CMPU102 Computer Science II
Lab 9

The SortingDemo project contains implementations of the following sorting algorithms: bubbleSort, selectionSort, insertionSort, mergeSort and quickSort. In this laboratory exercise, you will modify the implementations of these algorithms to generate data about their performance. You will use Libre Office Calc to graph the performance data. Finally, you will use the performance graphs to compare these algorithms to each other.

1. Modify the sorting methods to count the numbers of comparisons they use to sort an array. For this purpose, you should define a method: 
   ```
   public <T extends Comparable<T>> int compare(T foo, T bar)
   ```
   that increments a comparisonCount variable and then calls compareTo to compare foo and bar. Then modify each of the sorting methods to call this compare method rather than directly calling the compareTo method.

2. Modify the public void run() method to execute a selected sorting method on a series of arrays of sizes: 1*M, 2*M, ..., K*M where K and M are the constants: NUM_SIZES and START_SIZE. For each array size, the run method should test the sorting method on N different arrays of that size, where N is the constant ARRAYS_PER_SIZE. Use the (already defined) method Integer[] randomIntegerArray(int size, int range) to generate an array of length size with random integers 0 ... range-1. The run method should store the sizes of arrays in an Integer[] sizeTable variable and the average number of comparison used to sort arrays of each size in an Integer[] comparisonTable variable. After generating and recording all the test data, the run method should print sizeTable and comparisonTable to a console window.

3. Exercise your program to generate test data for each of the sorting methods NUM_SIZES = 15 and START_SIZE = 10000. Open the SortingData.xlsx in Office Libre Calc by double clicking on it. Cut and past the sorting data from NetBeans to Office Libre Calc, entering each type of data in the appropriate column. Use the Hide and UnHide tools to selectively show or hide columns/algorithms.
   a. Generate one graph showing all three naïve sorting algorithms. Use this graph to decide which naïve sorting algorithm has the best performance. Note the relationship between the bubbleSort data and the selectionSort data.
   b. Generate one graph comparing mergeSort to quickSort. Use this graph to decide which of these two algorithms has better performance.
   c. Finally generate a graph comparing the best naïve algorithm to the best of mergeSort and quickSort.

4. Modify the SortingExperiments program to generate data on the CPU time needed to sort arrays using each of the sorting methods. For this purpose, you should call System.currentTimeMillis() before and after each call to a sorting method to determine the elapsed CPU time. Generate an Excel plot comparing the CPU time used by mergeSort and quickSort in solving the same sets of problems as you used when comparing comparisons. Try to determine which of mergeSort and quickSort is faster in terms of this technique for measuring CPU time.