Ria Mukherji
Dr.Lin
Algo 3353
18 February 2024

NOTE: in some of the graphs the red line representing quickSort is obscured by the orange line representing quickSort2. It is there if you look close, just hard to see.

Average Comparisons:

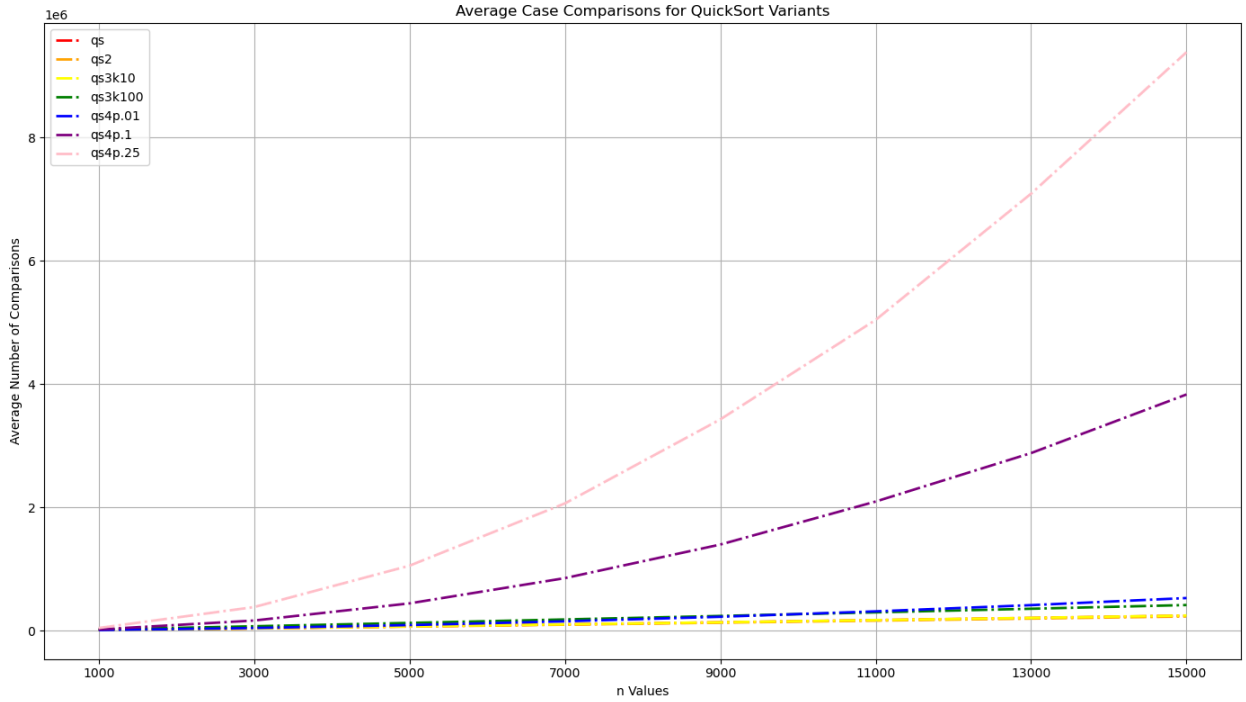
			Average Comparison				
n	qs	qs2	qs3k10	qs3k100	qs4p.01	qs4p.1	qs4p.25
1000	10987.54	10882.55	11142.19	22488.5	11142.19	22488.5	45725.69
3000	39592.09	38281.86	40056.4	73899.51	45368.91	166917.6	385190.8
5000	71212.65	68372.1	71989.1	128397.3	93201.46	445206.4	1056412
7000	104235.6	99620.54	105322.9	184519.3	154068	854882.6	2066515
9000	138043.3	131704.2	139439.3	241246.8	228021.4	1399578	3429681
11000	173366	165010.3	175069.3	299691.6	315866.7	2097570	5044515
13000	209020.1	198742.9	211033.4	358216.3	416645.7	2883631	7092248
15000	246434.6	232743.3	248774	418445.8	531710.8	3834410	9386853

Standard Deviation of Comparisons:

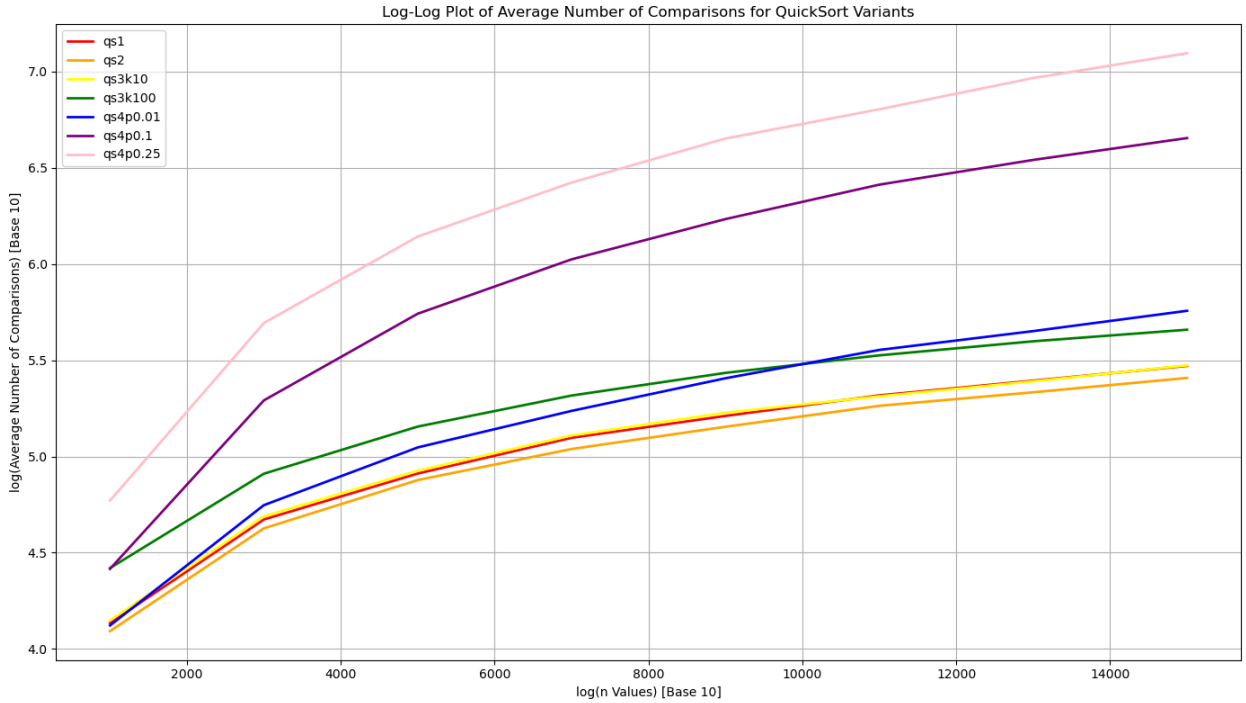
			Standard Deviaton				
n	qs	qs2	qs3k10	qs3k100	qs4p.01	qs4.1	qs4.25
1000	610.831	346.6815	610.8626	1145.661	610.8626	1145.661	4377.01
3000	1966.49	1041.71	1964.498	2760.242	1939.351	10316.45	40011.47
5000	3344.842	1760.319	3342.063	4129.32	3410.066	28363.29	118040.5
7000	4441.441	2483.72	4446.841	5188.178	4802.75	56375.38	236970
9000	5701.052	2774.635	5700.881	6333.288	6168.875	95792.02	383983.4
11000	7041.47	4121.979	7046.701	7698.029	8158.889	135330	540239.5
13000	8462.476	4378.556	8463.173	8955.775	9821.917	198765	792475.9
15000	9240.233	5048.738	9232.888	10117.23	12218.89	264419.8	1057482

Graphs Below:

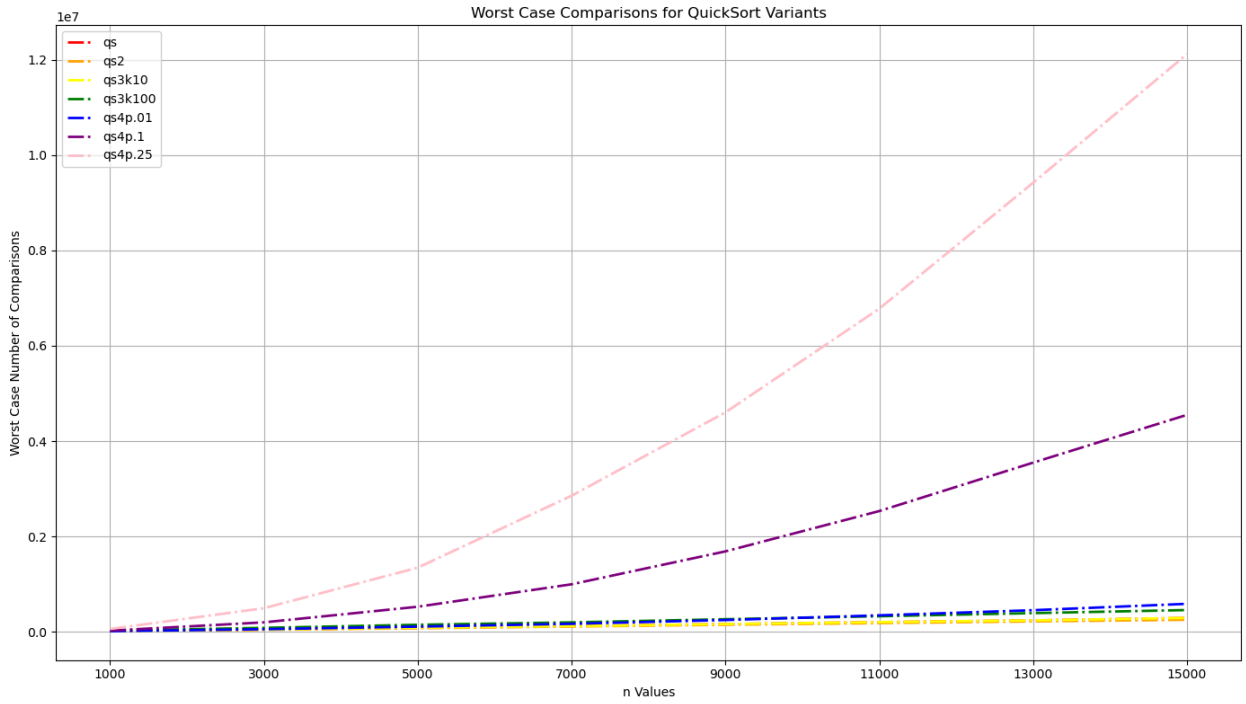
Graph 1:



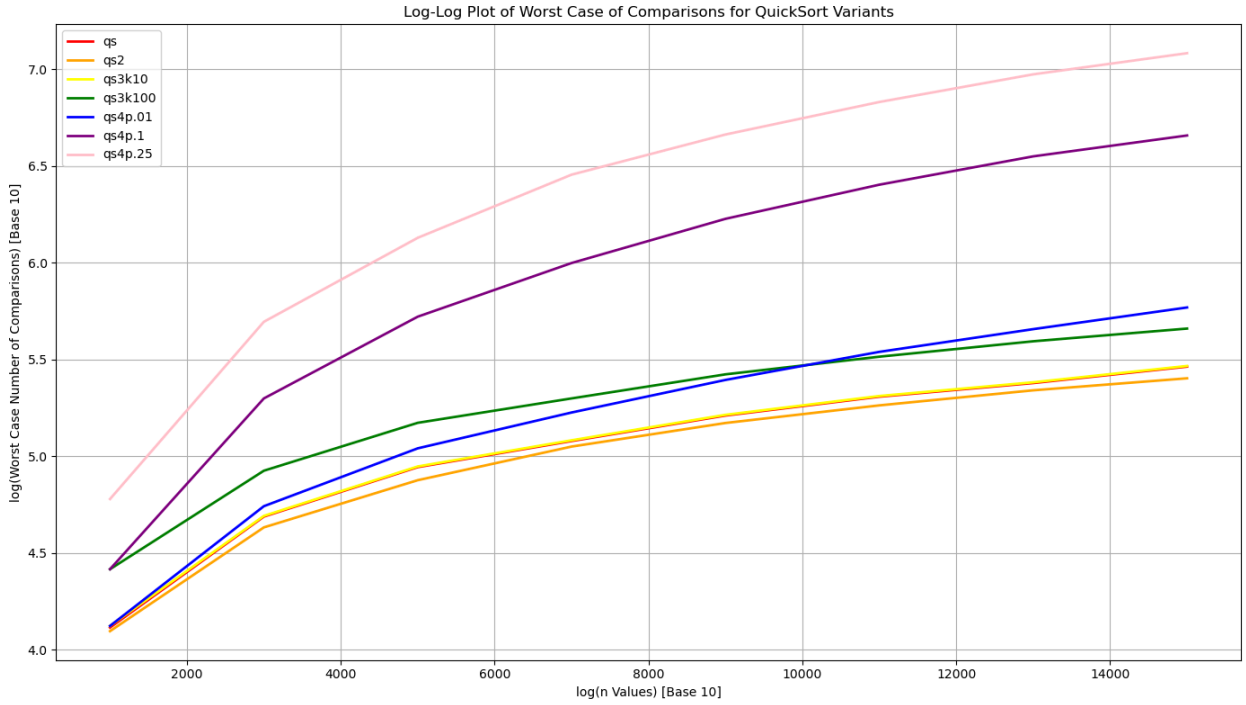
Graph 2:



Graph 3:


Worst Case Comparisons for QuickSort Variants
1e7
qs
qs2
qs3k10
qs3k100
qs4p.01
qs4p.1
qs4p.25
Worst Case Number of Comparisons
n Values
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1000
3000
5000
7000
9000
11000
13000
15000

Graph 4:


Log-Log Plot of Worst Case of Comparisons for QuickSort Variants
qs
qs2
qs3k10
qs3k100
qs4p.01
qs4p.1
qs4p.25
log(Worst Case Number of Comparisons) [Base 10]
log(n Values) [Base 10]
4.0
4.5
5.0
5.5
6.0
6.5
7.0
2000
4000
6000
8000
10000
12000
14000

.

8.

quickSort2 is the best option in terms of efficiency and speed due to its pivot being chosen as a median number. Although it only finds the median of the last 3 elements in a vector and not the whole vector, that is still enough to make a significant difference without a lot of overhead and excess comparisons. quickSort closely follows quickSort2 in Average comparisons and in some very small arrays may take less comparisons due to the fact that it doesn't need to compare to choose a pivot at each recursive call; however, if using more data, then quickSort2's ability to check the median is vital to ensure we are not wasting lots of comparisons on inefficient pivots. Although their averages are relatively similar it is evident in their worst case scenarios that quickSort2 can save you a lot of unnecessary comparisons if you choose the wrong pivot several times (i.e. a completely reversed vector). I would recommend using quickSort2 in most cases, especially if working with large data sets. quickSort3 is not any better than the original or 2. Setting the k value to 10 is alright for the average case, but becomes worse during the worst case. Setting k to 100 results in significantly worse average and worst comparisons. I would not recommend using quickSort4 unless you use a very small data set with a very small p value. Otherwise the 4th algorithm progressively worse in both average and worst case comparisons the larger the p value gets.

Asymptotically the graphs are quite interesting. all the quickSort functions with the exception of 4 tend to level off at around 5 comparisons on the log scale both for worst and average cases. quickSort4 on the other hand continues to grow. One thing that is interesting is that quickSort4 with a small p and n value actually are quite efficient when the n value is low, and even beats using quickSort3 before surpassing it until around the log(1000) mark for n. No graph levels off completely asymptotically but the functions other than 3 and 4 tend to level off lower than the others which is beneficial. I think the key takeaway from having average case next to worst case is the ability to truly see the benefits of using a quickSort with a procured median.