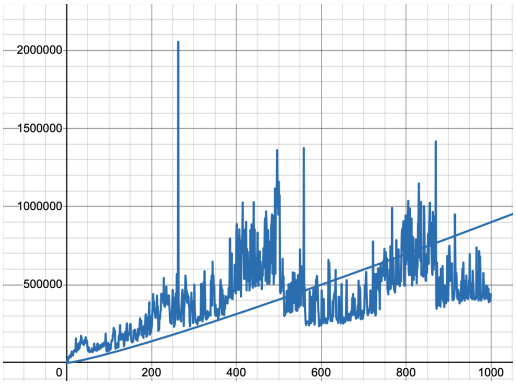
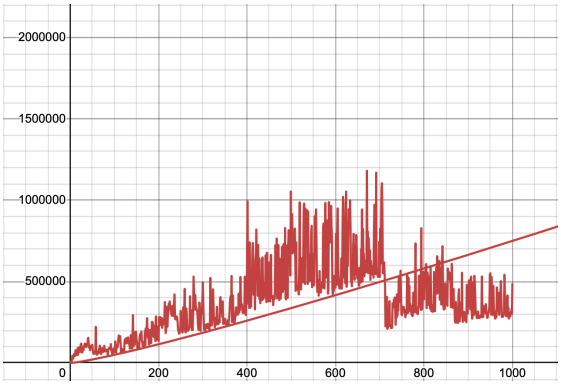
QuickSort Runtime Plots

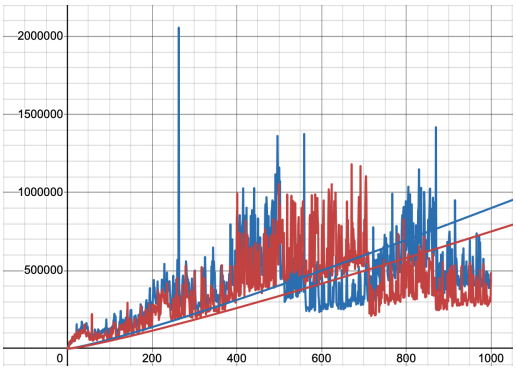
As shown by the graphs, both QuickSort functions tend to follow a trend function that is O(n log n), which is the time complexity determined in Homework 2 of both QuickSort algorithms. The Partition functions themselves were determined to each have O(n) time complexity, combined with the recursive QuickSort function to make for a total runtime of O(n log n). While, on average, the Separation Partition method was slightly slower than the Swap Partition method (the former had a trend function closer to 300n log n while the latter had 250n log n), both recorded runtimes were on order of n log n. So, in practice, while the swap version is slightly faster, as the input size increases, the two methods increase in runtime at the same rate (O(n log n)). The data for both graphs can be found in the separationTimes.txt and swapTimes.txt files respectively in CSV format. More data can be generated by running the program.



Separation QuickSort Runtime (vertical axis, nanoseconds) vs Array Size (horizontal axis, number of elements— 1 to 1000). The trend function is 300n log n and is shown as the smooth curve through the other points. Graphed using Desmos.



Swap QuickSort Runtime (vertical axis, nanoseconds) vs Array Size (horizontal axis, number of elements— 1 to 1000). The trend function is 250n log n and is shown as the smooth curve through the other points. Graphed using Desmos.



Overlay of both the Separation and Swap QuickSort Runtimes (vertical axis, nanoseconds) vs Array Sizes (horizontal axis, number of elements). Separation in blue and Swap in red. Graphed using Desmos.