AlgoTutor

# 75 MOST IMPORTANT LEETCODE DSA QUESTIONS

+91-7260058093

info@algotutor.io

www.algotutor.io

AlgoTutor

# Array

## 1. Two Sum

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

**Input**: nums = [2,7,11,15], target = 9
**Output**: [0,1]
**Explanation**: Because nums[0] + nums[1] == 9, we return [0, 1].

**PRACTICE NOW**

## 2. Best Time to Buy and Sell Stock

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

**Input**: prices = [7,1,5,3,6,4]
**Output**: 5
**Explanation**: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5

# 3. Contains Duplicate

**Given an integer array nums, return true if any value appears at least twice in the array, and return false if every element is distinct.**

**Input**: nums = [1,2,3,1]
**Output**: true

PRACTICE NOW

# 4. Product of Array Except Self

**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.**

**Input**: nums = [1,2,3,4]
**Output**: [24,12,8,6]

PRACTICE NOW

# 5. Maximum Subarray

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

**Input**: nums = [-2,1,-3,4,-1,2,1,-5,4]
**Output**: 6
**Explanation**: The subarray [4,-1,2,1] has the largest sum 6.

**Input**: nums = [1]
**Output**: 1
**Explanation**: The subarray [1] has the largest sum of 1.

**PRACTICE NOW**

# 6. Maximum Product Subarray

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

**Input**: nums = [2,3,-2,4]
**Output**: 6
**Explanation**: [2,3] has the largest product 6.

**Input**: nums = [-2,0,-1]
**Output**: 0
**Explanation**: The result cannot be 2, because [-2,-1] is not a subarray.

**PRACTICE NOW**

# 7. Find the Minimum in Rotated Sorted Array

**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.**

**Input:** nums = [3,4,5,1,2]
**Output:** 1
**Explanation:** The original array was [1,2,3,4,5] rotated 3 times.

**PRACTICE NOW**

# 8. Search in Rotated Sorted Array

**Given the array nums after the possible rotation and an integer target, return the index of the 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.**

**Input:** nums = [4,5,6,7,0,1,2], target = 0
**Output:** 4

**Input:** nums = [4,5,6,7,0,1,2], target = 3
**Output:** -1

**PRACTICE NOW**

# 9. 3Sum

**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.**

**Input**: nums = [-1,0,1,2,-1,-4]
**Output**: [[-1,-1,2],[-1,0,1]]
**Explanation**:
nums[0] + nums[1] + nums[2] = (-1) + 0 + 1 = 0.
nums[1] + nums[2] + nums[4] = 0 + 1 + (-1) = 0.
nums[0] + nums[3] + nums[4] = (-1) + 2 + (-1) = 0.
The distinct triplets are [-1,0,1] and [-1,-1,2].

**PRACTICE NOW**

# 10. Container With Most Water

**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.**

**Input**: height = [1,8,6,2,5,4,8,3,7]
**Output**: 49
**Explanation**: The above vertical lines are represented by an array [1,8,6,2,5,4,8,3,7]. In this case, the max area of water (blue section) the container can contain is 49.

AlgoTutor

# Why Choose AlgoTutor?

- 100 % PLACEMENT ASSISTANCE
- 1:1 PERSONAL MENTORSHIP FROM INDUSTRY EXPERTS
- 100 % SUCCESS RATE
- 23 LPA(AVG.)CTC
- 200+ SUCCESSFUL ALUMNI
- 147%(AVG.) SALARY HIKE
- LEARN FROM SCRATCH
- CAREER SERVICES

EXPLORE MORE

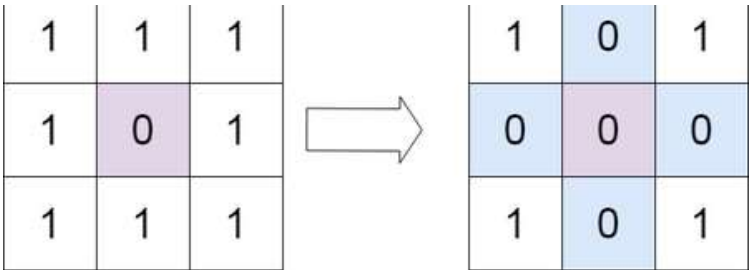
AlgoTutor

# Matrix

## 1. Set Matrix Zeroes

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

**Input:** matrix = [[1,1,1],[1,0,1],[1,1,1]]
**Output:** [[1,0,1],[0,0,0],[1,0,1]]

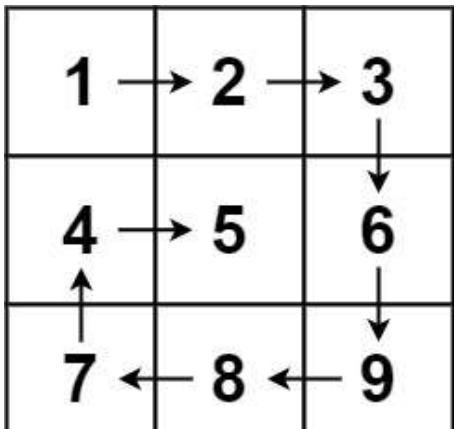



**PRACTICE NOW**

## 2. Spiral Matrix

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

**Input**: matrix = [[1,2,3],[4,5,6],[7,8,9]]
**Output**: [1,2,3,6,9,8,7,4,5]




**PRACTICE NOW**

# 3. Rotate Image

**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.**

**Input:** matrix = [[1,2,3],[4,5,6],[7,8,9]]
**Output:** [[7,4,1],[8,5,2],[9,6,3]]

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

⇨

| | | |
|---|---|---|
| 7 | 4 | 1 |
| 8 | 5 | 2 |
| 9 | 6 | 3 |

**PRACTICE NOW**

# 4. Word Search

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

**Input**: board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "ABCCED"
**Output**: true

| | | | |
|---|---|---|---|
| A | B | C | E |
| S | F | C | S |

AlgoTutor

# String

## 1. Longest Substring Without Repeating Characters

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

**Input**: s = "abcabcbb"
**Output**: 3
**Explanation**: The answer is "abc", with a length of 3.

PRACTICE NOW

## 2. Longest Repeating Character Replacement

**You are given a string s and an integer k. You can choose any character of the string and change it to any other uppercase English character. You can perform this operation at most k times.**

**Return the length of the longest substring containing the same letter you can get after performing the above operations.**

**Input**: s = "ABAB", k = 2
**Output**: 4
**Explanation**: Replace the two 'A's with two 'B's or vice versa.

PRACTICE NOW

# 3. Minimum Window Substring

**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 "".**

**Input:** s = "ADOBECODEBANC", t = "ABC"
**Output:** "BANC"
**Explanation:** The minimum window substring "BANC" includes 'A', 'B', and 'C' from string t.

PRACTICE NOW

# 4. Valid Anagram

**Given two strings s and t, return true if t is an anagram of s, and false otherwise.**

**An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.**

**Input:** s = "anagram", t = "nagaram"
**Output:** true

**Input**: s = "rat", t = "car"
**Output**: false

PRACTICE NOW

# 5. Group Anagrams

**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, typically using all the original letters exactly once.**

**Input**: strs = ["eat","tea","tan","ate","nat","bat"]
**Output**: [["bat"],["nat","tan"],["ate","eat","tea"]]

PRACTICE NOW

# 6. Valid Parentheses

**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.

Input: s = "()"
Output: true

**Input:** s = "()[]{}"
**Output:** true

PRACTICE NOW

# 7. Longest Palindromic Substring

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

**Input**: s = "babad"
**Output**: "bab"
**Explanation**: "aba" is also a valid answer.

**Input**: s = "cbbd"
**Output**: "bb"

PRACTICE NOW

# 8. Palindromic Substrings

**Given a string s, return the number of palindromic substrings in it. A string is a palindrome when it reads the same backward as forward.**

**A substring is a contiguous sequence of characters within the string.**

**Input**: s = "abc"
**Output**: 3
**Explanation**: Three palindromic strings: "a", "b", "c".

**Input:** s = "aaa"
**Output:** 6
**Explanation:** Six palindromic strings: "a", "a", "a", "aa", "aa", "aaa".

PRACTICE NOW

AlgoTutor

# 9. Encode and Decode Strings

**Design an algorithm to encode a list of strings to a string. The encoded string is then sent over the network and decoded back to the original list of strings.**

**implement encode and decode**

**Input**: [“lint”,“code”,“love”,“you”]
**Output**: [“lint”,“code”,“love”,“you”]
**Explanation**: One possible encode method is: “lint:;code:;love:;you”

**Input**: [“we”, “say”, “:”, “yes”]
**Output**: [“we”, “say”, “:”, “yes”]
**Explanation**: One possible encode method is: “we:;say:;::;yes”

**PRACTICE NOW**


AlgoTutor

AlgoTutor is a great e-learning platform for anyone who is looking to learn coding, programming, data structures and algorithms. With comprehensive courses, experienced instructors, and flexible pricing, it's a great option for both beginners and experienced learners alike.

CONTACT US TO LEARN MORE

Info@algotutor.io +91-7260058093

Want to up your Skill?
Join our Popular courses

DSA with System Design (HLD + LLD)

DSA & System Design with Full Stack

Advanced Data Science & Machine Learning

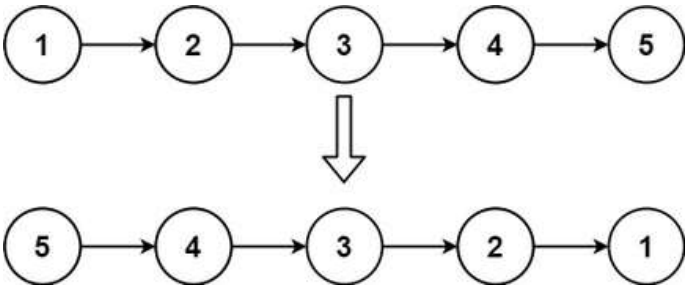
MERN Full Stack Web Development

AlgoTutor

# Linked List

## 1. Reverse Linked Lists

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




**Input**: head = [1,2,3,4,5]
**Output**: [5,4,3,2,1]

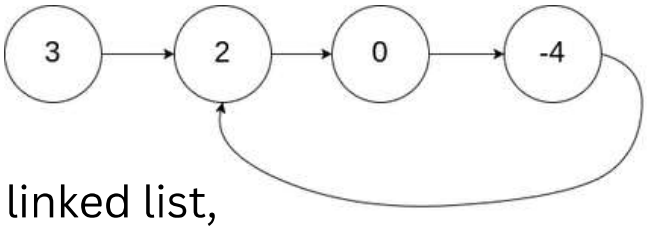
**PRACTICE NOW**

## 2. Linked List Cycle

**Given the 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 the 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.**




**Input:** head = [3,2,0,-4], pos = 1
**Output:** true
**Explanation:** There is a cycle in the linked list, where the tail connects to the 1st node (0-indexed).

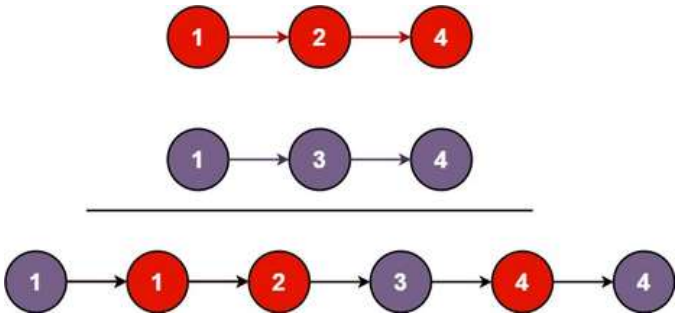
# 3. Merge Two Sorted Lists

**You are given the heads of two sorted linked lists list1 and list2. Merge the two lists in a 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.**

**Input**: list1 = [1,2,4], list2 = [1,3,4]
**Output**: [1,1,2,3,4,4]

**PRACTICE NOW**




# 4. Merge k Sorted Lists

**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.**

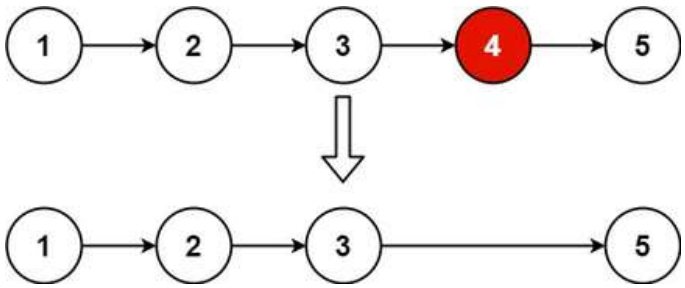
**Input:** lists = [[1,4,5],[1,3,4],[2,6]]
**Output:** [1,1,2,3,4,4,5,6]
**Explanation:** The linked lists are:
[
  1->4->5,
  1->3->4,
  2->6
]
merging them into one sorted list:
1->1->2->3->4->4->5->6

AlgoTutor

# 5. Remove Nth Node From End of List

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

**Input:** head = [1,2,3,4,5], n = 2
**Output:** [1,2,3,5]




**PRACTICE NOW**

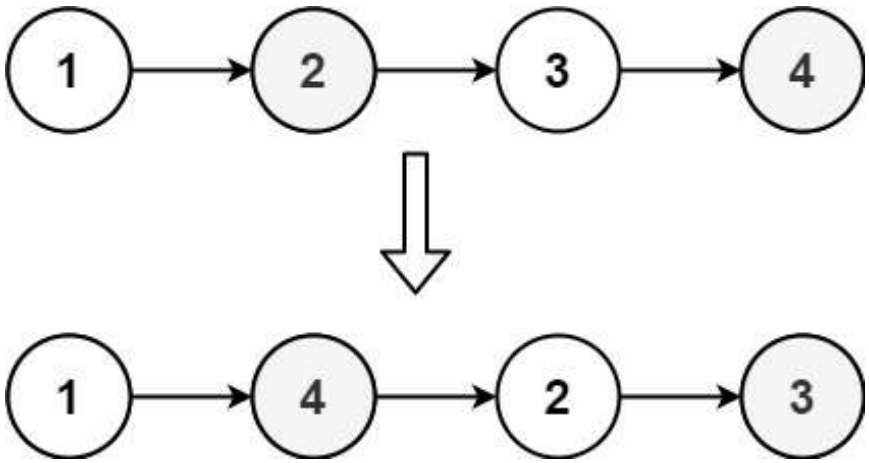
# 6. Reorder List

**You are given the head of a singly linked list. The list can be represented as:**
**L0 → L1 → … → Ln - 1 → Ln**

Reorder the list to be on the following form:
**L0 → Ln → L1 → Ln - 1 → L2 → Ln - 2 → …**

**You may not modify the values in the list's nodes. Only nodes themselves may be changed.**

**Input**: head = [1,2,3,4]
**Output**: [1,4,2,3]




**PRACTICE NOW**

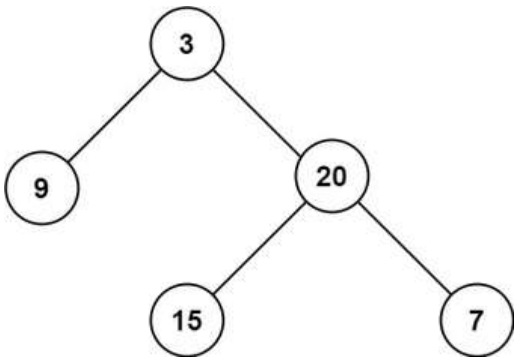
AlgoTutor

# Tree

## 1. Maximum Depth of Binary Tree

**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.**

**Input**: root = [3,9,20,null,null,15,7]
**Output**: 3



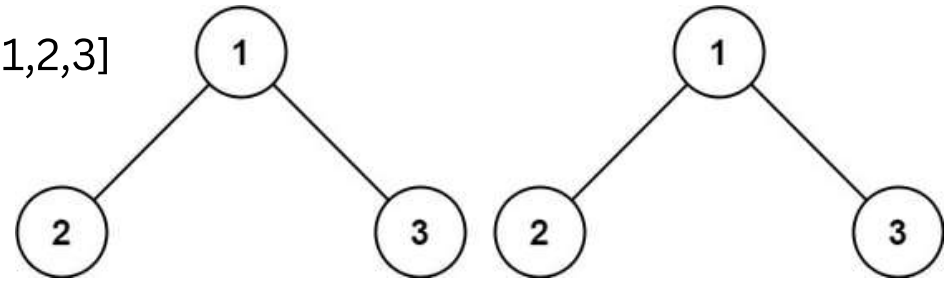

**PRACTICE NOW**

## 2. Same Tree

**Given the roots of two binary trees p and q, write a function to check if they are the same or not.**

**Two binary trees are considered the same if they are structurally identical, and the nodes have the same value.**

**Input:** p = [1,2,3], q = [1,2,3]
**Output:** true

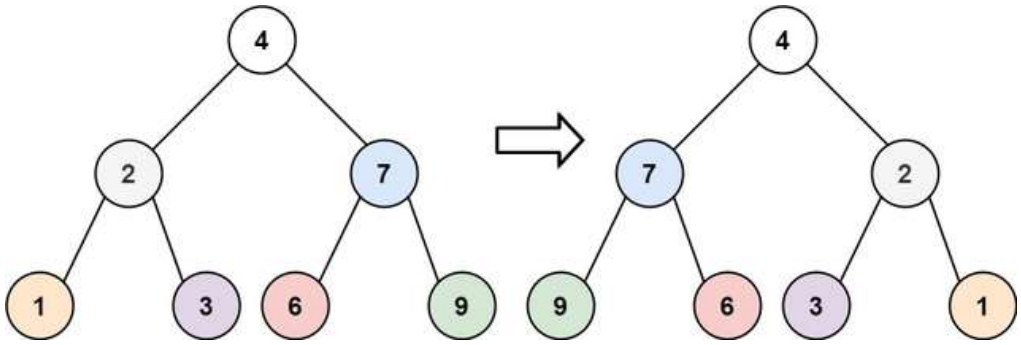



**PRACTICE NOW**

# 3. Invert Binary Tree

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

**Input**: root = [4,2,7,1,3,6,9]
**Output**: [4,7,2,9,6,3,1]



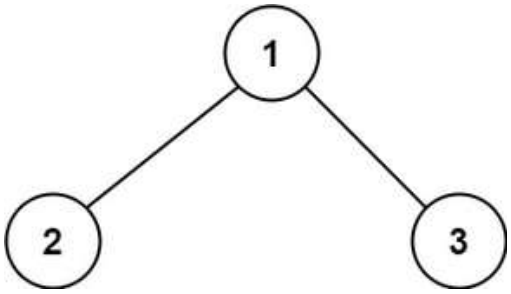

**PRACTICE NOW**

# 4. Binary Tree Maximum Path Sum

**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.**

**Input**: root = [1,2,3]
**Output**: 6
**Explanation**: The optimal path is 2 -> 1 -> 3 with a path sum of 2 + 1 + 3 = 6.



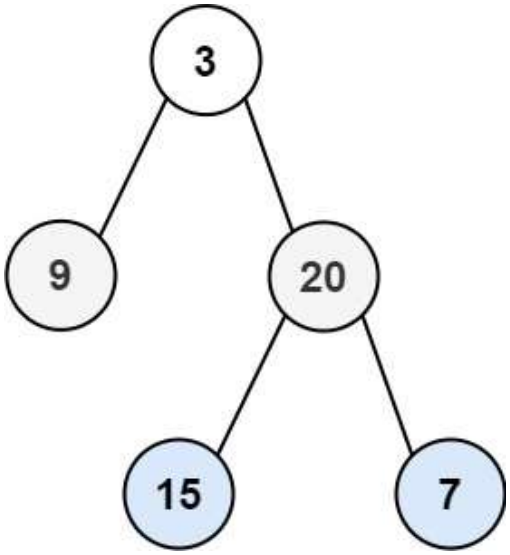
AlgoTutor

# 5. Binary Tree Level Order Traversal

**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).**

**Input**: root = [3,9,20,null,null,15,7]
**Output**: [[3],[9,20],[15,7]]

**PRACTICE NOW**



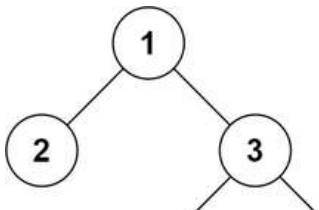

# 6. Serialize & Deserialize Binary Tree

**Serialization is the process of converting a data structure or object into a sequence of bits so that it can be stored in a file or memory buffer, or transmitted across a network connection link to be reconstructed later in the same or another computer environment.**

**Design an algorithm to serialize and deserialize a binary tree. There is no restriction on how your serialization/deserialization algorithm should work.**

**You just need to ensure that a binary tree can be serialized to a string and this string can be deserialized to the original tree structure.**

**Input:** root = [1,2,3,null,null,4,5]
**Output:** [1,2,3,null,null,4,5]



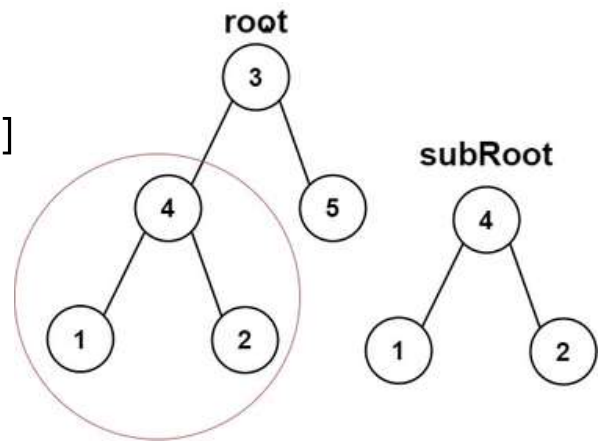
# 7. Subtree of Another Tree

**Given the roots of two binary trees root and subRoot, return true if there is a subtree of root with the same structure and node values of subRoot and false otherwise.**

**A subtree of a binary tree is a tree that consists of a node in a tree and all of this node's descendants. The tree could also be considered a subtree of itself.**

**Input**: root = [3,4,5,1,2], subRoot = [4,1,2]
**Output**: true

**PRACTICE NOW**

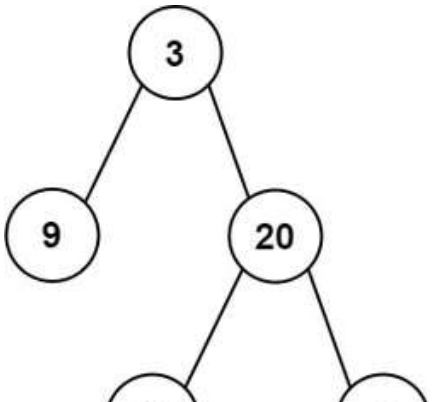



# 8. Construct Binary Tree from Preorder and Inorder Traversal

**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.**

**Input**: preorder = [3,9,20,15,7],
inorder = [9,3,15,20,7]
**Output**: [3,9,20, null, null,15,7]

**PRACTICE NOW**



AlgoTutor

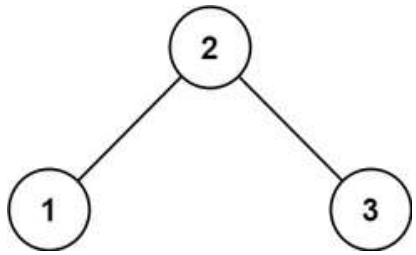
# 9. Validate Binary Search Tree

**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 less than the node's key.
- The right subtree of a node contains only nodes with keys greater than the node's key.
- Both the left and right subtrees must also be binary search trees.

**PRACTICE NOW**

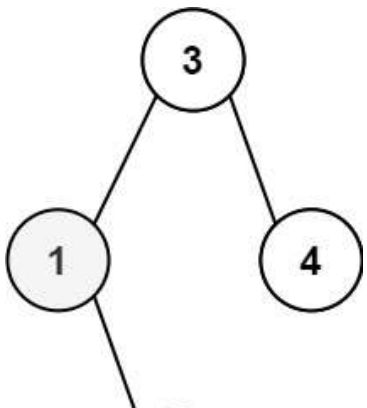



# 10. Kth Smallest Element in a BST

**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.**

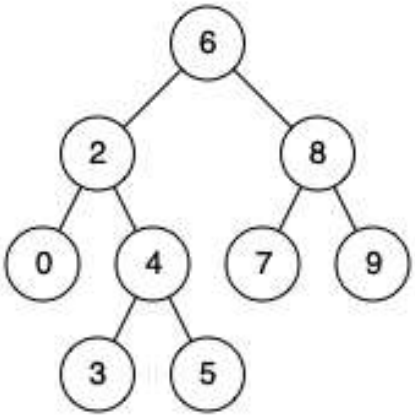
**Input:** root = [3,1,4,null,2], k = 1
**Output:** 1

**PRACTICE NOW**



# 11. Lowest Common Ancestor of a Binary Search Tree

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




**Input**: root = [6,2,8,0,4,7,9, null, null,3,5], p = 2, q = 8
**Output**: 6
**Explanation**: The LCA of nodes 2 and 8 is 6.

**PRACTICE NOW**

# 12. Implement Trie (Prefix Tree)

**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, and false otherwise.

**Input**: ["Trie", "insert", "search", "search", "startsWith", "insert", "search"] [[], ["apple"], ["apple"], ["app"], ["app"], ["app"], ["app"]]
**Output**: [null, null, true, false, true, null, true]

# 13. Design Add and Search Words Data Structure

**Design a data structure that supports adding new words and finding if a string matches any previously added string.**
**Implement the WordDictionary class:**

- WordDictionary() Initializes the object.
- void addWord(word) Adds a word to the data structure, it can be matched later.
- bool search(word) Returns true if there is any string in the data structure that matches the word or false otherwise. A word may contain dots '.' where dots can be matched with any letter.

**Input**:["WordDictionary", "addWord", "addWord", "addWord", "search", "search", "search", "search"]
[[],["bad"],["dad"],["mad"],["pad"],["bad"],[".ad"],["b.."]]
**Output**: [null, null, null, null, false, true, true, true]

**PRACTICE NOW**

# 14. Word Search II

**Given an m x n board of characters and a list of strings words, return all words on the board. Each word must 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 in a word.**

**Input**: board = [["o","a","a","n"],["e","t","a","e"],["i","h","k","r"],["i","f","l","v"]], words = ["oath","pea","eat","rain"]
**Output**: ["eat","oath"]

AlgoTutor

# Want to up your Skill?
# Join our Popular courses

DSA WITH SYSTEM DESIGN (HLD + LLD)

DSA & SYSTEM DESIGN WITH FULL STACK

ADVANCED DATA SCIENCE & MACHINE LEARNING

MERN FULL STACK WEB DEVELOPMENT

EXPLORE MORE

AlgoTutor

# Heap

## 1. Merge k Sorted Lists

**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.**

**Input**: lists = [[1,4,5],[1,3,4],[2,6]]
**Output**: [1,1,2,3,4,4,5,6]
**Explanation**: The linked lists are:
[
  1->4->5,
  1->3->4,
  2->6
]
merging them into one sorted list:
1->1->2->3->4->4->5->6

**PRACTICE NOW**

## 2. Top K Frequent Elements

**Given an integer array nums and an integer k, return the k most frequent elements.**

**You may return the answer in any order.**

**Input**: nums = [1,1,1,2,2,3], k = 2
**Output**: [1,2]

# 3. Find Median from Data Stream

**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.

**Input**: ["MedianFinder", "addNum", "addNum", "findMedian", "addNum", "findMedian"]
[[], [1], [2], [], [3], []]
**Output**: [null, null, null, 1.5, null, 2.0]

**Explanation**:
MedianFinder medianFinder = new MedianFinder();
medianFinder.addNum(1);  // arr = [1]
medianFinder.addNum(2);  // arr = [1, 2]
medianFinder.findMedian(); // return 1.5 (i.e., (1 + 2) / 2)
medianFinder.addNum(3);  // arr[1, 2, 3]
medianFinder.findMedian(); // return 2.0

PRACTICE NOW

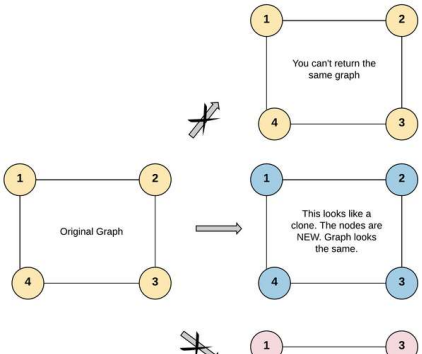
# Graph

## 1. Clone Graph

**Given a reference of a node in a connected undirected graph.**
**Return a deep copy (clone) of the graph.**
**Each node in the graph contains a value (int) and a list (List[Node]) of its neighbors.**

```
class Node {
   public int val;
   public List<Node> neighbors;
}
```

**Test case format:**
For simplicity, each node's value is the same as the node's index (1-indexed). For example, the first node with val == 1, the second node with val == 2, and so on. The graph is represented in the test case using an adjacency list.
**An adjacency list** is a collection of unordered **lists** used to represent a finite graph. Each list describes the set of neighbors of a node in the graph.
The given node will always be the first node with val = 1.
You must return the **copy of the given node** as a reference to the cloned graph.




**Input**: adjList = [[2,4],[1,3],[2,4],[1,3]]
**Output**: [[2,4],[1,3],[2,4],[1,3]]

# 2. Course Schedule

**There are a total of numCourses courses you have to take, labeled from 0 to numCourses - 1. You are given an array of 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.**

**Input**: numCourses = 2, prerequisites = [[1,0]]
**Output**: true
**Explanation**: There are a total of 2 courses to take.
To take course 1 you should have finished course 0. So it is possible.

PRACTICE NOW

# 3. Number of Islands

**Given an m x n 2D binary 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.**

**Input**: grid = [["1","1","1","1","0"],
["1","1","0","1","0"],
["1","1","0","0","0"],
["0","0","0","0","0"]]
**Output**: 1

# 4. Pacific Atlantic Water Flow

**There is an m x n rectangular island that borders both the Pacific Ocean and the Atlantic Ocean. The Pacific Ocean touches the island's left and top edges, and the Atlantic Ocean touches the island's right and bottom edges.**

**The island is partitioned into a grid of square cells. You are given an m x n integer matrix height where heights[r][c] represent the height above sea level of the cell at coordinate (r, c).**

**The island receives a lot of rain, and the rainwater can flow to neighboring cells directly north, south, east, and west if the neighboring cell's height is less than or equal to the current cell's height. Water can flow from any cell adjacent to an ocean into the ocean.**

**Return a 2D list of grid coordinates results where result[i] = [ri, ci] denotes that rainwater can flow from cell (ri, ci) to both the Pacific and Atlantic oceans.**

**Input**: heights = [[1,2,2,3,5],[3,2,3,4,4],[2,4,5,3,1],[6,7,1,4,5],[5,1,1,2,4]]
**Output**: [[0,4],[1,3],[1,4],[2,2],[3,0],[3,1],[4,0]]

PRACTICE NOW

# 5. Longest Consecutive Sequence

**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.**

**Input:** nums = [100,4,200,1,3,2]
**Output:** 4
**Explanation:** The longest consecutive elements sequence is [1, 2, 3, 4]. Therefore its length is 4.

**PRACTICE NOW**

# 6. Alien Dictionary

**There is a new alien language that uses the Latin alphabet. However, the order of letters is unknown to you. You receive a list of non-empty words from the dictionary, where words are sorted lexicographically by the rules of this new language. Derive the order of letters in this language.**

**Input**:
[
 "wrt",
 "wrf",
 "er",
 "ett",
 "rftt"
]
**Output**: "wertf"

# 7. Graph Valid Tree

**Given n nodes labeled from 0 to n-1 and a list of undirected edges (each edge is a pair of nodes), write a function to check whether these edges make up a valid tree.**

**Input**: n = 5, and edges = [[0,1], [0,2], [0,3], [1,4]]
**Output**: true

**PRACTICE NOW**

# 8. Number of Connected Components in an Undirected Graph

**Given n nodes labeled from 0 to n - 1 and a list of undirected edges (each edge is a pair of nodes), write a function to find the number of connected components in an undirected graph.**

**Input:** n = 5 and edges = [[0, 1], [1, 2], [3, 4]]

```
0        3
|        |
1 --- 2  4
```

**Output:** 2

**PRACTICE NOW**

AlgoTutor

# Dynamic Programming

## 1. Climbing Stairs

**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?**

**Input**: n = 2
**Output**: 2
**Explanation**: There are two ways to climb to the top.
1. 1 step + 1 step
2. 2 steps

**PRACTICE NOW**

## 2. Coin Change

**You are given an integer array of 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.**

**Input**: coins = [1,2,5], amount = 11
**Output**: 3
**Explanation**: 11 = 5 + 5 + 1

# 3. Longest Increasing Subsequence

**Given an integer array nums, return the length of the longest strictly increasing subsequence.**

**Input**: nums = [10,9,2,5,3,7,101,18]
**Output**: 4
**Explanation**: The longest increasing subsequence is [2,3,7,101], therefore the length is 4.

**Input**: nums = [0,1,0,3,2,3]
**Output**: 4.

**PRACTICE NOW**

# 4. Longest Common Subsequence

**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.

**Input**: text1 = "abcde", text2 = "ace"
**Output**: 3
**Explanation**: The longest common subsequence is "ace" and its length is 3.

# 5. Word Break

**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.**

**Input**: s = "leetcode", wordDict = ["leet","code"]
**Output**: true
**Explanation**: Return true because "leetcode" can be segmented as "leet code".

PRACTICE NOW

# 6. Combination Sum

**Given an array of distinct integer nums and a target integer target, return the number of possible combinations that add up to the target.**

**Input**: nums = [1,2,3], target = 4
**Output**: 7
**Explanation**: The possible combination ways are:
(1, 1, 1, 1)
(1, 1, 2)
(1, 2, 1)
(1, 3)
(2, 1, 1)
(2, 2)
(3, 1)
Note that different sequences are counted as different combinations

# 7. House Robber

**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 in each house, return the maximum amount of money you can rob tonight without alerting the police.**

**Input**: nums = [1,2,3,1]
**Output**: 4

**PRACTICE NOW**

# 8. House Robber II

**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 a security system 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 in each house, return the maximum amount of money you can rob tonight without alerting the police.**

**Input**: nums = [2,3,2]
**Output**: 3

# 9. Decode Ways

**A message containing letters from A-Z can be encoded into numbers using the following mapping:**
'A' -> "1", 'B' -> "2",... 'Z' -> "26"
**To decode an encoded message, all the digits must be grouped and then mapped back into letters using the reverse of the mapping above (there may be multiple ways). For example, "11106" can be mapped into:**

- "AAJF" with the grouping (1 1 10 6)
- "KJF" with the grouping (11 10 6)

**Given a string s containing only digits, return the number of ways to decode it.**

**Input**: s = "12"
**Output**: 2

**PRACTICE NOW**

# 10. Unique Paths

**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.**

**Input**: m = 3, n = 2
**Output**: 3

# 11. Jump Game

**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.**

**Input**: nums = [2,3,1,1,4]
**Output**: true
**Explanation**: Jump 1 step from index 0 to 1, then 3 steps to the last index.

**Input:** nums = [3,2,1,0,4]
**Output:** false
**Explanation:** You will always arrive at index 3 no matter what. Its maximum jump length is 0, which makes it impossible to reach the last index.

**Constraints**:

- 1 <= nums.length <= 104
- 0 <= nums[i] <= 105

**PRACTICE NOW**

For Admission Enquiry
+91-7260058093
info@algotutor.io

AlgoTutor

# Interval

## 1. Insert Interval

**You are given an array of non-overlapping intervals where intervals[i] = [starti, endi] represent the start and the end of the ith interval, and intervals are sorted in ascending order by start i. You are also given an interval new interval = [start, end] that represents the start and end of another interval.**

**Insert new intervals into intervals such that intervals are still sorted in ascending order by start i and intervals still do not have any overlapping intervals (merge overlapping intervals if necessary). Return intervals after the insertion.**

**Input**: intervals = [[1,3],[6,9]], newInterval = [2,5]
**Output**: [[1,5],[6,9]]

**PRACTICE NOW**

## 2. Merge Intervals

**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.**

**Input**: intervals = [[1,3],[2,6],[8,10],[15,18]]
**Output**: [[1,6],[8,10],[15,18]]
**Explanation**: Since intervals [1,3] and [2,6] overlap, merge them into [1,6].

# 3. Non-overlapping Intervals

**Given an array of intervals where intervals[i] = [starti, endi], return the minimum number of intervals you need to remove to make the rest of the intervals non-overlapping.**

**Input**: intervals = [[1,2],[2,3],[3,4],[1,3]]
**Output**: 1
**Explanation**: [1,3] can be removed and the rest of the intervals are non-overlapping.

PRACTICE NOW

# 4. Meeting Rooms

**Given an array of meeting time intervals consisting of start and end times[[s1,e1],[s2,e2],...](si< ei), determine if a person could attend all meetings.**

**Input**: [[0,30],[5,10],[15,20]]
**Output**: false

**Input**: [[7,10],[2,4]]
**Output**: true

PRACTICE NOW

# 5. Meeting Rooms II

**Given an array of meeting time intervals consisting of start and end times[[s1,e1],[s2,e2],...](si< ei).**

**Find the minimum number of conference rooms required.**

**Input**: [[0, 30],[5, 10],[15, 20]]
**Output**: 2

**Input**: [[7,10],[2,4]]
**Output**: 1

**PRACTICE NOW**


## Want to up your Skill?
## Join our Popular courses

DSA with System Design (HDL+LLD)

DSA & System Design with Full Stack

Advanced Data Science & Machine Learning

MERN Full Stack Web Development

AlgoTutor

# Bitwise Operator

## 1. Sum of Two Integers

**Given two integers a and b, return the sum of the two integers without using the operators + and -.**

**Input**: a = 1, b = 2
**Output**: 3

**Input**: a = 2, b = 3
**Output**: 5

**PRACTICE NOW**

## 2. Number of 1 Bit

**Write a function that takes the binary representation of an unsigned integer and returns the number of '1' bits it has (also known as the Hamming weight).**

**Input:** n = 00000000000000000000000000001011
**Output:** 3
**Explanation:** The input binary string **00000000000000000000000000001011** has a total of three '1' bits.

**PRACTICE NOW**

# 3. Counting Bits

**Given an integer n, return an array of length n + 1 such that for each i (0 <= i <= n), ans[i] is the number of 1's in the binary representation of i.**

**Input**: n = 2
**Output**: [0,1,1]
**Explanation**:
0 --> 0
1 --> 1
2 --> 10

**PRACTICE NOW**

# 4. Missing Number

**Given an array nums containing n distinct numbers in the range [0, n], return the only number in the range that is missing from the array.**

**Input:** nums = [3,0,1]
**Output:** 2
**Explanation:** n = 3 since there are 3 numbers, so all numbers are in the range [0,3]. 2 is the missing number in the range since it does not appear in nums.

**Input**: nums = [0,1]
**Output**: 2
**Explanation**: n = 2 since there are 2 numbers, so all numbers are in the range [0,2]. 2 is the missing number in the range since it does not appear in nums.

# 5. Reverse Bits

**Reverse bits of a given 32-bit unsigned integer.**

**Note:**

- Note that in some languages, such as Java, there is no unsigned integer type. In this case, both input and output will be given as a signed integer type. They should not affect your implementation, as the integer's internal binary representation is the same, whether it is signed or unsigned.
- In Java, the compiler represents the signed integers using 2's complement notation. Therefore, in Example 2 above, the input represents the signed integer -3 and the output represents the signed integer -1073741825.

**Input:** n = 00000010100101000001111010011100
**Output:** 964176192 (00111001011110000010100101000000)
**Explanation:** The input binary string **00000010100101000001111010011100** represents the unsigned integer 43261596, so return 964176192 whose binary representation is **00111001011110000010100101000000**.

**Input**: n = 11111111111111111111111111111101
**Output**: 3221225471 (10111111111111111111111111111111)
**Explanation**: The input binary string 11111111111111111111111111111101 represents the unsigned integer 4294967293, so return 3221225471 whose binary representation is 10111111111111111111111111111111.

**PRACTICE NOW**

AlgoTutor

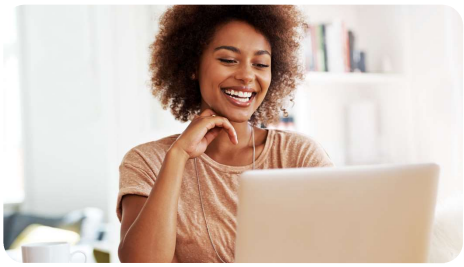
# Why Choose AlgoTutor?

- 100 % PLACEMENT ASSISTANCE
- 1:1 PERSONAL MENTORSHIP FROM INDUSTRY EXPERTS
- 100 % SUCCESS RATE
- 23 LPA(AVG.)CTC
- 200+ SUCCESSFUL ALUMNI
- 147%(AVG.) SALARY HIKE
- LEARN FROM SCRATCH
- CAREER SERVICES

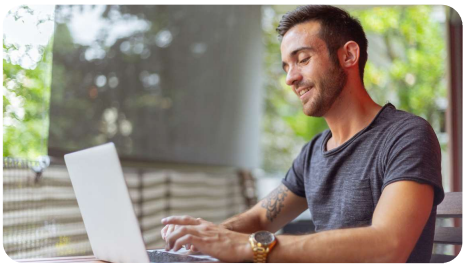
EXPLORE MORE

AlgoTutor

# Want to up your Skill?
# Join our Popular courses



DSA WITH SYSTEM DESIGN (HLD + LLD)



DSA & SYSTEM DESIGN WITH FULL STACK

ADVANCED DATA SCIENCE & MACHINE LEARNING

MERN FULL STACK WEB DEVELOPMENT

EXPLORE MORE