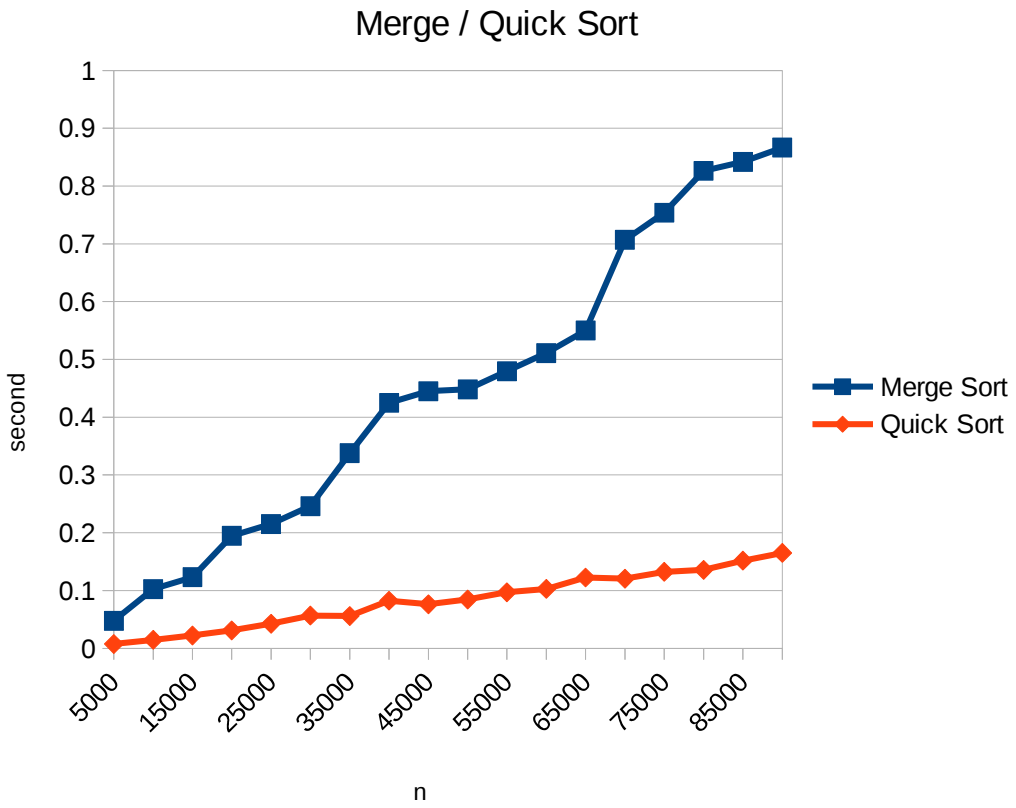
Name: Son Nguyen
Email: snguyen313@csu.fullerton.edu
Project #2: Merge sort and Quick sort

1 Plot and draw chart

n	Elapsed Time (sec)		Compare Quick/ Merge Sort
	Merge Sort	Quick Sort	
5000	0.0475248	0.00741819	15.61%
10000	0.102637	0.014629	14.25%
15000	0.123052	0.0223291	18.15%
20000	0.194585	0.031026	15.94%
25000	0.215059	0.0426009	19.81%
30000	0.245931	0.0565052	22.98%
35000	0.337586	0.0559638	16.58%
40000	0.424739	0.0823688	19.39%
45000	0.444991	0.0763649	17.16%
50000	0.448156	0.0845029	18.86%
55000	0.479541	0.097043	20.24%
60000	0.511006	0.102676	20.09%
65000	0.54998	0.122495	22.27%
70000	0.706988	0.120523	17.05%
75000	0.753815	0.132308	17.55%
80000	0.826313	0.135855	16.44%
85000	0.84176	0.151667	18.02%
90000	0.866507	0.164987	19.04%



- Looking at the above chart, we can see elapsed time of merge sort is represented by an linear line which efficiency is almost O(nlogn) as stated by the lecture
- For Quick sort, even though the lecture says that its efficiency is O(nlogn) for the best case,

and O(n^2) for the worst case, the chart and the data table shows that its elapsed time is just around 20% of merge sort

2 Calculate average and Standard Deviation, and analyze the efficiency

n	Merge Sort Elapsed Time (sec)	Quick Sort Elapsed Time (sec)
90000	0.575697	0.11988
90000	0.568006	0.126865
90000	0.564319	0.117663
90000	0.573233	0.118006
90000	0.594509	0.130277
90000	0.604516	0.142618
90000	0.601088	0.117737
90000	0.593053	0.119319
90000	0.593821	0.118751
90000	0.584646	0.122171
90000	0.602053	0.11948
90000	0.570258	0.120432
90000	0.578581	0.119678
90000	0.565514	0.120895
90000	0.573772	0.120264
90000	0.575847	0.120658
90000	0.576063	0.122117
90000	0.569005	0.119626
90000	0.579104	0.121096
90000	0.605777	0.12563
90000	0.587803	0.120809
90000	0.573682	0.122462
90000	0.569907	0.119072
90000	0.606939	0.124767
90000	0.575485	0.117663
90000	0.568097	0.123183
90000	0.571038	0.121529
90000	0.573803	0.121703

n	Merge Sort Elapsed Time (sec)	Quick Sort Elapsed Time (sec)
90000	0.565126	0.124167
90000	0.564772	0.126764
90000	0.569254	0.125395
90000	0.577639	0.121786
90000	0.570917	0.12021
90000	0.564934	0.122164
90000	0.577423	0.122389
90000	0.572948	0.122765
90000	0.566521	0.118248
90000	0.578256	0.1243
90000	0.580672	0.119954
90000	0.573408	0.120967
90000	0.56444	0.128691
90000	0.563807	0.122879
90000	0.59262	0.121782
90000	0.569657	0.116042
90000	0.575494	0.124625
90000	0.566829	0.124212
90000	0.571044	0.124663
90000	0.571948	0.121569
90000	0.57208	0.123218
90000	0.574358	0.11876
Average	**0.57719526**	**0.12219802**
Best	**0.563807**	**0.116042**
Worst	**0.606939**	**0.142618**
Stdev	**0.011894548**	**0.004161196**
Best / Avg	**0.976804626**	**0.949622588**
Worst / Avg	**1.051531504**	**1.167105654**
Worst / Best	**1.076501356**	**1.229020527**

- The two algorithms, merge sort and quick sort, are run 50 times for n = 90000

MERGE SORT

- As shown in the above table, Average of elapsed time of merge sort for n = 90000 is ~0.058s, and the standard deviation is ~0.012 which is very small. This means the values of elapsed time are very convergent to the average value. This proves what the lecture said: the efficiency of the merge sort is O(n*logn) for both best and worst case. This efficiency is very consistent
- In addition, let's look at the ratios of Best and worst cases over the average, the best case is 0.97, and the best case is 1.05 of the average which again states that the merge sort algorithm's efficiency is very consistent

QUICK SORT

- For quick sort, even though the lecture said that the best case efficiency is O(n*logn) which is the same with merge sort, we can see quick sort is a lot faster than merge sort (its average is just ~21% compare to merge Sort.) This is a lot better than O(n*logn)
- Comparing the worst and best cases shows that the worst case is only 23% worser than the best case while Lecture states the percentage is 39%
- While the lecture says that the worst case is O(n^2), we don't see any value of O**(**n^2**)** in the sample