## Bubble Sort

Algorithm used was the one demonstrated in class.

It is difficult to compare this algorithm to the others, as it is many times slower.

Due to it’s much slower speed, I processed the number of elements that would take about the same time as the other sorting algorithms tested.

From the results shown in the chart above, I would label the complexity of this algorithm as O(n^2). Each element added seems to cause the number of operations to increase exponentially.

## Heap Sort

The heap sort algorithm was copied from <https://www.geeksforgeeks.org/python-program-for-heap-sort/>.

Here the complexity best matches O(nlogn). Compared to Merge and Quick Sort, Heap sort seems to increase use more operations to process data as elements increase.

## Merge Sort

This merge sort algorithm was found at <https://www.geeksforgeeks.org/merge-sort/>. It was submitted by Mayank Khanna.

The complexity of this sorting algorithm is best described as O(2^n). There is a linear relationship between the elements processed and the operations required to sort.

## Quick Sort

This Quick Sort algorithm was found at https://www.geeksforgeeks.org/python-program-for-quicksort/

The complexity of this sorting algorithm is best described as O(2^n). There is a linear relationship between the elements processed and the operations required to sort.

## A Comparison of Sorts

The comparison above illustrates the performance of the best 3 algorithms tested. Bubble sort was slower by too many factors to effectively be included in this graph.