Marietta Asemwota

Implementing QS1 was easy enough. The algorithm from the powerpoint was pretty much it. There was an error in there at some point where my QS1 would not work for numbers larger that 10,000. I tinkered with my code and fixed the problem. My populate method had something to do with it.

QS2 was okay to implement using just random partition, but when I had to change it to call select() instead, the code kind of broke. There was some weird thing with the select method where it compared the elements at the position in the array instead of comparing the indices. I eventually got it to work correctly after changing little things with the code.

I tried to do the extra credit, which is finding the medians of the medians, but I couldn't really figure it out. But I left my pseudocode in there.

I created my own Junit test class. My code passes all my tests. My code also passes all the tests from the provided Junit test class. I also have a commented out main method in my Junit test class that runs the time and comparisons for QS1 and QS2. The results are reproduced in the chart below:

Run time measurements:

Array size	10,000	100,000	1,000,000	10,000,000	100,000,000
Total time (s) of QS1	0	0	0	1	19
Total time (s) of QS2	0	0	0	3	43
Comparisons in QS1	157653	2142210	25757006	298455347	3405710734
Comparisons in QS2	327819	4471420	56453897	697910724	7866674699

Marietta Asemwota

QS2 takes longer than QS1 because even though they are both order of (lgn), they have very different constants. QS2 has a higher constant than QS1, so it takes longer. That is why QS2 is not more efficient than QS1 like we thought before.