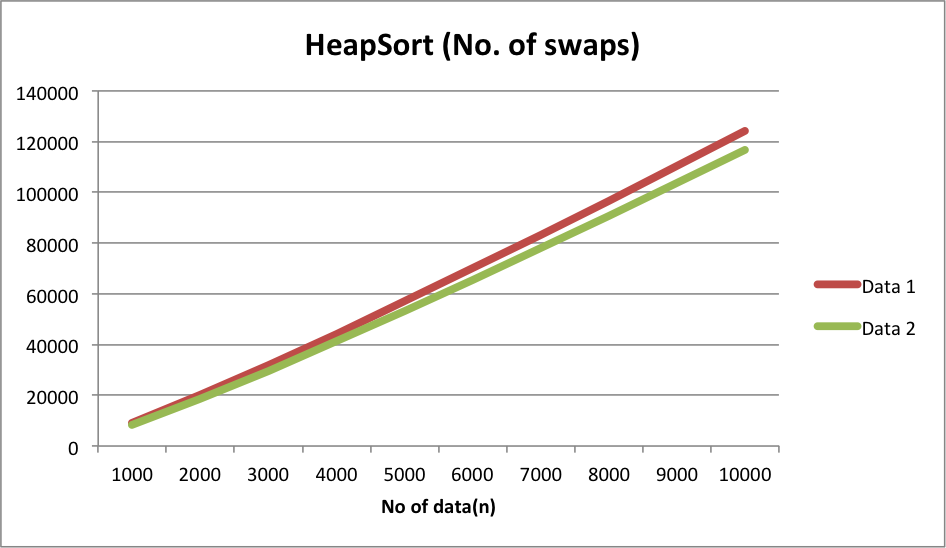
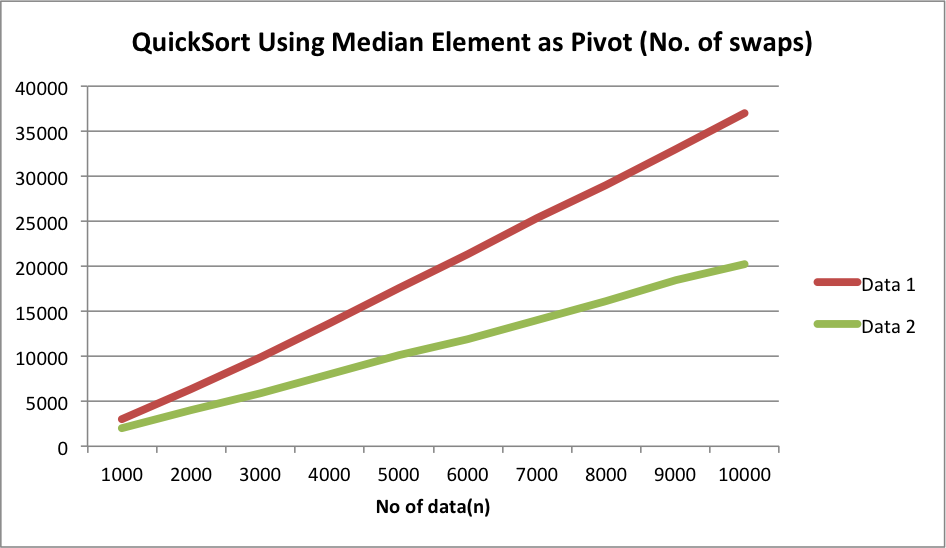
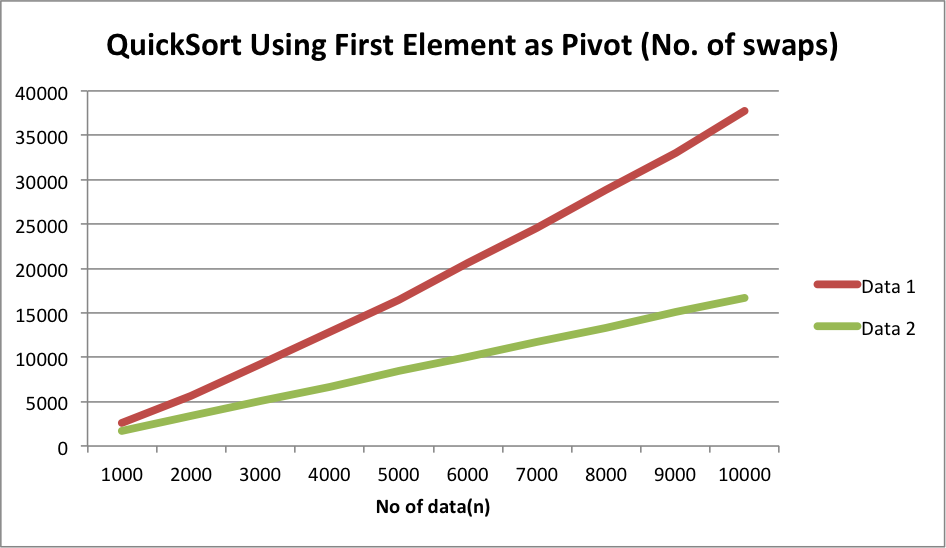
**Name: Garcia Anthony John Abril**

**Student ID: 4321819**

**CSCI203 Assignment 3**



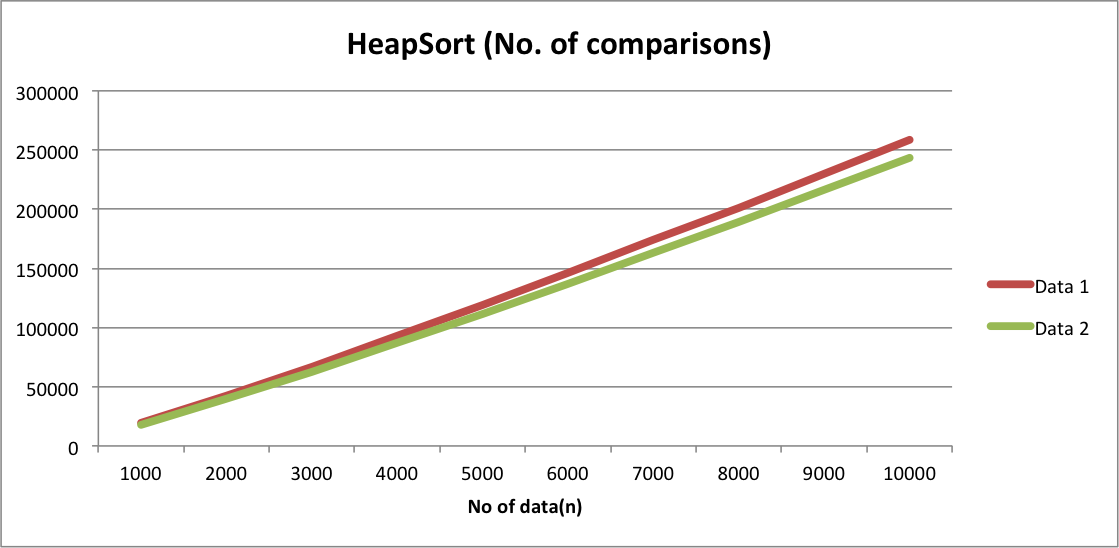
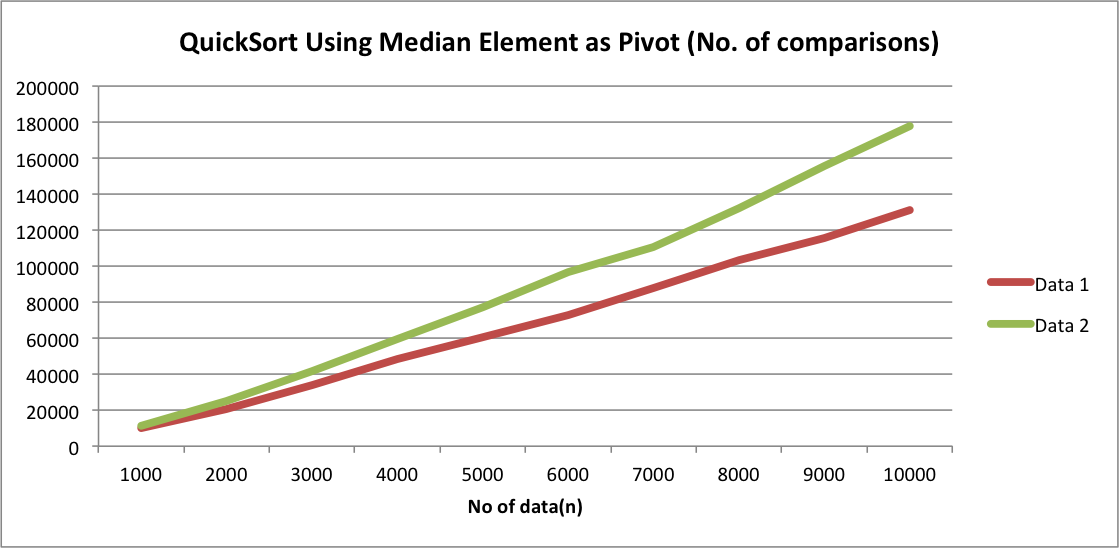
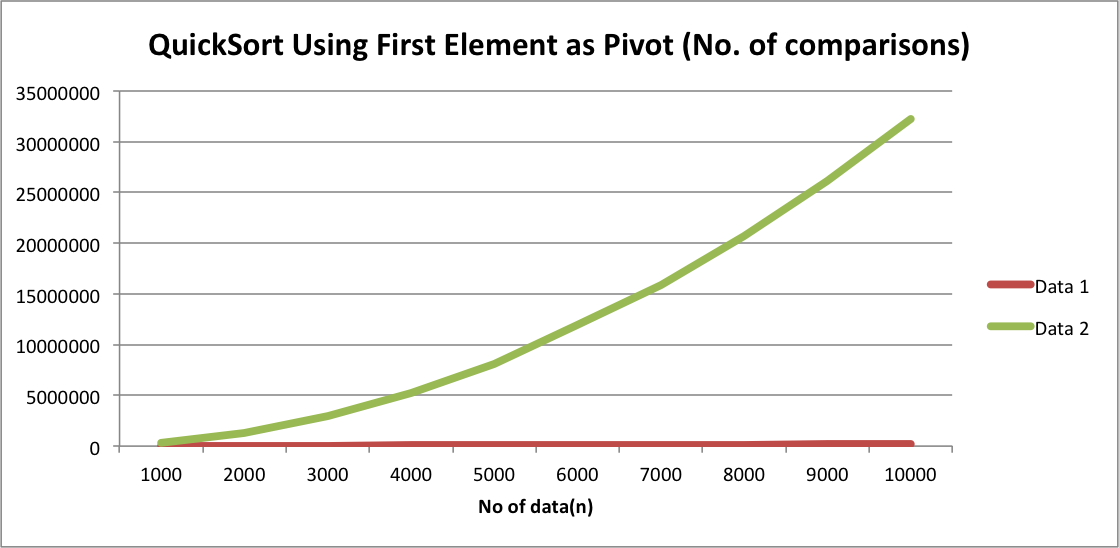
In the first set of graphs, we compare the number of swaps for 3 pairs of data. We can see that for all 3 graphs, Data 1, which numbers are arranged in randomized order, has more swaps that Data 2, which numbers are arranged in reverse order.

Comparing the 3 sorts, it can be seen that for data 1, heapsort has more swaps compared to the other two sorts, which are relatively close.

Again comparing the 3 sorts, it can be seen that for data 2, heapsort has more swaps compared to the other two sorts, where quicksort using median element has slightly less number of swaps.

Another observation is that heapsort has relatively close number of swap regardless of the order of the data.

It can also be seen that all graphs for number of swaps are linear.



This second set of graphs, we compare the number of comparisons for 3 pairs of data.

We can see that data 2, which is the data, arranged in worse case, has more comparisons for both quicksorts. This is not the case for heapsort.

We can also see that in the quicksort that uses the first element as a pivot, the number of comparisons is almost 35 million and is heading towards O(n2).

Both quicksorts has non-linear number of comparisons compared to heapsort. This is probably due to the recursive nature quicksort and iterative (for-loop) nature of heapsort.

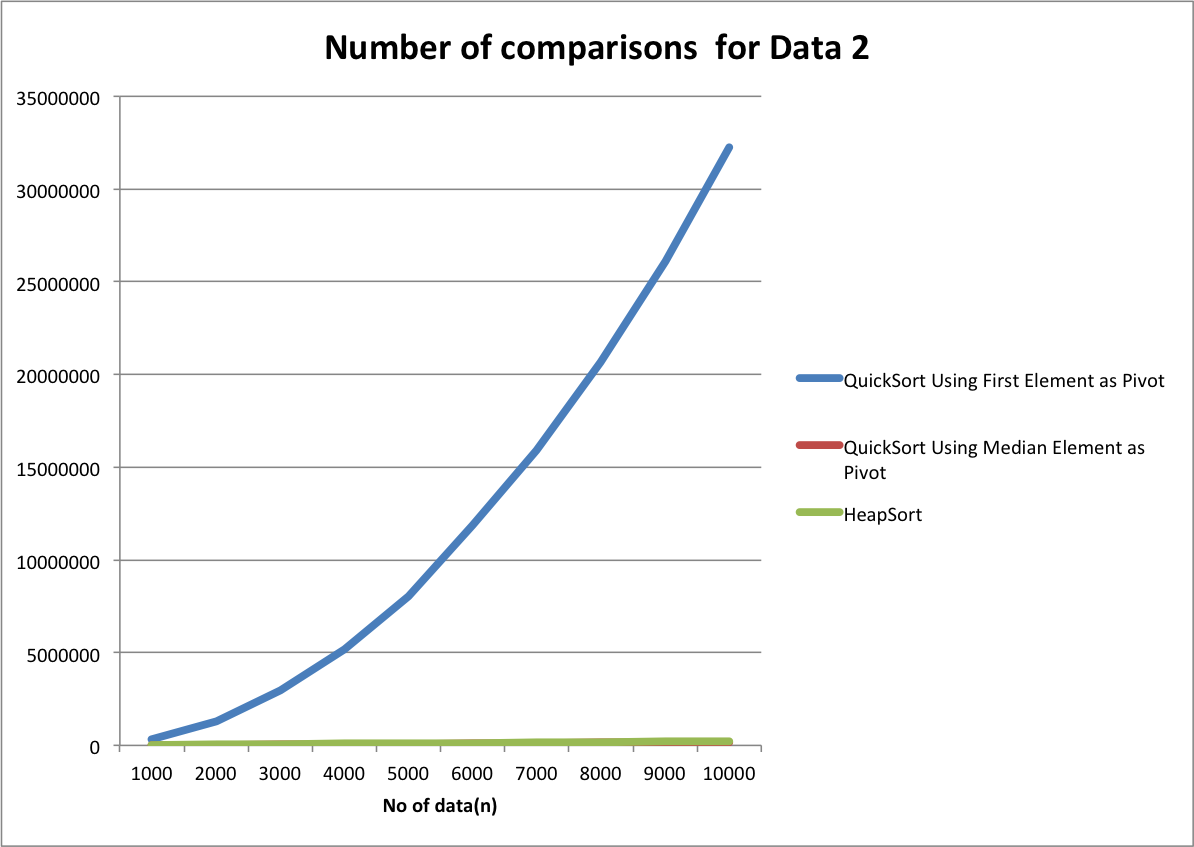
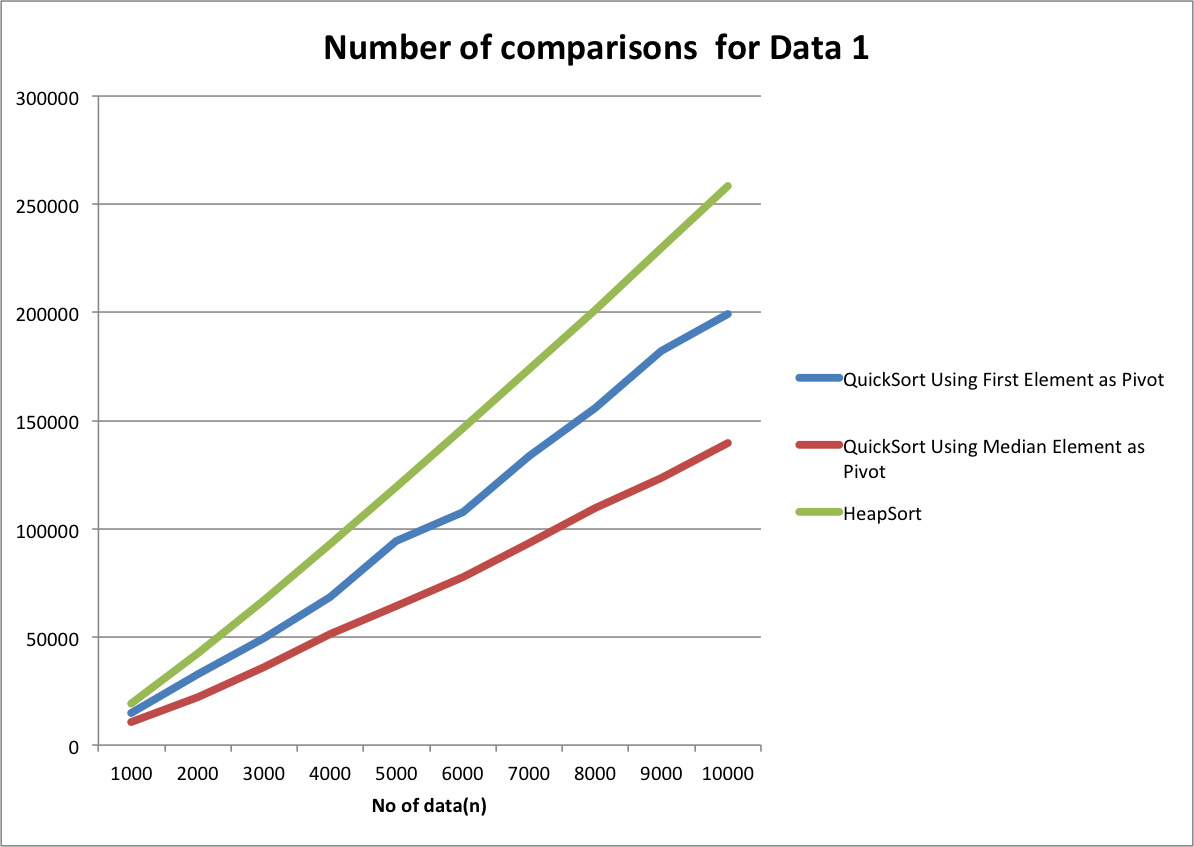
We can see that heapsort has a linear equation to the number of swaps and comparisons and this can be used to determine big O notation. Quicksort on the other hand has a variable big O notation due its recursive nature.

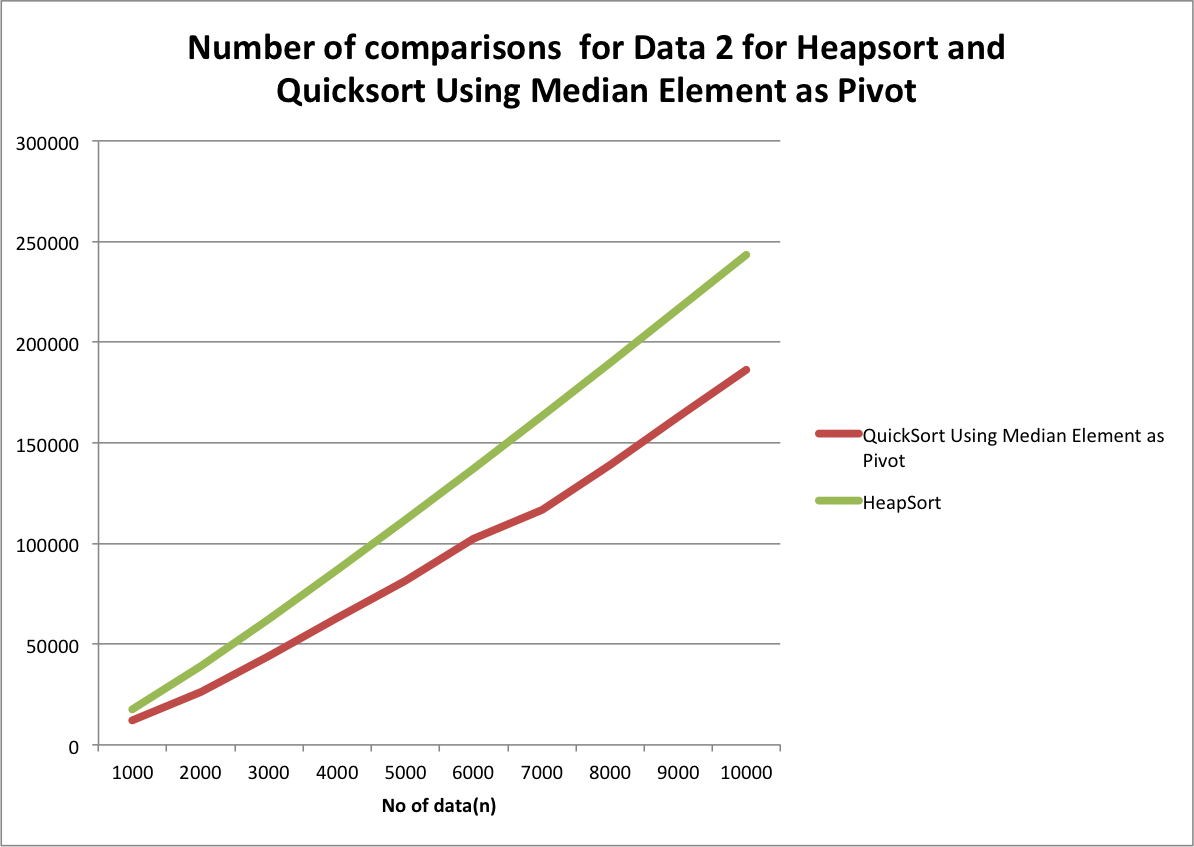
Quicksort first and median element average comparisons: O(n log n)

Quicksort first element worst comparisons: O(n2)

Quicksort median element worst comparisons: O(n log n)

Heapsort average and worst comparisons: O(n log n)

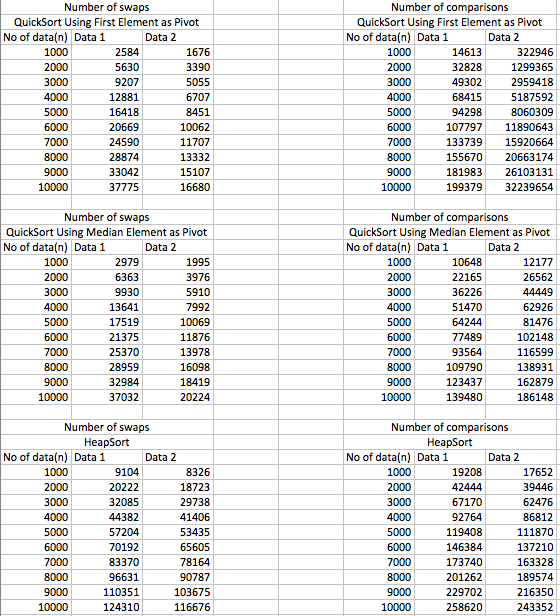




Here are comparisons of the 3 sorts combines to one graph for average case data. We can see here that heapsort has the greatest number of comparisons and quicksort using median element has the least. This is because quicksort is more efficient due to halving the size of the array to be sorted. The difference only being the coefficients of the O(n log n).

However, for worse case data, we can see that quicksort for first element has a lot more comparisons compared to the other two.

In conclusion, due to heapsort and quicksort average comparisons being O(n log n), the number of comparisons are quite close to each other with the average case data. However, worst comparison for quicksort using first element is O(n2) and quicksort using median and heapsort are O(n log n), the difference is clearly seen in the graph for data with worse case.



Data used in excel