Big O notation(complexity) analysis of Sorting Algorithms

For this analysis we were asked to implement four different sorting algorithms, namely, bubble, selection, insertion, and quick. Each algorithm operates differently.

Bubble for example with $O(n^2)$ complexity goes through an array and checks adjacent values swapping them if the left value is larger than the right and goes through until it doesn't swap any values. This is very inefficient to use in most settings.

However, both the selection and insertion algorithms also have $O(n^2)$ complexity. While they go about sorting better than their friend bubble by leaving holes where an element was copied out of and replacing it with different values depending. While the complexity isn't any better for these algorithms, the speed is faster than bubble sort.

And finally, we implemented quick sort which partitions an array into two halves and uses recursion to keep splitting up the halves and with the recursion it sorts these little pieces and brings it all together in a fraction of the time. Quick sort has complexity O(n*log(n)), making it the least complex. However, in terms of memory quick sort can cause a core dump easily because of its use of recursion.