**Time analysis for sorting algorithms**

The time analysis for four sorting algorithms was performed. The algorithms were: Quicksort, Randomized Quicksort, Mergesort, and Insertion sort.

The Big-Ohh notation for these sorting algorithms is as follows:

Quicksort – O(n log(n) )

rQuicksort – O(n log(n) )

Mergesort – O(n log(n) )

Insertion – O( n^2 )

**Analysis of the real data**

Tests were conducted to document the real run times of each of the four sorting algorithms. All the tests were made by sorting an array of integers. I used the Random class that is included in standard Java libraries to generate random integers from 0 – 99. “The Random java class uses a 48-bit seed, which is modified using a linear congruential formula… The algorithms implemented by class Random use a protected utility method that on each invocation can supply up to 32 pseudo randomly generated bits.” (Oracle Documentation)

In general, the tests followed their respective Big-Ohh representations. Using the graphs to create a visual model of the time results for each algorithm, the two quick sorts performed very well. The insertion sort also performed as expected increasing the time significantly as size of N increased.

The very strange outlier my results showed was the speed at which the merge sort algorithm performed.