BOSSCODER ACADEMY

100 MOST-LIKED LeetCode Questions

Disclaimer

No list alone can guarantee mastery.

What matters is how you break down problems, learn patterns, and adapt to challenges.

This doc gives you the practice to do exactly that.

BACKTRACKING

1. Letter Combinations of a Phone Number (Medium)

Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent. Return the answer in any order.

A mapping of digits to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters

Practice

2. Generate Parentheses (Medium)

Given n pairs of parentheses, write a function to generate all combinations of well-formed parentheses.

Example 1:

Input: n = 3

Output: ["((()))","(()())","(())()","()(())","()()()"]

Example 2:

Input: n = 1

Output: ["()"]

Constraints:

1 <= n <= 8

Practice

3. Combination Sum (Medium)

Given an array of distinct integers candidates and a target integer target, return a list of all unique combinations of candidates where the chosen numbers sum to target. You may return the combinations in any order.

The same number may be chosen from candidates an unlimited number of times. Two combinations are unique if the frequency of at least one of the chosen numbers is different.

The test cases are generated such that the number of unique combinations that sum up to the target is less than 150 combinations for the given input.

Practice

4. Permutations (Medium)

Given an array nums of distinct integers, return all the possible permutations. You can return the answer in any order.

Practice

5. N-Queens (Hard)

The n-queens puzzle is the problem of placing n queens on an n x n chessboard such that no two queens attack each other.

Given an integer n, return all distinct solutions to the n-queens puzzle. You may return the answer in any order.

Each solution contains a distinct board configuration of the n-queens' placement, where 'Q' and '.' both indicate a queen and an empty space, respectively.

Practice

6. Subsets (Medium)

Given an integer array nums of unique elements, return all possible subsets (the power set).

The solution set must not contain duplicate subsets. Return the solution in any order.

Practice

7. Word Search (Medium)

Given an m x n grid of characters board and a string word, return true if word exists in the grid.

The word can be constructed from letters of sequentially adjacent cells, where adjacent cells are horizontally or vertically neighboring. The same letter cell may not be used more than once.

Practice

8. Palindrome Partitioning (Medium)

Given a string s, partition s such that every substring of the partition is a palindrome. Return all possible palindrome partitioning of s.

Practice

BINARY SEARCH

9. Median of Two Sorted Arrays (Hard)

Given two sorted arrays nums1 and nums2 of size m and n respectively, return the median of the two sorted arrays.

The overall run time complexity should be O(log (m+n)).

Practice

10. Search in Rotated Sorted Array (Medium)

There is an integer array nums sorted in ascending order (with distinct values).

Prior to being passed to your function, nums is possibly left rotated at an unknown index k (1 <= k < nums.length) such that the resulting array is [nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]] (0-indexed). For example, [0,1,2,4,5,6,7] might be left rotated by 3 indices and become [4,5,6,7,0,1,2].

Given the array nums after the possible rotation and an integer target, return the index of target if it is in nums, or -1 if it is not in nums.

You must write an algorithm with O(log n) runtime complexity.

Practice

11. Find First and Last Position of Element in Sorted Array (Medium)

Given an array of integers nums sorted in non-decreasing order, find the starting and ending position of a given target value.

If target is not found in the array, return [-1, -1].

You must write an algorithm with O(log n) runtime complexity.

Practice

12. Search Insert Position (Easy)

Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.

You must write an algorithm with O(log n) runtime complexity.

Practice

13. Search a 2D Matrix (Medium)

You are given an m x n integer matrix matrix with the following two properties:

Each row is sorted in non-decreasing order.

The first integer of each row is greater than the last integer of the previous row.

Given an integer target, return true if target is in matrix or false otherwise.

You must write a solution in O(log(m * n)) time complexity.

Practice

14. Binary Tree Maximum Path Sum (Hard)

A path in a binary tree is a sequence of nodes where each pair of adjacent nodes in the sequence has an edge connecting them. A node can only appear in the sequence at most once. Note that the path does not need to pass through the root.

The path sum of a path is the sum of the node's values in the path.

Given the root of a binary tree, return the maximum path sum of any non-empty path.

Practice

15. Find Minimum in Rotated Sorted Array (Medium)

Suppose an array of length n sorted in ascending order is rotated between 1 and n times. For example, the array nums = [0,1,2,4,5,6,7] might become:

[4,5,6,7,0,1,2] if it was rotated 4 times.

[0,1,2,4,5,6,7] if it was rotated 7 times.

Notice that rotating an array [a[0], a[1], a[2], ..., a[n-1]] 1 time results in the array [a[n-1], a[0], a[1], a[2], ..., a[n-2]].

Given the sorted rotated array nums of unique elements, return the minimum element of this array.

You must write an algorithm that runs in O(log n) time.

nd the output represents the signed integer -1073741825.

Practice

BINARY TREE

16. Binary Tree Inorder Traversal (Easy)

Given the root of a binary tree, return the inorder traversal of its nodes' values.

Practice

17. Validate Binary Search Tree (Medium)

Given the root of a binary tree, determine if it is a valid binary search tree (BST).

A valid BST is defined as follows:

The left subtree of a node contains only nodes with keys strictly less than the node's key.

The right subtree of a node contains only nodes with keys strictly greater than the node's key.

Both the left and right subtrees must also be binary search trees.

Practice

18.Symmetric Tree (Easy)

Given the root of a binary tree, check whether it is a mirror of itself (i.e., symmetric around its center).

Practice

19. Binary Tree Level Order Traversal (Medium)

Given the root of a binary tree, return the level order traversal of its nodes' values. (i.e., from left to right, level by level).

Practice

20. Maximum Depth of Binary Tree (Easy)

Given the root of a binary tree, return its maximum depth.

A binary tree's maximum depth is the number of nodes along the longest path from the root node down to the farthest leaf node.

Practice

21. Construct Binary Tree from Preorder and Inorder Traversal (Medium)

Given two integer arrays preorder and inorder where preorder is the preorder traversal of a binary tree and inorder is the inorder traversal of the same tree, construct and return the binary tree.

Practice

22. Convert Sorted Array to Binary Search Tree (Easy)

Given an integer array nums where the elements are sorted in ascending order, convert it to a height-balanced binary search tree.

(A height-balanced binary tree is a binary tree in which the depth of the two subtrees of every node never differs by more than one.)

Practice

23. Flatten Binary Tree to Linked List (Medium)

Given the root of a binary tree, flatten the tree into a "linked list":

The "linked list" should use the same TreeNode class where the right child pointer points to the next node in the list and the left child pointer is always null.

The "linked list" should be in the same order as a pre-order traversal of the binary tree.

Practice

24. Binary Tree Right Side View (Medium)

Given the root of 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.

Practice

25.Invert Binary Tree (Easy)

Given the root of a binary tree, invert the tree, and return its root.

Practice

26. Kth Smallest Element in a BST (Medium)

Given the root of a binary search tree, and an integer k, return the kth smallest value (1-indexed) of all the values of the nodes in the tree.

Practice

27. Lowest Common Ancestor of a Binary Tree (Medium)

Given a binary tree, find the lowest common ancestor (LCA) of two given nodes in the tree.

According to the definition of LCA on Wikipedia: “The lowest common ancestor is defined between two nodes p and q as the lowest node in T that has both p and q as descendants (where we allow a node to be a descendant of itself).”

Practice

28. Path Sum III (Medium)

Given the root of a binary tree and an integer targetSum, return the number of paths where the sum of the values along the path equals targetSum.

The path does not need to start or end at the root or a leaf, but it must go downwards (i.e., traveling only from parent nodes to child nodes).

Practice

29. Diameter of Binary Tree (Easy)

Given the root of a binary tree, return the length of the diameter of the tree.

The diameter of a binary tree is the length of the longest path between any two nodes in a tree. This path may or may not pass through the root.

The length of a path between two nodes is represented by the number of edges between them.

Practice

DYNAMIC PROGRAMMING

30. Longest Palindromic Substring (Medium)

Given a string s, return the longest palindromic substring in s.

(A substring is a contiguous non-empty sequence of characters within a string.)

Practice

31. Longest Valid Parentheses (Hard)

Given a string containing just the characters '(' and ')', return the length of the longest valid (well-formed) parentheses substring.

Practice

32. Unique Paths (Medium)

There is a robot on an m x n grid. The robot is initially located at the top-left corner (i.e., grid[0][0]). The robot tries to move to the bottom-right corner (i.e., grid[m - 1][n - 1]). The robot can only move either down or right at any point in time.

Given the two integers m and n, return the number of possible unique paths that the robot can take to reach the bottom-right corner.

The test cases are generated so that the answer will be less than or equal to 2 * 109.

Practice

---

33. Minimum Path Sum (Medium)

Given a m x n grid filled with non-negative numbers, find a path from top left to bottom right, which minimizes the sum of all numbers along its path.

Note: You can only move either down or right at any point in time.

Practice

34. Climbing Stairs (Easy)

You are climbing a staircase. It takes n steps to reach the top.

Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

Practice

35. Edit Distance (Medium)

Given two strings word1 and word2, return the minimum number of operations required to convert word1 to word2.

You have the following three operations permitted on a word:

- Insert a character
- Delete a character
- Replace a character

Practice

36. Pascal's Triangle (Easy)

Given an integer numRows, return the first numRows of Pascal's triangle.

In Pascal's triangle, each number is the sum of the two numbers directly above it.

Practice

37.Word Break (Medium)

Given a string s and a dictionary of strings wordDict, return true if s can be segmented into a space-separated sequence of one or more dictionary words.

Note that the same word in the dictionary may be reused multiple times in the segmentation.

Practice

38.Maximum Product Subarray (Medium)

Given an integer array nums, find a subarray that has the largest product, and return the product.

The test cases are generated so that the answer will fit in a 32-bit integer.

(A subarray is a contiguous non-empty sequence of elements within an array.)

Practice

39.House Robber (Medium)

You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security systems connected and it will automatically contact the police if two adjacent houses were broken into on the same night.

Given an integer array nums representing the amount of money of each house, return the maximum amount of money you can rob tonight without alerting the police.

Practice

LINKED LIST

40. Reverse a Linked List

Given the head of a singly linked list, reverse the list, and return the reversed list.

Practice

41. Detect Cycle in a Linked List

Given head, the head of a linked list, determine if the linked list has a cycle in it.

There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to. Note that pos is not passed as a parameter.

Return true if there is a cycle in the linked list. Otherwise, return false.

Practice

42. Coin Change (Medium)

You are given an integer array coins representing coins of different denominations and an integer amount representing a total amount of money.

Return the fewest number of coins that you need to make up that amount. If that amount of money cannot be made up by any combination of the coins, return -1.

You may assume that you have an infinite number of each kind of coin.

Practice

43. Partition Equal Subset Sum (Medium)

Given an integer array nums, return true if you can partition the array into two subsets such that the sum of the elements in both subsets is equal or false otherwise.

Practice

44. Longest Common Subsequence (Medium)

Given two strings text1 and text2, return the length of their longest common subsequence. If there is no common subsequence, return 0.

A subsequence of a string is a new string generated from the original string with some characters (can be none) deleted without changing the relative order of the remaining characters.

For example, "ace" is a subsequence of "abcde".

A common subsequence of two strings is a subsequence that is common to both strings.

Practice

GRAPH

45. Number of Islands (Medium)

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.

An island is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water..

Practice

46. Course Schedule (Medium)

There are a total of numCourses courses you have to take, labeled from 0 to numCourses - 1. You are given an array prerequisites where prerequisites[i] = [ai, bi] indicates that you must take course bi first if you want to take course ai.

For example, the pair [0, 1], indicates that to take course 0 you have to first take course 1.

Return true if you can finish all courses. Otherwise, return false.

Practice

47. Rotting Oranges (Medium)

You are given an m x n grid where each cell can have one of three values:

- 0 representing an empty cell,
- 1 representing a fresh orange, or
- 2 representing a rotten orange.
- Every minute, any fresh orange that is 4-directionally adjacent to a rotten orange becomes rotten.

Return the minimum number of minutes that must elapse until no cell has a fresh orange. If this is impossible, return -1.

Practice

GREEDY

48. Jump Game II (Medium)

You are given a 0-indexed array of integers nums of length n. You are initially positioned at index 0.

Each element nums[i] represents the maximum length of a forward jump from index i. In other words, if you are at index i, you can jump to any index (i + j) where:

- 0 <= j <= nums[i] and
- i + j < n

Return the minimum number of jumps to reach index n - 1. The test cases are generated such that you can reach index n - 1.

Practice

49. Jump Game (Medium)

You are given an integer array nums. You are initially positioned at the array's first index, and each element in the array represents your maximum jump length at that position.

Return true if you can reach the last index, or false otherwise.

Practice

50. Best Time to Buy and Sell Stock (Easy)

You are given an array prices where prices[i] is the price of a given stock on the ith day.

You want to maximize your profit by choosing a single day to buy one stock and choosing a different day in the future to sell that stock.

Return the maximum profit you can achieve from this transaction. If you cannot achieve any profit, return 0.

Practice

51. Partition Labels (Medium)

You are given a string s. We want to partition the string into as many parts as possible so that each letter appears in at most one part. For example, the string "ababcc" can be partitioned into ["abab", "cc"], but partitions such as ["aba", "bcc"] or ["ab", "ab", "cc"] are invalid.

Note that the partition is done so that after concatenating all the parts in order, the resultant string should be s.

Return a list of integers representing the size of these parts.

Practice

HASHING

52. Two Sum (Easy)

Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.

You may assume that each input would have exactly one solution, and you may not use the same element twice.

You can return the answer in any order.

Practice

53. Group Anagrams (Medium)

Given an array of strings strs, group the anagrams together. You can return the answer in any order.

(An anagram is a word or phrase formed by rearranging the letters of a different word or phrase, using all the original letters exactly once.)

Practice

54. Longest Consecutive Sequence (Medium)

Given an unsorted array of integers nums, return the length of the longest consecutive elements sequence.

You must write an algorithm that runs in O(n) time.

Practice

55. Subarray Sum Equals K (Medium)

Given an array of integers nums and an integer k, return the total number of subarrays whose sum equals to k.

A subarray is a contiguous non-empty sequence of elements within an array.

Practice

HEAP

56. Kth Largest Element in an Array (Medium)

Given an integer array nums and an integer k, return the kth largest element in the array.

Note that it is the kth largest element in the sorted order, not the kth distinct element.

Can you solve it without sorting?

Practice

57. Find Median from Data Stream (Hard)

The median is the middle value in an ordered integer list. If the size of the list is even, there is no middle value, and the median is the mean of the two middle values.

- For example, for arr = [2,3,4], the median is 3.
- For example, for arr = [2,3], the median is (2 + 3) / 2 = 2.5.

Implement the MedianFinder class:

- MedianFinder() initializes the MedianFinder object.
- void addNum(int num) adds the integer num from the data stream to the data structure.
- double findMedian() returns the median of all elements so far. Answers within 10-5 of the actual answer will be accepted.

Practice

58. Top K Frequent Elements (Medium)

Given an integer array nums and an integer k, return the k most frequent elements. You may return the answer in any order.

Practice

LINKED LISTS

59. Add Two Numbers (Medium)

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 contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

Practice

60. Remove Nth Node From End of List (Medium)

Given the head of a linked list, remove the nth node from the end of the list and return its head.

Practice

61. Merge Two Sorted Lists (Easy)

You are given the heads of two sorted linked lists list1 and list2.

Merge the two lists into one sorted list. The list should be made by splicing together the nodes of the first two lists.

Return the head of the merged linked list.

Practice

62.Merge k Sorted Lists (Hard)

You are given an array of k linked-lists lists, each linked-list is sorted in ascending order.

Merge all the linked-lists into one sorted linked-list and return it.

Practice

63. Swap Nodes in Pairs (Medium)

Given a linked list, swap every two adjacent nodes and return its head. You must solve the problem without modifying the values in the list's nodes (i.e., only nodes themselves may be changed.)

Practice

64. Reverse Nodes in k-Group (Hard)

Given the head of a linked list, reverse the nodes of the list k at a time, and return the modified list.

k is a positive integer and is less than or equal to the length of the linked list. If the number of nodes is not a multiple of k then left-out nodes, in the end, should remain as it is.

You may not alter the values in the list's nodes, only nodes themselves may be changed.

Practice

65.Copy List with Random Pointer (Medium)

A linked list of length n is given such that each node contains an additional random pointer, which could point to any node in the list, or null.

Construct a deep copy of the list. The deep copy should consist of exactly n brand new nodes, where each new node has its value set to the value of its corresponding original node. Both the next and random pointer of the new nodes should point to new nodes in the copied list such that the pointers in the original list and copied list represent the same list state. None of the pointers in the new list should point to nodes in the original list.

For example, if there are two nodes X and Y in the original list, where X.random → Y, then for the corresponding two nodes x and y in the copied list, x.random → y.

Return the head of the copied linked list.

The linked list is represented in the input/output as a list of n nodes. Each node is represented as a pair of [val, random_index] where:val: an integer representing Node.valrandom_index: the index of the node (range from 0 to n-1) that the random pointer points to, or null if it does not point to any node.

Your code will only be given the head of the original linked list.

Practice

66. Linked List Cycle (Easy)

Given head, the head of a linked list, determine if the linked list has a cycle in it.

There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to. Note that pos is not passed as a parameter.

Return true if there is a cycle in the linked list. Otherwise, return false.

Practice

67. Linked List Cycle II (Medium)

Given the head of a linked list, return the node where the cycle begins. If there is no cycle, return null.

There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to (0-indexed). It is -1 if there is no cycle. Note that pos is not passed as a parameter.

Do not modify the linked list.

Practice

68. LRU Cache (Medium)

Design a data structure that follows the constraints of a Least Recently Used (LRU) cache.

Implement the LRUCache class:

- LRUCache(int capacity) Initialize the LRU cache with positive size capacity.
- int get(int key) Return the value of the key if the key exists, otherwise return -1.
- void put(int key, int value) Update the value of the key if the key exists. Otherwise, add the key-value pair to the cache. If the number of keys exceeds the capacity from this operation, evict the least recently used key.

The functions get and put must each run in O(1) average time complexity.

Practice

69. Kth Smallest Element in a BST

Given the head of a linked list, return the list after sorting it in ascending order.

Practice

70. Intersection of Two Linked Lists (Easy)

Given the heads of two singly linked-lists headA and headB, return the node at which the two lists intersect. If the two linked lists have no intersection at all, return null.

Note that the linked lists must retain their original structure after the function returns.

Custom Judge:

The inputs to the judge are given as follows (your program is not given these inputs):

- intersectVal - The value of the node where the intersection occurs. This is 0 if there is no intersected node.
- listA - The first linked list.
- listB - The second linked list.
- skipA - The number of nodes to skip ahead in listA (starting from the head) to get to the intersected node.
- skipB - The number of nodes to skip ahead in listB (starting from the head) to get to the intersected node.

The judge will then create the linked structure based on these inputs and pass the two heads, headA and headB to your program. If you correctly return the intersected node, then your solution will be accepted.

Follow up: Could you write a solution that runs in O(m + n) time and use only O(1) memory?

Practice

71. Reverse Linked List (Easy)

Given the head of a singly linked list, reverse the list, and return the reversed list.

Practice

72. Palindrome Linked List (Easy)

Given the head of a singly linked list, return true if it is a palindrome or false otherwise.

(A palindrome is a sequence that reads the same forward and backward.)

Practice

MATRIX

73. Rotate Image (Medium)

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

You have to rotate the image in-place, which means you have to modify the input 2D matrix directly. DO NOT allocate another 2D matrix and do the rotation..

Practice

---

74. Spiral Matrix (Medium)

Given an m x n matrix, return all elements of the matrix in spiral order.

Practice

75. Set Matrix Zeroes (Medium)

Given an m x n integer matrix matrix, if an element is 0, set its entire row and column to 0's. You must do it in place.

Follow up:

- A straightforward solution using O(mn) space is probably a bad idea.
- A simple improvement uses O(m + n) space, but still not the best solution.
- Could you devise a constant space solution?

Practice

---

76. Search a 2D Matrix II (Medium)

Write an efficient algorithm that searches for a value target in an m x n integer matrix matrix. This matrix has the following properties:

Integers in each row are sorted in ascending from left to right.

Integers in each column are sorted in ascending from top to bottom.

Practice

SLIDING WINDOW

77.Longest Substring Without Repeating Characters (Medium)

Given a string s, find the length of the longest substring without duplicate characters.

(A substring is a contiguous non-empty sequence of characters within a string.)

Practice

78. Minimum Window Substring (Hard)

Given two strings s and t of lengths m and n respectively, return the minimum window substring of s such that every character in t (including duplicates) is included in the window. If there is no such substring, return the empty string "".

The test cases will be generated such that the answer is unique.

Practice

79. Sliding Window Maximum (Hard)

You are given an array of integers nums, there is a sliding window of size k which is moving from the very left of the array to the very right. You can only see the k numbers in the window. Each time the sliding window moves right by one position.

Return the max sliding window.

Practice

80. Find All Anagrams in a String (Medium)

Given two strings s and p, return an array of all the start indices of p's anagrams in s. You may return the answer in any order.

(An anagram is a word or phrase formed by rearranging the letters of a different word or phrase, using all the original letters exactly once.)

Practice

STACK

81. Valid Parentheses (Easy)

Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.
3. Every close bracket has a corresponding open bracket of the same type.

Practice

82. Largest Rectangle in Histogram (Hard)

Given an array of integers heights representing the histogram's bar height where the width of each bar is 1, return the area of the largest rectangle in the histogram.

Practice

83. Min Stack (Medium)

Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

Implement the MinStack class:

- MinStack() initializes the stack object.
- void push(int val) pushes the element val onto the stack.
- void pop() removes the element on the top of the stack.
- int top() gets the top element of the stack.
- int getMin() retrieves the minimum element in the stack.

You must implement a solution with O(1) time complexity for each function.

Practice

84. Decode String (Medium)

Given an encoded string, return its decoded string.

The encoding rule is: k[encoded_string], where the encoded_string inside the square brackets is being repeated exactly k times. Note that k is guaranteed to be a positive integer.

You may assume that the input string is always valid; there are no extra white spaces, square brackets are well-formed, etc. Furthermore, you may assume that the original data does not contain any digits and that digits are only for those repeat numbers, k. For example, there will not be input like 3a or 2[4].

The test cases are generated so that the length of the output will never exceed 105.

Practice

85. Daily Temperatures (Medium)

Given an array of integers temperatures represents the daily temperatures, return an array answer such that answer[i] is the number of days you have to wait after the ith day to get a warmer temperature. If there is no future day for which this is possible, keep answer[i] == 0 instead.

Practice

TWO POINTERS

86. Container With Most Water (Medium)

You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the ith line are (i, 0) and (i, height[i]).

Find two lines that together with the x-axis form a container, such that the container contains the most water.

Return the maximum amount of water a container can store.

Notice that you may not slant the container.

Practice

87. 3Sum (Medium)

Given an integer array nums, return all the triplets [nums[i], nums[j], nums[k]] such that i != j, i != k, and j != k, and nums[i] + nums[j] + nums[k] == 0.

Notice that the solution set must not contain duplicate triplets.

Practice

88. Trapping Rain Water (Hard)

Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.

Practice

89. Move Zeroes (Easy)

Given an integer array nums, move all 0's to the end of it while maintaining the relative order of the non-zero elements.

Note that you must do this in-place without making a copy of the array.

Practice

TRIE

90. Implement Trie [Prefix Tree] (Medium)

A trie (pronounced as "try") or prefix tree is a tree data structure used to efficiently store and retrieve keys in a dataset of strings. There are various applications of this data structure, such as autocomplete and spellchecker.

Implement the Trie class:

- Trie() Initializes the trie object.
- void insert(String word) Inserts the string word into the trie.
- boolean search(String word) Returns true if the string word is in the trie (i.e., was inserted before), and false otherwise.
- boolean startsWith(String prefix) Returns true if there is a previously inserted string word that has the prefix prefix, and false otherwise.

Practice

MISC

91. Next Permutation (Medium)

A permutation of an array of integers is an arrangement of its members into a sequence or linear order.

- For example, for arr = [1,2,3], the following are all the permutations of arr: [1,2,3], [1,3,2], [2, 1, 3], [2, 3, 1], [3,1,2], [3,2,1].

The next permutation of an array of integers is the next lexicographically greater permutation of its integer. More formally, if all the permutations of the array are sorted in one container according to their lexicographical order, then the next permutation of that array is the permutation that follows it in the sorted container. If such arrangement is not possible, the array must be rearranged as the lowest possible order (i.e., sorted in ascending order).

- For example, the next permutation of arr = [1,2,3] is [1,3,2].
- Similarly, the next permutation of arr = [2,3,1] is [3,1,2].
- While the next permutation of arr = [3,2,1] is [1,2,3] because [3,2,1] does not have a lexicographical larger rearrangement.

Given an array of integers nums, find the next permutation of nums.The replacement must be in place and use only constant extra memory

Practice

92. First Missing Positive (Hard)

Given an unsorted integer array nums. Return the smallest positive integer that is not present in nums.

You must implement an algorithm that runs in O(n) time and uses O(1) auxiliary space.

Practice

93. Maximum Subarray (Medium)

Given an integer array nums, find the subarray with the largest sum, and return its sum.

Practice

94. Merge Intervals (Medium)

Given an array of intervals where intervals[i] = [starti, endi], merge all overlapping intervals, and return an array of the non-overlapping intervals that cover all the intervals in the input.

Practice

95. Sort Colors (Medium)

Given an array nums 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.

We will use the integers 0, 1, and 2 to represent the color red, white, and blue, respectively.

You must solve this problem without using the library's sort function.

Practice

96. Single Number (Easy)

Given a non-empty array of integers nums, every element appears twice except for one. Find that single one.

You must implement a solution with a linear runtime complexity and use only constant extra space

Practice

97. Majority Element (Easy)

Given an array nums of size n, return the majority element.

The majority element is the element that appears more than ⌊n / 2⌋ times. You may assume that the majority element always exists in the array.

Practice

98. Rotate Array (Medium)

Given an integer array nums, rotate the array to the right by k steps, where k is non-negative.

Practice

99.Product of Array Except Self (Medium)

Given an integer array nums, return an array answer such that answer[i] is equal to the product of all the elements of nums except nums[i].

The product of any prefix or suffix of nums is guaranteed to fit in a 32-bit integer.

You must write an algorithm that runs in O(n) time and without using the division operation.

Practice

100. Find the Duplicate Number (Medium)

Given an array of integers nums containing n + 1 integers where each integer is in the range [1, n] inclusive.

There is only one repeated number in nums, return this repeated number.

You must solve the problem without modifying the array nums and using only constant extra space.

Practice

BOSSCODER ACADEMY

WHY BOSSCODER?

2200+ Alumni placed at Top Product-based companies.

More than **120% hike** for every 2 out of 3 Working Professional.

Average Package of **24LPA**.

The syllabus is most **up-to-date** and the list of problems provided covers **all important topics.**

Lavanya
Meta

Course is very well **structured** and **streamlined** to crack any **MAANG** company .

Rahul
Google

EXPLORE MORE