Textbook Chapter 4 – Sorting Algorithms – Merge and Quick Sort

Chapter 4 provides an introduction, analysis, and pseudo code for various sorting algorithms. These include Insertion Sort, Bubble Sort, Shell Sort, Radix Sort, Heap Sort, Merge Sort, and Quick Sort. Comparisons and criteria of applicability are discussed. Being one of the fundamental computer applications, it is important to understand each of the techniques.

# Assignment

**Merge Sort**

Implement the *Merge Sort* algorithm based on the pseudo code given on page 107/109 of the textbook, sorting the values from lowest to highest. Instrument the code with one comparison for each time through the while loop within MergeLists.

Run the code in a Monte Carlo fashion on list sizes of , recording the comparison counts for each value of N. The initial data are the values [0..N-1] uniformly, randomly distributed throughout the array. Plot the comparison counts using Excel or some other graphing tool.

Run the same simulation as described above but for lists that are in pre-sorted in ascending (smallest to largest) order. Plot the results.

Run the same simulation as described above but for lists that are in pre-sorted in descending (largest to smallest) order. Plot the results.

**Quick Sort**

Implement the *Quick Sort* algorithm based on the pseudo code given on page 115-116 of the textbook, sorting the values from lowest to highest. Instrument the code with one comparison for each time through the for loop within PivotList.

Run the code in a Monte Carlo fashion on list sizes of , recording the comparison counts for each value of N. The initial data are the values [0..N-1] uniformly, randomly distributed throughout the array. Plot the comparison counts using Excel or some other graphing tool.

Run the same simulation as described above but for lists that are in pre-sorted in ascending (smallest to largest) order. Plot the results.

Run the same simulation as described above but for lists that are in pre-sorted in descending (largest to smallest) order. Plot the results.

The above steps are to be performed for three times for these PivotList configurations:

* The pivot point is the first location of the list
* The pivot point is the last location of the list
* The pivot point is a random location of the list (changes each time PivotList is called)

# Notes

* Your counter variable needs to be of type **long** (do not use **int**).
* To randomize the placement of the data in the array, assign initial values of [0..N-1] then shuffle them using the *Fisher-Yates* algorithm [[Wikipedia](https://en.wikipedia.org/wiki/Fisher%E2%80%93Yates_shuffle)] which you will need to implement.
* Before running simulations, verify that your code works by running it on this array of values

{6, 2, 4, 7, 1, 3, 8, 5}

# Deliverables

* Source code
* A reflection document including
  + Plot size vs. count for the three merge sort specifications
  + Plot size vs. count for the nine quick sort specifications
  + Essay describing successes and difficulties

Reflection document must be a PDF file. Do not submit documents of type MSWord, Pages, OpenOffice, etc.