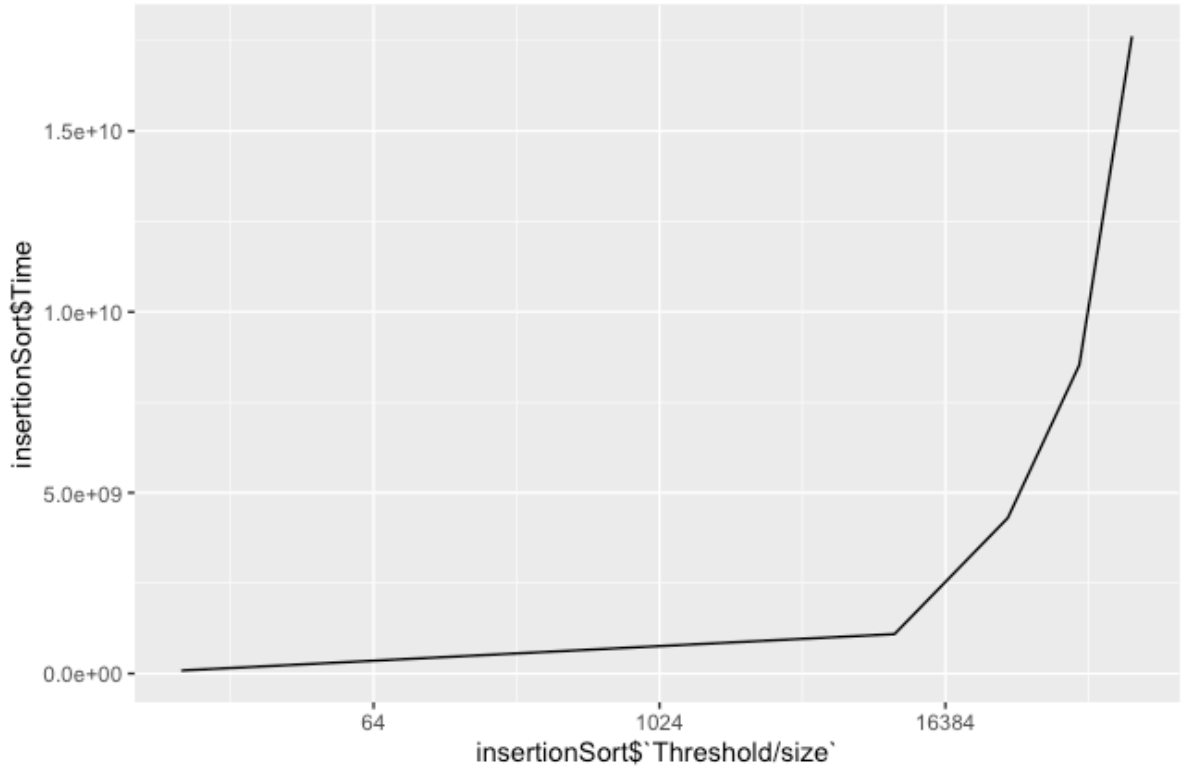
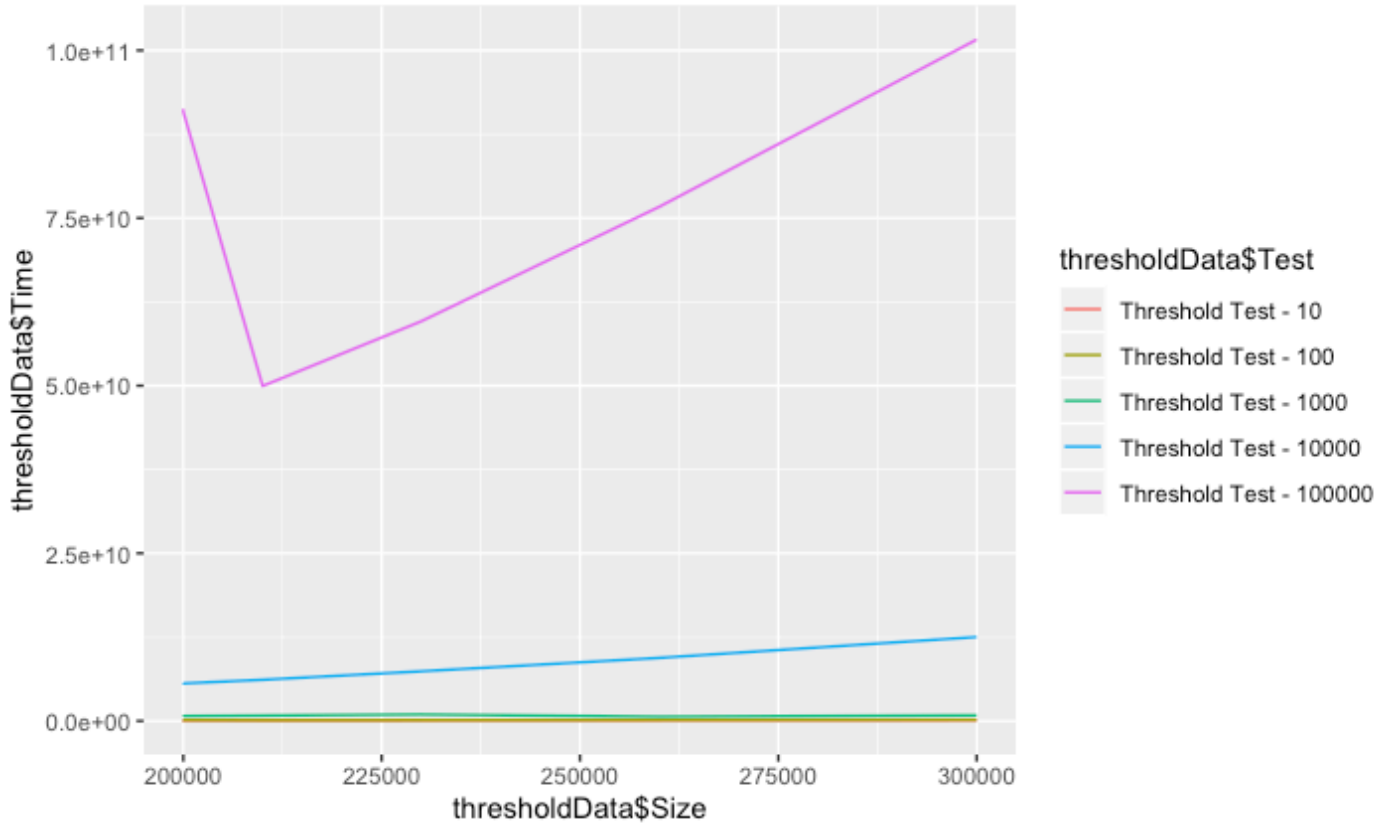
Mergesort Threshold Experiment: Determine the best threshold value for which Mergesort switches over to insertion sort.

The plot below was created by iterating over a range of threshold values where the threshold was systematically increased by 10,000 each iteration. A randomized or average case of the same size was generated each iteration to ensure that only increases in threshold were evaluated. The plot was log transformed on the x-axis, illustrating a rapid increase in the time to complete sorting when insertion sort is used above about 70,000 elements.

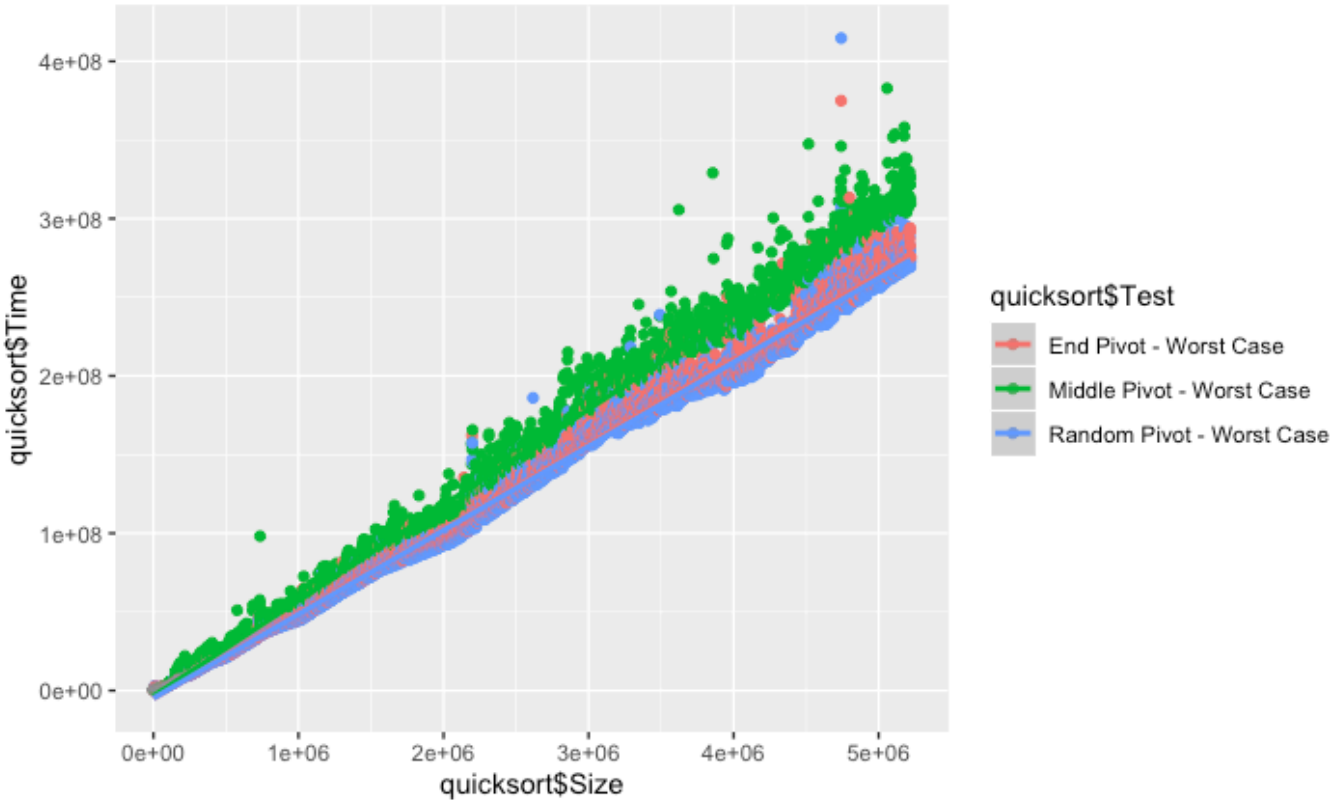


The following plot shows that the optimal location for the insertion threshold is between 10,000 and 100,000 elements. More resolution in the data would have been beneficial for this analysis.



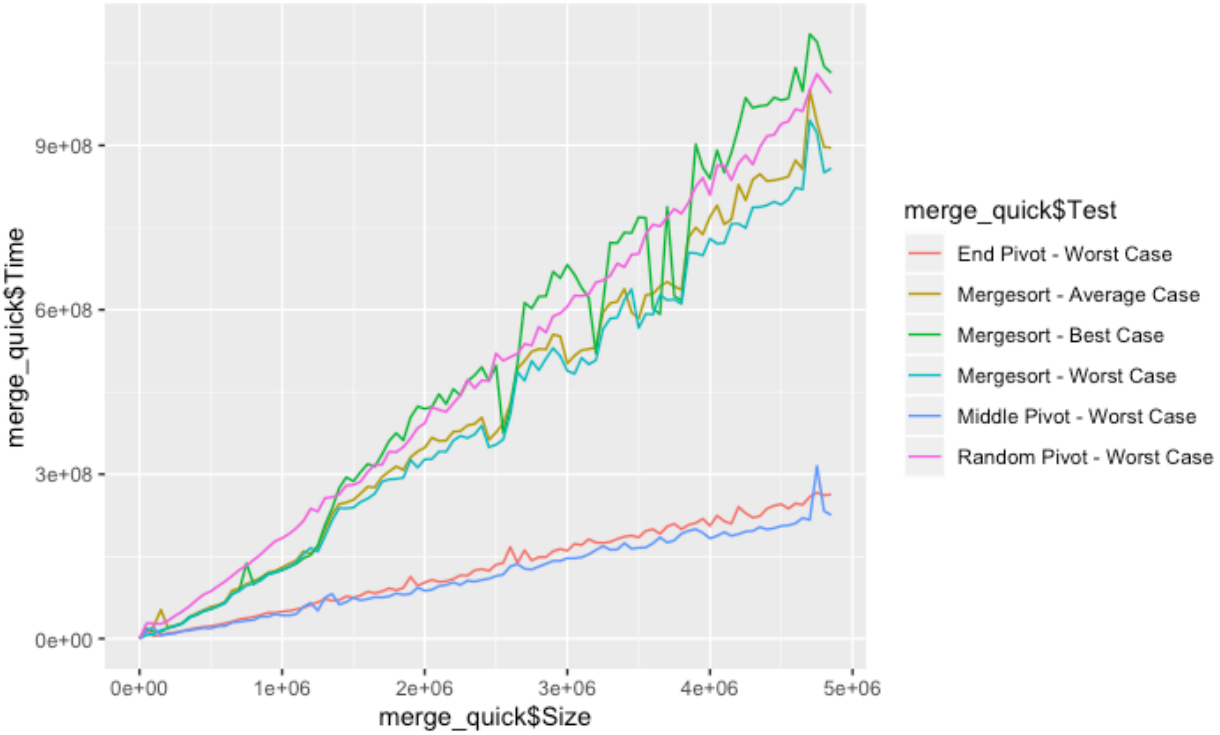
Quicksort Pivot Experiment: Determine the best pivot-choosing strategy for Quicksort.

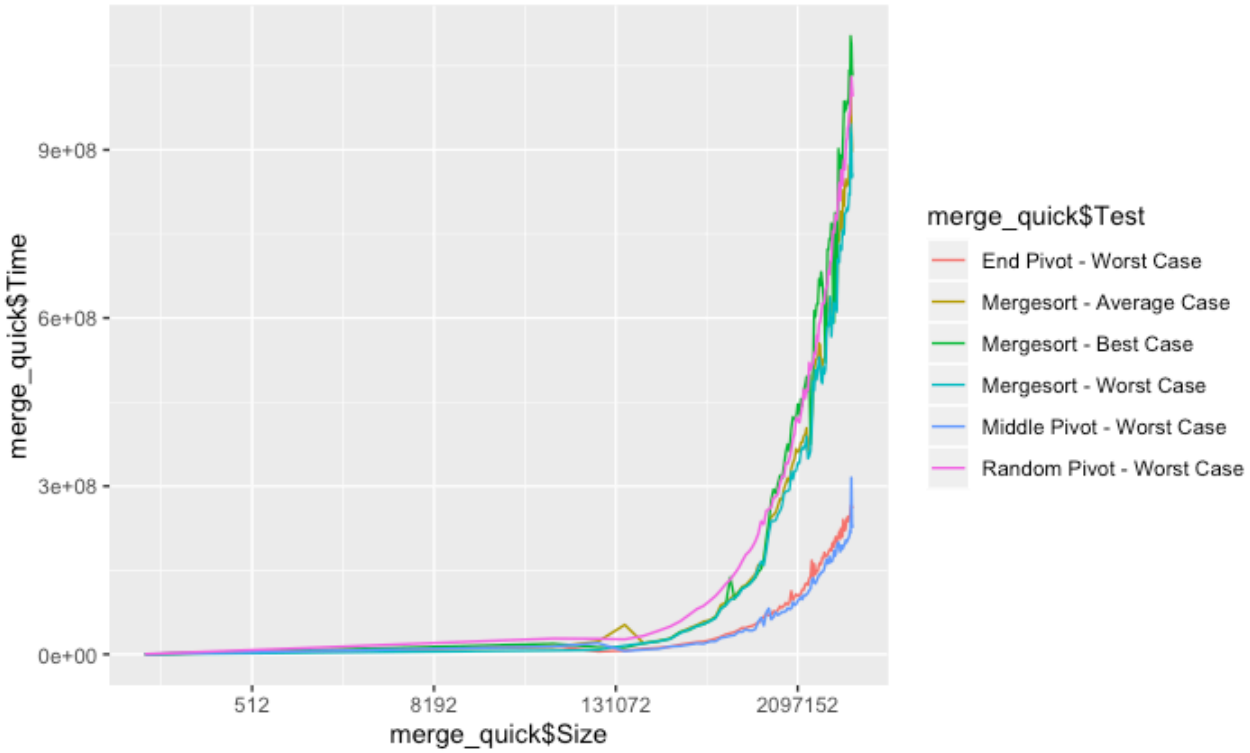
The plot below evaluates the time to sort varying sizes of arrays sorted in descending or, “worst case”, order to ascending order. It appears as though the middle pivot method under performed relative to the end pivot, which should never happen given the reverse sorted order of the elements in the array. Array sizes ranged from 100 to over 5 million and were incremented by 1000 with each iteration. One outlier was removed from the analysis in order to make the graph more presentable, and it was part of the middle pivot class.



Mergesort vs. Quicksort Experiment: Determine the best sorting algorithm for each of the three categories of lists (best-, average-, and worst-case).

The plots below compare the running times of the Quicksort and Mergesort algorithms against arrays sorted in ascending (best case), random (average case), and descending (worst case) orders. It appears as though my Quicksort algorithms are out performing my Mergesort algorithms for nearly every case. This may be due to the fact that I used a random pivot for the Quicksort analysis, not the end of the array. It also appears that based on the log transformed plot (below) my Mergesort is not achieving NlogN complexity. After the insertion sort threshold Mergesort begins to slow down rapidly. This may be due to memory constraints or how I implemented the algorithm, probably the latter.




9e+08
6e+08
3e+08
0e+00
merge_quick$Time
512
8192
131072
2097152
merge_quick$Size
merge_quick$Test
End Pivot - Worst Case
Mergesort - Average Case
Mergesort - Best Case
Mergesort - Worst Case
Middle Pivot - Worst Case
Random Pivot - Worst Case