Alex Chuoy alexchuoy@csu.fullerton.edu
Gregory Vasquez gvasquez11@csu.fullerton.edu
Paul Rodriguez senorrodington@csu.fullerton.edu

Project 2 Analysis

Analysis

Data Set 1: Alex

n = 5000	Merge Sort	Quick Sort
Min Performance	1.94567 sec.	0.251593 sec.
Max Performance	3.79181 sec.	1.54984 sec.
Range	1.84614	1.298247
Average Performance	2.10085888 sec.	0.560729468 sec.
Std Dev.	0.217187098 sec.	0.1267723707 sec.

Data Set 2: Greg

n = 5000	Merge Sort	Quick Sort
Min Performance	0.334933 sec.	0.0758519 sec.
Max Performance	0.80417 sec.	0.206701 sec.
Range	0.459237	0.1308491
Average Performance	0.411398406 sec.	0.1316182204 sec.
Std Dev.	0.03334348802 sec.	0.02695660304 sec.

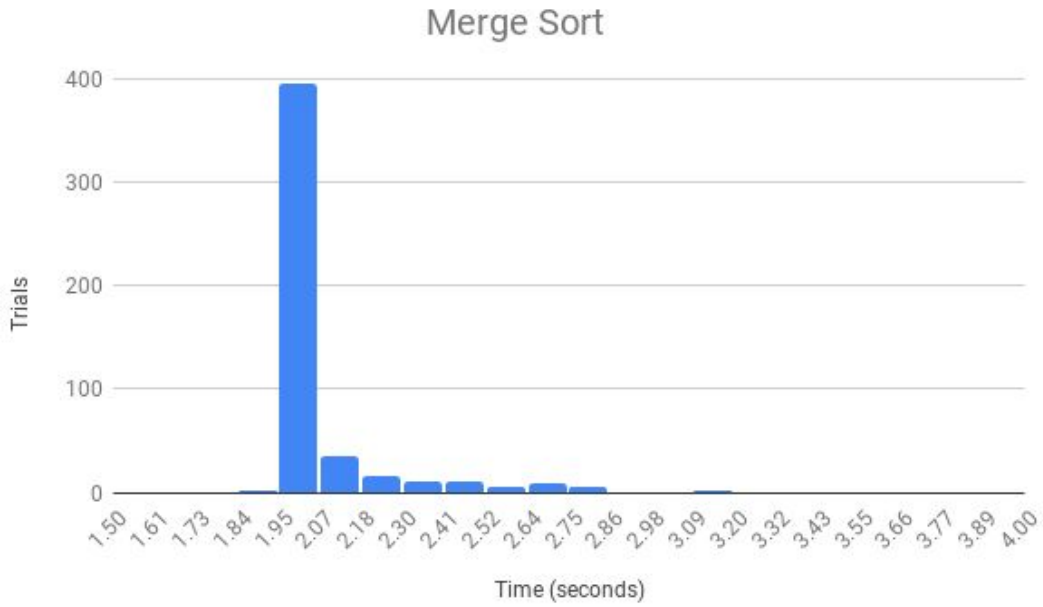
Data Set 3: Paul

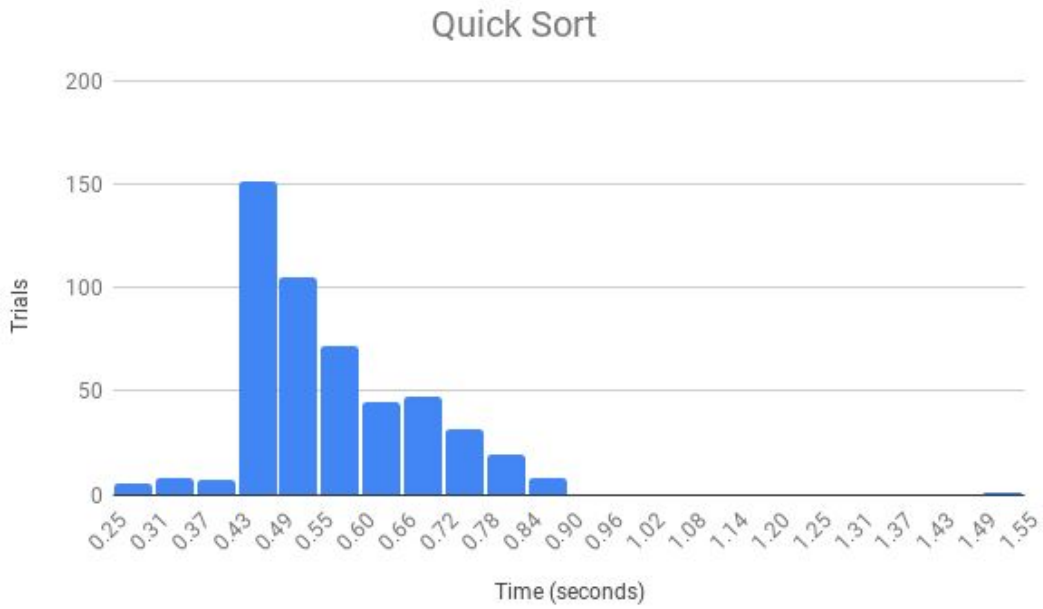
n = 5000	Merge Sort	Quick Sort
Min Performance	0.445844 sec.	0.163325 sec.
Max Performance	0.608765 sec.	0.676909 sec.
Range	0.162921	0.513584
Average Performance	0.515212022 sec.	0.378069976 sec.
Std Dev.	0.02479923459 sec.	0.08919160647 sec.

Alex Chuoy alexchuoy@csu.fullerton.edu
Gregory Vasquez gvasquez11@csu.fullerton.edu
Paul Rodriguez senorrodington@csu.fullerton.edu

Graphs

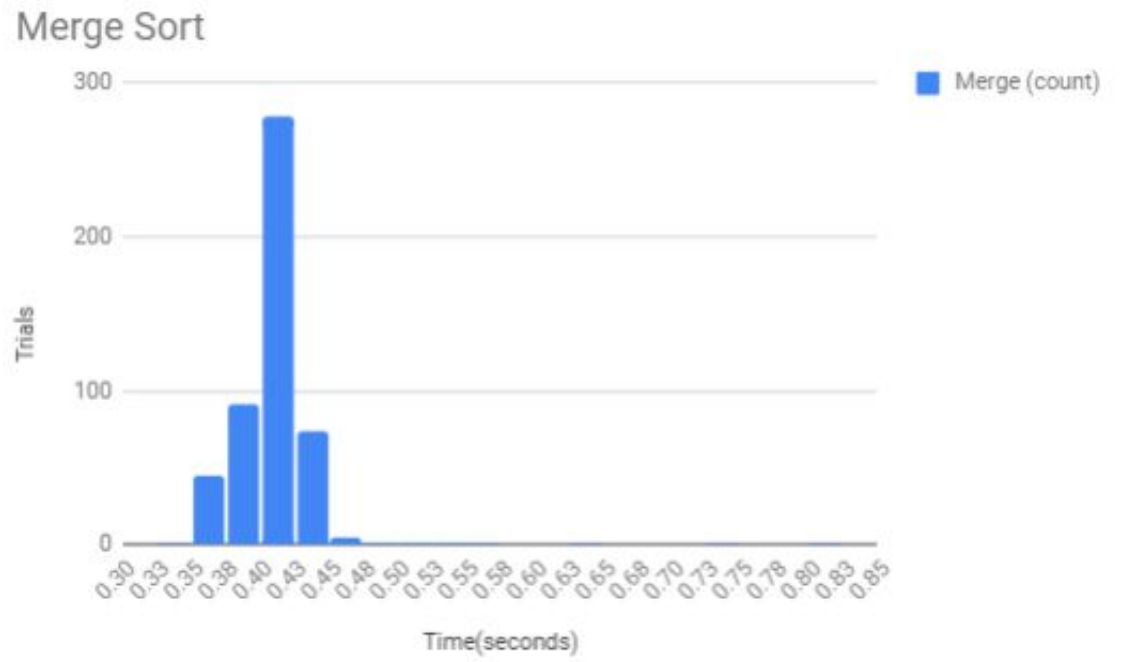
Data Set 1: Alex

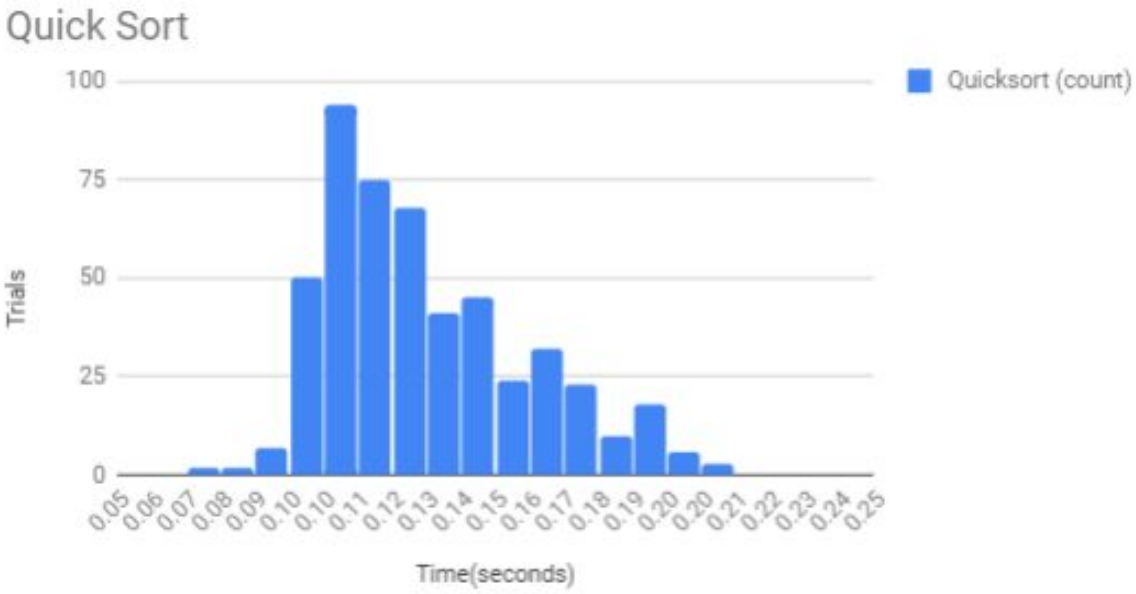




Alex Chuoy alexchuoy@csu.fullerton.edu
Gregory Vasquez gvasquez11@csu.fullerton.edu
Paul Rodriguez senorrodington@csu.fullerton.edu

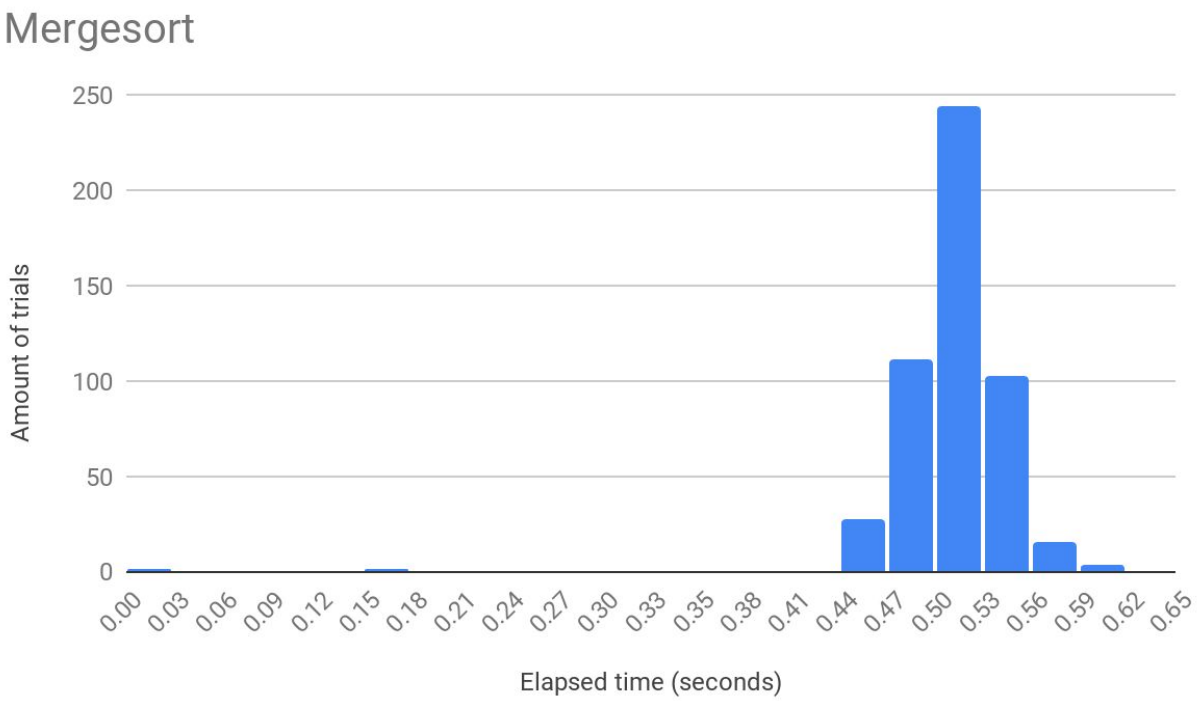
Data Set 2: Greg

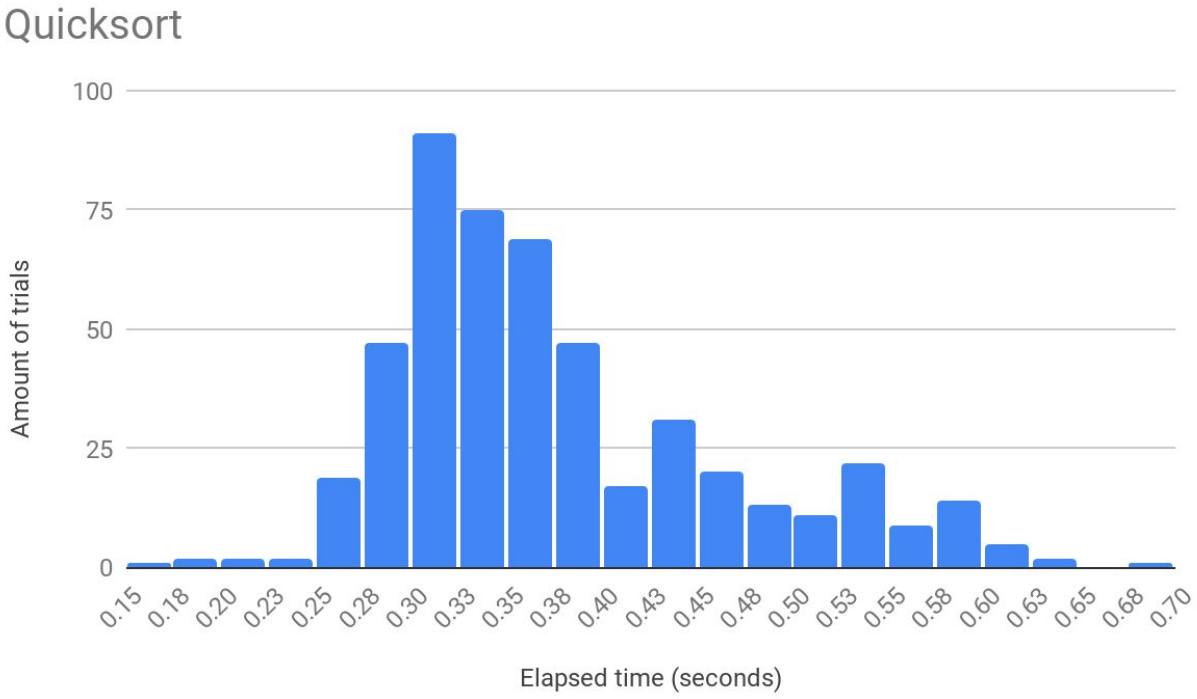




Alex Chuoy alexchuoy@csu.fullerton.edu
Gregory Vasquez gvasquez11@csu.fullerton.edu
Paul Rodriguez senorrodington@csu.fullerton.edu

Data Set 3: Paul





Alex Chuoy alexchuoy@csu.fullerton.edu
Gregory Vasquez gvasquez11@csu.fullerton.edu
Paul Rodriguez senorrodington@csu.fullerton.edu

Conclusion

Merge sort was very consistent since it took about 2-4 seconds for 400 of the 500 trials for Alex's data set, 0.4 seconds for 300 of the trials for Greg's data set, and 0.5 seconds for half of Paul's trials; it looks like randomization has very little impact on the performance of merge sort. Quick sort, on the other hand, had much more variable run times for it's 500 trials; most trials took at least 0.43 seconds on Alex's set, at least 0.10 seconds on Greg's and 0.30 seconds on Paul's while very few trials took less than that time. These trials seem to be randomized lists that were already well sorted.

For Alex's data set, merge sort had a larger range and standard deviation than quick sort, however most of the data for merge sort was concentrated into a thin slice of the graph while quicksort was much more spread out. Merge sort had a big outlier trial that took over 3 seconds.As a result, trials that took over about 2.3 seconds should also be considered outliers in merge sort since the bulk of the data took about 2 seconds. These outliers would have skewed the range and standard deviation of merge sort, therefore if these outliers were removed, the range and standard deviation will be less than the range and standard deviation of quick sort. Quick sort had an outlier trial that 1.5 seconds, but other than that there doesn't seem to be any major outliers within the data.

For Greg's data set, merge sort was much slower than quick sort as expected. The graphs for merge sort were much more concentrated on the front end of the graph around 0.35-0.45 seconds while the quick sort graphs is much more spread out. The standard deviation for both merge and quicksort were similar which tells us that each sorting algorithm was fairly consistent on their respective tests. Overall, merge sort had a much slower and higher range than quick sort as expected.

For Paul's data set, merge sort was slower on average than quick sort. However, mergesort had a considerably smaller range than quicksort. This is evident in the graphs as there are much less outliers in mergesort than quicksort and with the mode of the mergesort lying in the middle around 0.5 seconds while quicksort is much more frontloaded. This concludes that mergesort is much more stable than quicksort.

Quick sort over all took less time than merge sort, but that seems to be mostly a case of implementation. If quick sort and merge sort were implemented in way such that they had similar run times, we would see that merge sort would still be very consistent, and that quicksort could either take longer or shorter to run depending on what the list is.