**100 coding Interview Questions**

**Arrays and Strings**

1. **Find the Missing Number in an Array**  
   Given an array of n-1 integers ranging from 1 to n with no duplicates, find the missing number. The array contains distinct integers.  
   **Example:**  
   Input: [1, 2, 4, 5, 6]  
   Output: 3
2. **Find the Duplicate Number**  
   Given an array containing n integers where each number is between 1 and n-1, find the duplicate number.  
   **Example:**  
   Input: [3, 1, 3, 4, 2]  
   Output: 3
3. **Find the Majority Element (Moore’s Voting Algorithm)**  
   Given an array of size n, find the majority element (an element that appears more than n/2 times).  
   **Example:**  
   Input: [3, 3, 4, 2, 4, 4, 2, 4, 4]  
   Output: 4
4. **Two Sum Problem**  
   Given an array of integers and a target number, return the indices of the two numbers such that their sum is equal to the target.  
   **Example:**  
   Input: nums = [2, 7, 11, 15], target = 9  
   Output: [0, 1]
5. **Maximum Subarray Sum (Kadane’s Algorithm)**  
   Given an integer array, find the contiguous subarray (containing at least one number) that has the largest sum.  
   **Example:**  
   Input: [-2, 1, -3, 4, -1, 2, 1, -5, 4]  
   Output: 6 (Subarray [4, -1, 2, 1])
6. **Product of Array Except Self**  
   Given an array of numbers, return an array such that each element at index i is the product of all the numbers in the original array except the one at index i.  
   **Example:**  
   Input: [1, 2, 3, 4]  
   Output: [24, 12, 8, 6]
7. **Merge Intervals**  
   Given a collection of intervals, merge all overlapping intervals.  
   **Example:**  
   Input: [[1,3], [2,6], [8,10], [15,18]]  
   Output: [[1,6], [8,10], [15,18]]
8. **Container with Most Water**  
   Given an array where each element represents the height of a vertical line, find two lines that together with the x-axis form a container that holds the most water.  
   **Example:**  
   Input: [1,8,6,2,5,4,8,3,7]  
   Output: 49
9. **Trapping Rain Water**  
   Given an array representing the height of walls, calculate how much water can be trapped after raining.  
   **Example:**  
   Input: [0,1,0,2,1,0,1,3,2,1,2,1]  
   Output: 6
10. **Find the Kth Largest Element**  
    Given an integer array, find the kth largest element.  
    **Example:**  
    Input: [3, 2, 1, 5, 6, 4], k = 2  
    Output: 5
11. **3 Sum Problem**  
    Given an array of n integers, find all unique triplets in the array that sum to zero.  
    **Example:**  
    Input: [-1, 0, 1, 2, -1, -4]  
    Output: [[-1, -1, 2], [-1, 0, 1]]
12. **Largest Number Formed from an Array**  
    Given a list of non-negative integers, arrange them in such a way that they form the largest number.  
    **Example:**  
    Input: [3, 30, 34, 5, 9]  
    Output: 9534330
13. **Move Zeroes to the End**  
    Given an array of integers, move all the zeroes to the end while maintaining the relative order of the non-zero elements.  
    **Example:**  
    Input: [0, 1, 2, 0, 3, 4]  
    Output: [1, 2, 3, 4, 0, 0]
14. **Longest Substring Without Repeating Characters**  
    Given a string, find the length of the longest substring without repeating characters.  
    **Example:**  
    Input: "abcabcbb"  
    Output: 3 (Substring "abc")
15. **Permutations of a String**  
    Given a string, find all possible permutations of the string.  
    **Example:**  
    Input: "abc"  
    Output: ["abc", "acb", "bac", "bca", "cab", "cba"]
16. **Palindrome Partitioning**  
    Given a string, partition it such that every substring is a palindrome. Return all possible palindrome partitioning.  
    **Example:**  
    Input: "aab"  
    Output: [["a", "a", "b"], ["aa", "b"]]
17. **Zigzag Conversion**  
    Given a string and a number of rows, convert the string into a zigzag pattern on the given number of rows and read line by line.  
    **Example:**  
    Input: "PAYPALISHIRING", 3  
    Output: "PAHNAPLSIIGYIR"
18. **Group Anagrams**  
    Given a list of strings, group the anagrams together.  
    **Example:**  
    Input: ["eat", "tea", "tan", "ate", "nat", "bat"]  
    Output: [["eat", "tea", "ate"], ["tan", "nat"], ["bat"]]
19. **Rotate Array**  
    Given an array and an integer k, rotate the array to the right by k steps.  
    **Example:**  
    Input: [1, 2, 3, 4, 5, 6, 7], k = 3  
    Output: [5, 6, 7, 1, 2, 3, 4]
20. **Find the Longest Palindromic Substring**  
    Given a string, return the longest palindromic substring.  
    **Example:**  
    Input: "babad"  
    Output: "bab" (or "aba")
21. **Next Permutation**  
    Implement the function to get the next lexicographically greater permutation of an integer array.  
    **Example:**  
    Input: [1, 2, 3]  
    Output: [1, 3, 2]
22. **Find All Substrings of a String**  
    Given a string, find all possible substrings.  
    **Example:**  
    Input: "abc"  
    Output: ["a", "b", "c", "ab", "bc", "abc"]
23. **Longest Common Subsequence**  
    Given two strings, find the length of the longest common subsequence (LCS).  
    **Example:**  
    Input: "abcde", "ace"  
    Output: 3 (LCS is "ace")
24. **Reverse a Linked List**  
    Given a singly linked list, reverse the list and return the reversed list.  
    **Example:**  
    Input: 1 → 2 → 3 → 4 → 5 → NULL  
    Output: 5 → 4 → 3 → 2 → 1 → NULL
25. **Detect a Cycle in a Linked List (Floyd’s Cycle Detection)**  
    Given a linked list, determine if there is a cycle in it. Use Floyd’s Tortoise and Hare algorithm to detect a cycle.  
    **Example:**  
    Input: 1 → 2 → 3 → 4 → 2 (cycle starts at node 2)  
    Output: True
26. **Merge Two Sorted Linked Lists**  
    Given two sorted linked lists, merge them into a single sorted linked list.  
    **Example:**  
    Input: 1 → 2 → 4, 1 → 3 → 4  
    Output: 1 → 1 → 2 → 3 → 4 → 4
27. **Find the Middle Element of a Linked List**  
    Given a singly linked list, find the middle element.  
    **Example:**  
    Input: 1 → 2 → 3 → 4 → 5  
    Output: 3
28. **Detect and Remove Loop in a Linked List**  
    Given a linked list that may contain a cycle, detect and remove the loop.  
    **Example:**  
    Input: 1 → 2 → 3 → 4 → 5 → 2 (loop starts at node 2)  
    Output: 1 → 2 → 3 → 4 → 5
29. **Flatten a Linked List**  
    Flatten a multilevel doubly linked list where each node may have a next and a child pointer.  
    **Example:**  
    Input: 1 → 2 → 3 → 4 → 5 → 6  
    Output: 1 → 2 → 3 → 4 → 5 → 6
30. **Add Two Numbers Represented by Linked Lists**  
    Given two linked lists representing non-negative integers, where each node represents a digit, add the two numbers and return the result as a linked list.  
    **Example:**  
    Input: 2 → 4 → 3, 5 → 6 → 4  
    Output: 7 → 0 → 8
31. **Intersection of Two Linked Lists**  
    Given two singly linked lists, find the node where the two lists intersect.  
    **Example:**  
    Input: 1 → 2 → 3 → 4 → 5, 6 → 7 → 8 → 3 → 4 → 5  
    Output: 3
32. **Rotate a Linked List**  
    Given a linked list, rotate the list to the right by k places.  
    **Example:**  
    Input: 1 → 2 → 3 → 4 → 5, k = 2  
    Output: 4 → 5 → 1 → 2 → 3
33. **Merge K Sorted Linked Lists**  
    Given k sorted linked lists, merge them into a single sorted linked list.  
    **Example:**  
    Input: [[1 → 4 → 5], [1 → 3 → 4], [2 → 6]]  
    Output: 1 → 1 → 2 → 3 → 4 → 4 → 5 → 6
34. **Rearrange a Linked List (Alternating Nodes)**  
    Rearrange a linked list so that all odd-positioned nodes appear first and all even-positioned nodes appear later.  
    **Example:**  
    Input: 1 → 2 → 3 → 4 → 5  
    Output: 1 → 3 → 5 → 2 → 4
35. **Palindrome Linked List**  
    Given a singly linked list, determine whether it is a palindrome.  
    **Example:**  
    Input: 1 → 2 → 2 → 1  
    Output: True
36. **Clone a Linked List with Random Pointers**  
    Clone a linked list where each node has two pointers: one to the next node and another to a random node.  
    **Example:**  
    Input: 1 → 2 → 3 → 4 with random pointers to arbitrary nodes.  
    Output: A deep copy of the list.
37. **Find the Nth Node from the End of a Linked List**  
    Given a singly linked list, find the nth node from the end of the list.  
    **Example:**  
    Input: 1 → 2 → 3 → 4 → 5, n = 2  
    Output: 4
38.  **Implement Stack Using Queues**  
    Implement a stack using two queues.  
    **Example:**  
    Operations: push(1), push(2), pop()  
    Output: 2
39.  **Implement Queue Using Stacks**  
    Implement a queue using two stacks.  
    **Example:**  
    Operations: enqueue(1), enqueue(2), dequeue()  
    Output: 1
40.  **Valid Parentheses**  
    Given a string containing just the characters ()[]{}, determine if the input string is valid.  
    **Example:**  
    Input: "{[()]}"  
    Output: True
41.  **Evaluate Reverse Polish Notation**  
    Evaluate the value of an arithmetic expression in Reverse Polish Notation.  
    **Example:**  
    Input: ["2", "1", "+", "3", "\*"]  
    Output: 9
42.  **Next Greater Element**  
    Given an array, find the next greater element for each element.  
    **Example:**  
    Input: [4, 5, 2, 10]  
    Output: [5, 10, 10, -1]
43.  **Daily Temperature Problem**  
    Given a list of daily temperatures, return a list of days you have to wait until a warmer temperature.  
    **Example:**  
    Input: [73, 74, 75, 71, 69, 72, 76, 73]  
    Output: [1, 1, 4, 2, 1, 1, 0, 0]
44.  **Largest Rectangle in Histogram**  
    Given a histogram, find the largest rectangle that can be formed using the bars of the histogram.  
    **Example:**  
    Input: [2, 1, 5, 6, 2, 3]  
    Output: 10
45.  **Sliding Window Maximum**  
    Given an array and a number k, return the maximum value in the sliding window of size k.  
    **Example:**  
    Input: [1,3,-1,-3,5,3,6,7], k = 3  
    Output: [3, 3, 5, 5, 6, 7]
46. **Merge K Sorted Lists**  
    Merge k sorted linked lists into one sorted list.  
    **Example:**  
    Input: [[1,4,5], [1,3,4], [2,6]]  
    Output: [1, 1, 2, 3, 4, 4, 5, 6]
47. **Kth Smallest Element in a Sorted Matrix**  
    Find the kth smallest element in a sorted matrix.  
    **Example:**  
    Input: matrix = [[1,5,9], [10,11,13], [12,13,15]], k = 8  
    Output: 13
48. **Find Median from Data Stream**  
    Given a stream of integers, find the median of the current data stream.  
    **Example:**  
    Input: 1, 2, 3, 4, 5  
    Output: 3
49. **Top K Frequent Elements**  
    Given a list of integers, find the k most frequent elements.  
    **Example:**  
    Input: [1,1,1,2,2,3], k = 2  
    Output: [1, 2]
50. **Sort an Array Using Heap**  
    Given an array, sort it using a heap data structure.  
    **Example:**  
    Input: [3,2,1,5,4]  
    Output: [1,2,3,4,5]
51. **Find the Kth Largest Element in an Array**  
    Find the kth largest element in an unsorted array.  
    **Example:**  
    Input: [3,2,1,5,6,4,7], k = 4  
    Output: 4

**Graphs**

1. **Depth First Search (DFS)**  
   Given a graph, perform depth-first search.  
   **Example:**  
   Input: graph = {0:[1, 2], 1:[0, 3], 2:[0, 3], 3:[1, 2]}  
   Output: DFS traversal
2. **Breadth First Search (BFS)**  
   Given a graph, perform breadth-first search.  
   **Example:**  
   Input: graph = {0:[1, 2], 1:[0, 3], 2:[0, 3], 3:[1, 2]}  
   Output: BFS traversal
3. **Detect a Cycle in a Directed Graph**  
   Given a directed graph, detect if there is a cycle.  
   **Example:**  
   Input: graph = [[1], [2], [3], [1]]  
   Output: True
4. **Detect a Cycle in an Undirected Graph**  
   Given an undirected graph, detect if there is a cycle.  
   **Example:**  
   Input: graph = [[1, 2], [2, 0], [3, 1]]  
   Output: True
5. **Topological Sort**  
   Given a directed acyclic graph (DAG), perform topological sorting.  
   **Example:**  
   Input: graph = [[2, 3], [3], []]  
   Output: [1, 2, 3]
6. **Find Shortest Path in a Weighted Graph (Dijkstra’s Algorithm)**  
   Given a weighted graph, find the shortest path using Dijkstra’s algorithm.  
   **Example:**  
   Input: graph = {(0, 1): 1, (1, 2): 2}, start = 0  
   Output: Shortest path from 0 to other vertices
7. **Find Shortest Path in an Unweighted Graph (BFS)**  
   Given an unweighted graph, find the shortest path using BFS.  
   **Example:**  
   Input: graph = {0: [1], 1: [0, 2], 2: [1]}  
   Output: Shortest path from 0 to other vertices
8. **Number of Islands**  
   Given a grid of 1s (land) and 0s (water), count the number of islands (connected components of 1s).  
   **Example:**  
   Input:

csharp

CopyEdit

[

[1,1,0,0,0],

[1,1,0,0,0],

[0,0,1,0,0],

[0,0,0,1,1]

]

Output: 3

1. **Clone a Graph**  
   Clone an undirected graph, where each node has a value and neighbors.  
   **Example:**  
   Input: graph = {1: [2, 4], 2: [1, 3], 3: [2, 4], 4: [1, 3]}  
   Output: Cloned graph

**Other questions**

**Ishaan Loves Chocolates**

As we know, Ishaan has a love for chocolates. He has bought a huge chocolate bar that contains N chocolate squares. Each of the squares has a tastiness level which is denoted by List  
Ishaan can eat the first or the last square of the chocolate at once. Ishaan has a sister who loves chocolates too and she demands the last chocolate square. Now, Ishaan being greedy eats the more tasty square first.   
Determine the tastiness level of the square which his sister gets.

**Input :** list = [5, 3, 1, 6, 9]

**Output :** 1

**Explanation:**

**Initially: 5 3 1 6 9**

In first step: 5 3 1 6

In Second step: 5 3 1

In Third step: 3 1

In Fourth step: 1

Return 1

**Knapsack problem**

Given weights and values of N items, we need to put these items in a knapsack of capacity W to get the maximum total value in the knapsack.  
Note: you are allowed to break the item.

Approach we required to get maximum profit.

N=3 and W=20

Profits=[25,24,15]

Weights=[18,15,10]

Max profit=31.5 around

**Insertion of Binary search Tree using recursion return dictionary as tree**

Problem:

Bheem has N friends and he has a ladoo for each day given an list denoting on which day i th index friend wants a ladoo if bheem is unable to give a ladoo to the friend he loses his friendship with them find out the maximum number of friends he can have at the end

Example: N=5

List of days=[3,3,1,2,4]

Output=4

Problem:

Given an array **list** of non-negative integers, return the maximum product of two numbers possible.

**Input:** arr[] = [1, 4, 3, 6, 7, 0]

**Output:** 42  
**Explanation:** 6 and 7 have the maximum product.

### ****Job Sequencing Problem****

**Problem Statement:**  
There are n jobs to be done. Each job has a deadline and a profit associated with it. The goal is to schedule the jobs such that the total profit is maximized. A job can only be performed once and must be completed by its deadline.

**Input:**  
An array of jobs, each having a deadline and profit.

**Output:**  
The maximum profit and the sequence of jobs to be completed.

**Example:**  
Input:  
Job[] = { (4, 20), (1, 10), (1, 40), (1, 30) }  
Output:  
Maximum profit: 60  
Job sequence: [ (1, 40), (1, 30) ]

### ****Activity Selection Problem****

**Problem Statement:**  
You are given n activities with their start and finish times. Select the maximum number of activities that can be performed by a single person, assuming that a person can only work on one activity at a time.

**Input:**  
An array of n activities, each having a start and finish time.

**Output:**  
Return the maximum number of activities that can be performed.

**Example:**  
Input:  
Start[] = [1, 3, 0, 5, 8, 5]  
Finish[] = [2, 4, 6, 7, 9, 9]  
Output:  
4 (Activities selected: 1, 2, 4, 5)

### ****Coin Change Problem (Greedy Version)****

**Problem Statement:**  
You are given an infinite supply of coins of different denominations. You need to find the minimum number of coins required to make a given amount.

**Input:**  
A set of coin denominations and a total amount.

**Output:**  
The minimum number of coins required to make the amount.

**Example:**  
Input:  
Coins[] = {1, 5, 10, 25}  
Amount = 30  
Output:  
2 (Using coins 25 and 5)

### ****Minimum Number of Platforms Required****

**Problem Statement:**  
There are n trains arriving and departing at different times at a railway station. Find the minimum number of platforms required for the trains to park, such that no two trains overlap in time.

**Input:**  
An array of arrival times and an array of departure times.

**Output:**  
The minimum number of platforms required.

**Example:**  
Input:  
Arrival[] = [10:00, 10:15, 10:30]  
Departure[] = [10:30, 10:45, 11:00]  
Output:  
2

### ****Cake Cutting Problem****

**Problem Statement:**  
Given n pieces of cake, each with a certain amount of frosting on it, and m people, determine the minimum frosting amount that can be distributed to each person such that no person gets a larger piece than others.

**Input:**  
An array of integers representing frosting amounts and an integer m representing the number of people.

**Output:**  
The minimum maximum frosting amount that a person can receive.

**Example:**  
Input:  
Cake[] = [2, 3, 5, 1, 6], m = 3  
Output:  
8 (Divide into pieces such that the largest piece is 8)

### ****Minimum Coins to Make a Change (Greedy Version)****

**Problem Statement:**  
Given different denominations of coins and an amount, find the minimum number of coins required to make the exact amount using the given denominations.

**Input:**  
An array coins[] and an integer amount.

**Output:**  
The minimum number of coins required.

**Example:**  
Input:  
coins = [1, 2, 5], amount = 11  
Output:  
3 (Using coins 5, 5, and 1)

### ****Subset Sum Problem****

**Problem Statement:**  
Given a set of integers, find all subsets whose sum is equal to a given target number.

**Input:**  
An array arr[] of integers and a target sum S.

**Output:**  
All subsets of arr[] whose sum is equal to S.

**Example:**  
Input:  
arr = [3, 34, 4, 12, 5, 2], S = 9  
Output:  
[[4, 5], [3, 2, 4]]

### ****Generate All Permutations of a String****

**Problem Statement:**  
Given a string, generate all possible permutations of the characters of the string.

**Input:**  
A string s.

**Output:**  
All possible permutations of the string.

**Example:**  
Input: s = "ABC"  
Output:  
["ABC", "ACB", "BAC", "BCA", "CAB", "CBA"]

### ****5. Combination Sum****

**Problem Statement:**  
Given an array of integers candidates and a target target, find all unique combinations of numbers from the array that sum to the target. The same number can be chosen multiple times.

**Input:**  
An array candidates[] and an integer target.

**Output:**  
All combinations that sum to target.

**Example:**  
Input:  
candidates = [2, 3, 6, 7], target = 7  
Output:  
[[2, 2, 3], [7]]

### ****6. Letter Combinations of a Phone Number****

**Problem Statement:**  
Given a string representing digits from 2 to 9, return all possible letter combinations that the number could represent. The mapping of digits to letters is the same as on a telephone keypad.

**Input:**  
A string of digits.

**Output:**  
All possible letter combinations.

**Example:**  
Input: digits = "23"  
Output:  
["ad", "ae", "af", "bd", "be", "bf", "cd", "ce", "cf"]

### ****7. Palindrome Partitioning****

**Problem Statement:**  
Given a string, partition it such that every substring is a palindrome. Return all possible palindrome partitioning of the string.

**Input:**  
A string s.

**Output:**  
A list of lists, where each list is a palindrome partitioning.

**Example:**  
Input:  
s = "aab"  
Output:  
[["a", "a", "b"], ["aa", "b"]]

### ****8. Permute Unique****

**Problem Statement:**  
Given an array of numbers, return all unique permutations of the numbers. The input may contain duplicates.

**Input:**  
An array nums[].

**Output:**  
All unique permutations of the array.

**Example:**  
Input:  
nums = [1, 1, 2]  
Output:  
[[1, 1, 2], [1, 2, 1], [2, 1, 1]]

### ****9. N-Queens II****

**Problem Statement:**  
This is a variation of the N-Queens problem where the task is to find the total number of distinct solutions for placing n queens on an n x n chessboard such that no two queens threaten each other.

**Input:**  
An integer n representing the size of the chessboard and the number of queens.

**Output:**  
The total number of distinct solutions.

**Example:**  
Input: n = 4  
Output: 2

### ****10. Combination Sum II****

**Problem Statement:**  
Given a collection of candidate numbers (which may include duplicates) and a target number, find all unique combinations that sum to the target. Each number in the candidates may only be used once in the combination.

**Input:**  
An array candidates[] and an integer target.

**Output:**  
All unique combinations that sum to target.

**Example:**  
Input:  
candidates = [10, 1, 2, 7, 6, 5], target = 8  
Output:  
[[1, 2, 5], [1, 7], [2, 6]]

### ****11. Solve the Sudoku Puzzle (Backtracking)****

**Problem Statement:**  
Given a partially filled Sudoku board, solve it using backtracking by filling in the missing values.

**Input:**  
A partially filled 9x9 Sudoku board.

**Output:**  
A solved Sudoku board.

**Example:**  
Input:  
A partially filled board.  
Output:  
A completed Sudoku board.

### ****12. All Paths from Source to Destination in a Directed Graph****

**Problem Statement:**  
Given a directed graph and two nodes, find all possible paths from the source node to the destination node.

**Input:**  
A directed graph and two nodes: source and destination.

**Output:**  
All possible paths from source to destination.

**Example:**  
Input:  
Graph: [[1, 2], [3], [3], []], source: 0, destination: 3  
Output:  
[[0, 1, 3], [0, 2, 3]]

### ****13. Sudoku Solver (Backtracking)****

**Problem Statement:**  
Write a program to solve a partially filled Sudoku board using backtracking.

**Input:**  
A partially filled 9x9 Sudoku grid.

**Output:**  
A solved Sudoku board.

**Example:**  
Input:  
A partially filled Sudoku board.  
Output:  
A completed Sudoku grid.

### ****14. Find All Valid IP Addresses****

**Problem Statement:**  
Given a string, find all valid IP addresses that can be formed by inserting three dots into the string.

**Input:**  
A string s representing the digits of an IP address.

**Output:**  
All valid IP addresses formed by inserting three dots.

**Example:**  
Input: "25525511135"  
Output:  
["255.255.11.135", "255.255.111.35"]