Problem 1: SquareRoot of An Interger
Time efficiency: O(log(n))
Space efficiency: O(1)
Function: sqrt is the function I'm using to get worse case for both space and time complexity.

Problem 2: SearchInRotatedArray
Time efficiency: O(log(n))
Space efficiency: O(1)
Function: Using Binary search function to get worse case for time and space.

Problem 3: ReArrangeDigits
Time efficiency: O(n*log(n))
Space efficiency: O(n)
Function: Using merge function to get time and space complexity.

Problem 4: DutchNationalFlag
Time efficiency: O(n)
Space efficiency: O(n)
Function: Using sort_012 function to come up with time and space complexity.

Problem 5: AutoComplete
Time efficiency: O(n)
Space efficiency: O(n^2)
Function: Using findSuffix for time complexity. When I'm thinking of worst case for space I'm generally think about how nested the TrieNode.Children is.

Problem 6: Unsorted IntergerArray
Time efficiency: O(n)
Space efficiency: O(n)
Function: Using get_min_max function to come up with time and space complexity.

Problem 7: HttpRouterTrie
Time efficiency: O(n)
Space efficiency: O(n^2)
Function: General insert in RouteTrie takes O(n) that is what I'm using for worst case in time. When thinking Space complexity I put O(n^2) because of the nature of nested Node.children.