6.

**Bubble Sort:**

The Bubble Sort works by comparing each element in the array with the element after it. If the element after is less than the current element, the two are swapped. A nested loop is used for this comparison.

Best case:

**If the bubble sort has been coded to do so,** the best possible case can be O(n). A “number sorted” variable can be added to a bubble sort to check if the list is already sorted. If the number of values sorted is zero at the end of a pass, the sorting can end. If the array is already sorted, a bubble sort with this functionality would end after the first pass resulting in O(n) time complexity.

Since the provided implementation does not contain this functionality, the nested loop will still check each element. Therefore, the time complexity will be O(n^2). I did not want to change the functionality of the bubble sort, as I understood the assignment prompt to only want the exact implementations provided.

Worst case:

If the bubble sort is provided a list in descending order, the if statement will always be true and it will always need to swap values. This would be n swaps for n iterations. Therefore, the time complexity would be O(n^2).

**Quick Sort**

The Quick Sort works by splitting the input array on a pivot, where everything before the pivot is less than the pivot value and everything after the pivot is equal or greater than the pivot value. Each split results in a new “level”, where each element of the array is visited once. The last level is reached when all arrays have reached size 1.

Best case:

The Quick Sort is finished when no subarrays can be split further. If the pivot is perfectly in the middle of the subarrays each time, the least amount of "levels" will have to be made. The amount of levels is equal to how many times the array can be divided. In the best case, this would be equal to Log2(n) (for an array of size 8, there would be 3 levels.) Therefore, the time complexity would be N \* log(n) or O(n log(n)).

Worst case:

If the pivot is always the least or greatest number in the array (on the edge), then there would need to be n levels until all branches reach a length of 1 (for an array of size 8, 8 levels would be needed). Therefore, the worst time complexity would be N \* N or O(n^2).

7.

The following times were collected:

**1,000**

Recursive quick sort: 0.0015038

Non Recursive Quick Sort: 0.0000455

Bubble Sort: 0.0014758

**10,000**

Recursive quick sort: 0.1764133

Non Recursive Quick Sort: 0.0006014

Bubble Sort: 0.1794773

**100,000**

Recursive quick sort: 16.5537539

Non Recursive Quick Sort: 0.0422195

Bubble Sort: 16.4571430

**1,000,000**

Recursive quick sort: 03.6684266

Non Recursive Quick Sort: 02.4996222

Bubble Sort: 29:58.1884710

**10,000,000:** Did not finish due to time constraints and needing to use computer for other homework.

8b. In 30 minutes, the most amount of characters my Monkeys Writing Shakespeare Simulation was able to guess was 16 out of 64. (Correct results were not saved, 16 out of 64 was in one “guess”).