Project 5 - Sorting Algorithms

1 - We make a new parameter equal to A + B and call it C. This will make the new called sorted sequence merge(A, B, C). Since A and B are sorted sequences, let us assume their lengths are n1 and n2. For the merge sort to form a new sorted sequence, it will take O(n1 + n2). We then have to do a linear scan through the sequence C by removing all of the duplicate elements. We check the current element with the next element. If they are equal, we remove the duplicate element. The total time taken will be O(n1 + n2 + n1 + n2) which is equal to O(n) for the big values of n where n is the size sequence of C. This is how we get the sorted sequence in O(n) time.

2 - We have to recurse on n - 1 elements at each step for the worst case of quick sort to occur. When we choose the middle element to be the pivot, we want all the other elements to be greater than or equal to the pivot element. Since we are doing it recursively, the sequence must be such that the middle element is lesser than all of the elements and this property is recursively followed for both halves of the sequence of length greater than 2.

3 - Using the merge sort, its best, worst, and average time complexity is the exact same. This means that it must take O(nlogn) time. Using the quick sort, if the array is already sorted and we are choosing pivot as the last element, then this is the worst case scenario of the quick sort. Because it partitions the array such that one partition is empty and the other array contains (n - 2) elements. That means that worst case time complexity is O(n^2). The best and average case would be the time complexity of O(nlogn).

4 - First, you sort the array using a merge sort or a quick sort. The time complexity of this would be nlogn. Then, you traverse the array for finding duplicates and putting the unique elements into a separate array. The time complexity of this would be O(n). Combining the two time complexities together results in O(nlogn).

5 - After the quick partitioning procedure, if the middle element is placed at the ending or starting position of the list. Then, the equation is $T(n) = O(n) + T(n - 1)$. This means that the time complexity is $O(n^2)$.