: *Merge Sort* and *Quick Sort*

1. Download the provided starter code.
2. Implement two sorting algorithms.
3. Use the provided tester to check if each methods are working correctly.
4. **FILES TO DOWNLOAD:** AdvancedSorts.zip
5. **FILES TO SUBMIT:** AdvancedSorts.java

1.1 ADVANCEDSORTS.JAVA

You will be implementing two sorting algorithms for this homework. They are listed below:

```
1 /**
    * Recursive merge sort and associated private helper method
3   * the second method below provides the portion of the array
```

```
   * (i.e., index i to j inclusive) that we want to sort.
   */
public static<T extends Comparable<T>> void mergeSort(T[] list);
private static<T extends Comparable<T>> void
                                    mergeSort(T[] list, int i, int j);

/**
   * Recursive quick sort and associated private helper method
   * the second method below provides the portion of the array
   * (i.e., index i to j inclusive) that we want to sort.
   */
public static<T extends Comparable<T>> void quickSort(T[] list);
private static<T extends Comparable<T>> void
                                    quickSort(T[] list, int i, int j);
```

You may add additional helper methods that might be useful in implementing these sorting algorithms. Some methods that may be useful, but are optional, include swapping two elements given indices *i* and *j*, merging two sorted lists into a single sorted list, partitioning a list around a pivot, etc. You'll notice the starter code provides method headers for two important helper methods: *partition()* and *merge()*

The provided tester (main method inside of *Tester.java*) can be used to test your code. The method takes in two values as input: the size of the array to sort and which method to use (1 = merge sort and 2 = quicksort). The tester will call the appropriate sorting method on a list of the given size. It will take the result and check if 1) the sizes of the unsorted and sorted lists are still the same, 2) the elements in the sorted list are indeed in sorted order, and 3) the sorted list contains the same elements in total that the original list does. If any of these checks fail, you should get a notification to that effect.

1.2 GRADESCOPE

You should submit your code to *Gradescope*. If you are having trouble with your submission, you should double check the following common problems:

1. Make sure you are only submitting one file, and it is called *AdvancedSorts.java* exactly.
2. Make sure you keep the package statement at the top of the file (package sorting;).
3. Your code should **NOT contain any additional import statements**
4. Make sure you **DO NOT** submit a *main method* in your file.

5. Make sure your file is **NOT printing anything to the console**. Your methods should simply return the correct results.