MEDIUM QUESTIONS:

1. **Two Sum**: Given an array of integers, return indices of the two numbers such that they add up to a specific target.
2. **Reverse a String**: Write a function that reverses a string in-place.
3. **Palindrome Check**: Determine whether a given string is a palindrome.
4. **FizzBuzz**: Print numbers from 1 to n, but print "Fizz" for multiples of 3, "Buzz" for multiples of 5, and "FizzBuzz" for multiples of both.
5. **Merge Two Sorted Lists**: Merge two sorted linked lists and return it as a new sorted list.
6. **Find the Missing Number**: Given an array containing n distinct numbers taken from 0, 1, 2, ..., n, find the one that is missing from the array.
7. **Binary Search**: Implement the binary search algorithm for a sorted array.
8. **Anagram Check**: Check if two strings are anagrams of each other.
9. **Maximum Subarray**: Find the contiguous subarray within an array (containing at least one number) that has the largest sum.
10. **Valid Parentheses**: Given a string containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.
11. **Reverse Linked List**: Reverse a singly linked list.
12. **Count and Say**: The count-and-say sequence is a series of numbers with the first five terms as following: 1, 11, 21, 1211, 111221, ... Find the nth term.
13. **Longest Common Prefix**: Find the longest common prefix string amongst an array of strings.
14. **Merge Intervals**: Given a collection of intervals, merge overlapping intervals.
15. **Rotate Array**: Rotate an array to the right by k steps.
16. **Linked List Cycle**: Given a linked list, determine if it has a cycle in it.
17. **Valid Sudoku**: Determine if a 9x9 Sudoku board is valid.
18. **Majority Element**: Find the majority element in an array.

19. **Linked List Cycle II**: Given a linked list, return the node where the cycle begins. If there is no cycle, return null.
20. **Search in Rotated Sorted Array**: Search for a target value in a rotated sorted array.
21. **First Missing Positive**: Given an unsorted integer array, find the smallest missing positive integer.
22. **Container With Most Water**: Given n non-negative integers representing the height of each bar in a bar chart, find the area of largest rectangle that can be formed.
23. **Add Two Numbers**: You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order and each of their nodes contain a single digit. Add the two numbers and return it as a linked list.
24. **Letter Combinations of a Phone Number**: Given a string containing digits from 2-9, return all possible letter combinations that the number could represent.
25. **Roman to Integer**: Convert a Roman numeral to an integer.
26. **Integer to Roman**: Convert an integer to a Roman numeral.
27. **Longest Valid Parentheses**: Given a string containing just the characters '(' and ')', find the length of the longest valid (well-formed) parentheses substring.
28. **Generate Parentheses**: Given n pairs of parentheses, write a function to generate all combinations of well-formed parentheses.
29. **Sudoku Solver**: Write a program to solve a Sudoku puzzle.
30. **Permutations**: Given a collection of distinct integers, return all possible permutations.
31. **Combination Sum**: Given a set of candidate numbers (candidates) and a target number (target), find all unique combinations in candidates where the candidate numbers sum to target.
32. **Next Permutation**: Implement next permutation, which rearranges numbers into the lexicographically next greater permutation of numbers.
33. **Search in Rotated Sorted Array II**: Search for a target value in an array of rotated sorted numbers.
34. **Jump Game**: Determine if you are able to reach the last index from the first index in a given array of non-negative integers.
35. **Multiply Strings**: Given two non-negative integers num1 and num2 represented as strings, return the product of num1 and num2, also represented as a string.
36. **Subsets**: Given a set of distinct integers, nums, return all possible subsets (the power set).

37. **Merge k Sorted Lists**: Merge k sorted linked lists and return it as one sorted list.
38. **Group Anagrams**: Given an array of strings, group anagrams together.
39. **Pow(x, n)**: Implement pow(x, n), which calculates x raised to the power n.
40. **Word Search**: Given a 2D board and a word, find if the word exists in the grid.
41. **Longest Substring Without Repeating Characters**: Given a string, find the length of the longest substring without repeating characters.
42. **Median of Two Sorted Arrays**: There are two sorted arrays nums1 and nums2 of size m and n respectively. Find the median of the two sorted arrays.
43. **Valid Palindrome II**: Given a non-empty string s, you may delete at most one character. Judge whether you can make it a palindrome.
44. **Serialize and Deserialize Binary Tree**: Design an algorithm to serialize and deserialize a binary tree.
45. **Find All Duplicates in an Array**: Given an array of integers, 1 ≤ a[i] ≤ n (n = size of array), some elements appear twice and others appear once. Find all the elements that appear twice in this array.
46. **Meeting Rooms**: Given an array of meeting time intervals consisting of start and end times [[s1,e1],[s2,e2],...], determine if a person could attend all meetings.
47. **Meeting Rooms II**: Given an array of meeting time intervals consisting of start and end times [[s1,e1],[s2,e2],...], find the minimum number of conference rooms required.
48. **Implement Trie (Prefix Tree)**: Implement a trie with insert, search, and startsWith methods.
49. **Minimum Window Substring**: Given a string S and a string T, find the minimum window in S which will contain all the characters in T in complexity O(n).
50. **Kth Largest Element in an Array**: Find the kth largest element in an unsorted array.
51. **Unique Paths**: A robot is located at the top-left corner of a m x n grid (marked 'Start' in the diagram below). The robot can only move either down or right at any point in time. The robot is trying to reach the bottom-right corner of the grid (marked 'Finish' in the diagram below). How many possible unique paths are there?
52. **Word Break**: Given a non-empty string s and a dictionary wordDict containing a list of non-empty words, determine if s can be segmented into a space-separated sequence of one or more dictionary words.

53. **Best Time to Buy and Sell Stock**: Say you have an array for which the ith element is the price of a given stock on day i. If you were only permitted to complete at most one transaction (i.e., buy one and sell one share of the stock), design an algorithm to find the maximum profit.

54. **Convert Sorted Array to Binary Search Tree**: Given an array where elements are sorted in ascending order, convert it to a height-balanced binary search tree.

55. **Maximal Square**: Given a 2D binary matrix filled with 0's and 1's, find the largest square containing only 1's and return its area.

56. **Longest Increasing Subsequence**: Given an unsorted array of integers, find the length of longest increasing subsequence.

57. **Combination Sum II**: Given a collection of candidate numbers (candidates) and a target number (target), find all unique combinations in candidates where the candidate numbers sum to target. Each number in candidates may only be used once in the combination.

58. **Binary Tree Maximum Path Sum**: Given a non-empty binary tree, find the maximum path sum.

59. **Minimum Depth of Binary Tree**: Given a binary tree, find its minimum depth.

60. **Best Time to Buy and Sell Stock II**: Say you have an array prices for which the ith element is the price of a given stock on day i. Design an algorithm to find the maximum profit. You may complete as many transactions as you like (i.e., buy one and sell one share of the stock multiple times).

61. **Symmetric Tree**: Given a binary tree, check whether it is a mirror of itself (i.e., symmetric around its center).

62. **Validate Binary Search Tree**: Given a binary tree, determine if it is a valid binary search tree (BST).

63. **Maximum Depth of Binary Tree**: Given a binary tree, find its maximum depth.

64. **Construct Binary Tree from Preorder and Inorder Traversal**: Given preorder and inorder traversal of a tree, construct the binary tree.

65. **Palindrome Partitioning**: Given a string s, partition s such that every substring of the partition is a palindrome.

66. **Longest Palindromic Substring**: Given a string s, find the longest palindromic substring in s.

67. **Combination Sum III**: Find all possible combinations of k numbers that add up to a number n, given that only numbers from 1 to 9 can be used, and each combination should be a unique set of numbers.

68. **Search a 2D Matrix**: Write an efficient algorithm that searches for a value in an m x n matrix. This matrix has the following properties: Integers in each row are sorted from left to right. The first integer of each row is greater than the last integer of the previous row.
69. **House Robber**: You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed. All houses at this place are arranged in a circle. That means the first house is the neighbor of the last one. Meanwhile, adjacent houses have security system connected, and it will automatically contact the police if two adjacent houses were broken into on the same night. Given a list of non-negative integers representing the amount of money of each house, determine the maximum amount of money you can rob tonight without alerting the police.
70. **Word Ladder**: Given two words (beginWord and endWord), and a dictionary's word list, find the length of shortest transformation sequence from beginWord to endWord, such that only one letter can be changed at a time and each transformed word must exist in the word list.
71. **Unique Binary Search Trees**: Given n, how many structurally unique BST's (binary search trees) that store values 1 ... n?
72. **Distinct Subsequences**: Given two strings s and t, return the number of distinct subsequences of s which equals t.
73. **Jump Game II**: Given an array of non-negative integers, you are initially positioned at the first index of the array. Each element in the array represents your maximum jump length from that position. Your goal is to reach the last index in the minimum number of jumps.
74. **First Unique Character in a String**: Given a string, find the first non-repeating character in it and return its index. If it doesn't exist, return -1.
75. **Best Time to Buy and Sell Stock III**: You are given an array prices where prices[i] is the price of a given stock on the ith day. Find the maximum profit you can achieve. You may complete at most two transactions.
76. **Climbing Stairs**: You are climbing a stair case. It takes n steps to reach to the top. Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?
77. **Word Search II**: Given a 2D board and a list of words from the dictionary, find all words in the board.
78. **Sort Colors**: Given an array with n objects colored red, white, or blue, sort them in-place so that objects of the same color are adjacent, with the colors in the order red, white, and blue.
79. **Combination Sum IV**: Given an integer array with all positive numbers and no duplicates, find the number of possible combinations that add up to a positive integer target.

80. **Find Minimum in Rotated Sorted Array**: Suppose an array sorted in ascending order is rotated at some pivot unknown to you beforehand. Find the minimum element.

81. **Find Peak Element**: A peak element is an element that is strictly greater than its neighbors. Given an integer array nums, find a peak element, and return its index.

82. **3Sum**: Given an array nums of n integers, are there elements a, b, c in nums such that a + b + c = 0? Find all unique triplets in the array which gives the sum of zero.

83. **Subarray Sum Equals K**: Given an array of integers nums and an integer k, return the total number of continuous subarrays whose sum equals to k.

84. **Minimum Size Subarray Sum**: Given an array of positive integers nums and a positive integer target, return the minimal length of a contiguous subarray [numsl, numsl+1, ..., numsr-1, numsr] of which the sum is greater than or equal to target.

85. **Longest Consecutive Sequence**: Given an unsorted array of integers nums, return the length of the longest consecutive elements sequence.

86. **Valid Number**: Validate if a given string can be interpreted as a decimal number.

87. **Combination Sum IV**: Given an integer array with all positive numbers and no duplicates, find the number of possible combinations that add up to a positive integer target.

88. **Best Time to Buy and Sell Stock IV**: You are given an integer array prices where prices[i] is the price of a given stock on the ith day, and an integer k. Find the maximum profit you can achieve. You may complete at most k transactions.

89. **Add and Search Word - Data structure design**: Design a data structure that supports adding new words and finding if a string matches any previously added string.

90. **Meeting Scheduler**: Given the availability time slots arrays slots1 and slots2 of two people and a meeting duration duration, return the earliest time slot that works for both of them and is of duration duration.

91. **Longest Palindromic Subsequence**: Given a string s, find the longest palindromic subsequence's length in s.

92. **Shortest Path in Binary Matrix**: Given an n x n binary matrix grid, return the length of the shortest clear path in the matrix. If there is no clear path, return -1.

93. **Word Break II**: Given a non-empty string s and a dictionary wordDict containing a list of non-empty words, add spaces in s to construct a sentence where each word is a valid dictionary word. Return all such possible sentences.

94. **Minimum Window Substring**: Given two strings s and t, return the minimum window in s which will contain all the characters in t. If there is no such window in s that covers all characters in t, return an empty string "".

95. **Longest Increasing Path in a Matrix**: Given an m x n integers matrix, return the length of the longest increasing path in matrix.

96. **Jump Game III**: Given an array of non-negative integers arr, you are initially positioned at start index of the array. When you are at index i, you can jump to i + arr[i] or i - arr[i], check if you can reach to any index with value 0.

97. **Maximum Product Subarray**: Given an integer array nums, find the contiguous subarray within an array (containing at least one number) which has the largest product.

98. **Binary Tree Right Side View**: Given a binary tree, imagine yourself standing on the right side of it, return the values of the nodes you can see ordered from top to bottom.

99. **Non-overlapping Intervals**: Given a collection of intervals, find the minimum number of intervals you need to remove to make the rest of the intervals non-overlapping.

100. **Kth Smallest Element in a Sorted Matrix**: Given an n x n matrix where each of the rows and columns are sorted in ascending order, return the kth smallest element in the matrix.

101. **Minimum Path Sum**: Given a grid filled with non-negative numbers, find a path from the top-left corner to the bottom-right corner, which minimizes the sum of all numbers along its path.

102. **Max Points on a Line**: Given an array of points where points[i] = [xi, yi] represents a point on the X-Y plane, return the maximum number of points that lie on the same straight line.

103. **Find All Anagrams in a String**: Given a string s and a non-empty string p, find all the start indices of p's anagrams in s.

104. **Rotate Image**: You are given an n x n 2D matrix representing an image, rotate the image by 90 degrees (clockwise).

105. **Two City Scheduling**: There are 2n people, labeled from 0 to 2n - 1 where each person has a cost[i] and a city[i] representing the cost of living in that city for the person. Find the minimum cost to fly every person to a city such that exactly n people arrive in each city.

106. **Find Median from Data Stream**: Design a data structure that supports the following two operations: void addNum(int num) - Add a integer number from the data stream to the data structure. double findMedian() - Return the median of all elements so far.

107. **Counting Bits**: Given a non-negative integer num, for every number i in the range 0 ≤ i ≤ num, calculate the number of 1's in their binary representation and return them as an array.

108. **Implement Stack using Queues**: Implement the following operations of a stack using queues: push(x), pop(), top(), and empty().

109. **Insert Delete GetRandom O(1)**: Design a data structure that supports all following operations in average O(1) time: insert(val), remove(val), getRandom().

110. **Reorder Data in Log Files**: You have an array of logs. Each log is a space-delimited string of words, where the first word is the identifier and the rest of the words are words in the log message. Reorder the logs so that all the letter-logs come before any digit-log. The letter-logs are ordered lexicographically ignoring identifier, with the identifier used in case of ties. The digit-logs should be in their original order.

111. **Largest Rectangle in Histogram**: Given n non-negative integers representing the histogram's bar height where the width of each bar is 1, find the area of the largest rectangle in the histogram.

112. **Evaluate Reverse Polish Notation**: Evaluate the value of an arithmetic expression in Reverse Polish Notation.

113. **Longest Substring with At Least K Repeating Characters**: Given a string s and an integer k, return the length of the longest substring of s such that the frequency of each character in this substring is greater than or equal to k.

114. **Partition Equal Subset Sum**: Given a non-empty array nums containing only positive integers, determine if it can be partitioned into two subsets such that the sum of elements in both subsets is equal.

115. **Longest Valid Parentheses**: Given a string containing just the characters '(' and ')', find the length of the longest valid (well-formed) parentheses substring.

116. **Merge Sorted Array**: Given two sorted integer arrays nums1 and nums2, merge nums2 into nums1 as one sorted array.

117. **Flatten Nested List Iterator**: Implement an iterator to flatten a nested list of integers. Each element is either an integer or a list whose elements may also be integers or other lists.

118. **Reconstruct Itinerary**: Given a list of airline tickets represented by pairs of departure and arrival airports [from, to], reconstruct the itinerary in order. All of the tickets belong to a man who departs from "JFK", thus, the itinerary must begin with "JFK".

119. **Number of Islands**: Given an m x n 2D binary grid grid which represents a map of '1's (land) and '0's (water), return the number of islands.

120. **Design Hit Counter**: Design a hit counter which counts the number of hits received in the past 5 minutes (i.e., the past 300 seconds).