Algorithm Explanation

Insertion Sort

* The insertion sort algorithm sorts an algorithm in increasing order. It does this by reading the array starting from the first value and compares each value (i) to the value that precedes it (i-1). If (i-1) > (i), the two swap positions in the array. This process will continue until the array is fully sorted.
* Insertion Sort has a worst case time complexity of O(n^2)

Bubble Sort

* The bubble sort algorithm works very similarly to the insertion sort algorithm in that it repeatedly makes a comparison between two neighboring values in the array going from left to right until the array is sorted (starting from the second index however). The main difference is that the comparison is being made between the target value (i) and the value to the right of I (i + 1) rather than the left in this algorithm. Because of this, the lower terms will slowly rise or “bubble” to the beginning of the array hence the name bubble sort.
* Bubble Sort has a worst case time complexity of O(n^2)

Merge Sort

* The merge sort algorithm is unique from the previous sorting algorithms in that it sorts an array recursively. It does this because it separates the array into halves, and further until each value is on its own. It then takes a pair of values (starting from the beginning) and sorts them, similar to what a bubble sort does. Finally, it takes all of the smaller sections, and merges them into one sorted array.
* Merge Sort has a worst case time complexity of O(nlogn)

Quick Sort

* Being the fastest algorithm, Quick Sort works by selecting a pivot value from one of the numbers in the array, and makes a comparison between the first value from the left that is greater than the pivot and the first value from the right that is less than the pivot. Once the comparisons cannot go any further, the array is partitioned into multiple halves that will go through the same process until the whole array is sorted.
* Quick Sort has a worst case time complexity of O(n^2)