Optimized Quick Sort:

I decided to include an insertion sort method for my optimized quick sort function. The reason for this was I realized that insertion sort is good at sorting small arrays. Therefore, I decided on a threshold that would stop recursion once the array size was less than or equal to size 10. After recursively splitting the array with a pivot, eventually the array size would reach that threshold and call insertion sort which best case would sort the 10 elements in O(n) time. This small adjustment made sorting larger arrays much faster and more efficient. It did however affect the sorting time when the array was inversely sorted. Originally, I tried to enhance the performance of my function by creating my own stack, sorting smaller arrays with special cases, and tweaking my insertion sort method. However, even though my function was much faster, it was very inconsistent and inaccurate. Creating my own stack was a little more tricker than I had thought and the special cases I had created for really small arrays was too much of a task. Tweaking my insertion sort method enhanced the performance by a fraction of a millisecond and once the array size was very large, it caused the sorting to be very inaccurate. I attempted to adjust the partition method but I felt that it was a little overwhelming at times since the output was never what I wanted it to be.