Top 12 Topics of DSA and Classical Questions Related to it with source.

Array Manipulation:

- **Max Subarray Sum**: Find the contiguous subarray within an array (containing at least one number) that has the largest sum. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Two Sum**: Given an array of integers, return indices of the two numbers such that they add up to a specific target. (Source: LeetCode)
- **Merge Intervals**: Given a collection of intervals, merge all overlapping intervals. (Source: LeetCode)
- **Container With Most Water**: 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. (Source: LeetCode)
- **Product of Array Except Self**: Given an array nums of n integers where n > 1, return an array output such that output[i] is equal to the product of all the elements of nums except nums[i]. (Source: LeetCode)
- **Rotate Array**: Rotate an array to the right by k steps, where k is non-negative. (Source: LeetCode)
- **Jump Game**: Given an array of non-negative integers, you are initially positioned at the first index of the array. Each element in the array represents your maximum jump length at that position. Determine if you can reach the last index. (Source: LeetCode)
- **Median of Two Sorted Arrays**: Given two sorted arrays nums1 and nums2 of size m and n respectively, return the median of the two sorted arrays. (Source: LeetCode)
- **Next Permutation**: Implement next permutation, which rearranges numbers into the lexicographically next greater permutation of numbers. (Source: LeetCode)
- **Set Matrix Zeroes**: Given an m x n matrix, if an element is 0, set its entire row and column to 0. Do it in-place. (Source: LeetCode)

Linked List Operations:

- **Reverse Linked List**: Reverse a singly linked list. (Source: "Cracking the Coding Interview" by Gayle Laakmann McDowell)
- **Detect Cycle in a Linked List**: Given a linked list, return the node where the cycle begins. If there is no cycle, return null. (Source: LeetCode)
- **Merge Two Sorted Lists**: Merge two sorted linked lists and return it as a new sorted list. The new list should be made by splicing together the nodes of the first two lists. (Source: LeetCode)
- **Intersection of Two Linked Lists**: Write a program to find the node at which the intersection of two singly linked lists begins. (Source: LeetCode)
- **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. (Source: LeetCode)
- **Palindrome Linked List**: Given a singly linked list, determine if it is a palindrome. (Source: LeetCode)
- **Copy List with Random Pointer**: 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. Return a deep copy of the list. (Source: LeetCode)
- **Linked List Cycle II**: Given a linked list, return the node where the cycle begins. If there is no cycle, return null. (Source: LeetCode)
- **Add Two Numbers**: You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list. (Source: LeetCode)
- **Flatten a Multilevel Doubly Linked List**: Flatten a multilevel doubly linked list. (Source: LeetCode)

Binary Search Trees (BST):

- **Validate BST**: Given the root of a binary tree, determine if it is a valid binary search tree (BST). (Source: LeetCode)
- **Inorder Traversal**: Given the root of a binary tree, return the inorder traversal of its nodes' values. (Source: LeetCode)
- **Kth Smallest Element in a BST**: Given the root of a binary search tree, return the kth smallest element in it. (Source: LeetCode)

- **Lowest Common Ancestor of a Binary Tree**: Given a binary tree, find the lowest common ancestor (LCA) of two given nodes in the tree. (Source: LeetCode)
- **Binary Tree Maximum Path Sum**: Given a non-empty binary tree, find the maximum path sum. (Source: LeetCode)
- **Construct Binary Tree from Preorder and Inorder Traversal**: Given preorder and inorder traversal of a tree, construct the binary tree. (Source: LeetCode)
- **Binary Tree Level Order Traversal**: Given a binary tree, return the level order traversal of its nodes' values. (Source: LeetCode)
- **Binary Tree Zigzag Level Order Traversal**: Given a binary tree, return the zigzag level order traversal of its nodes' values. (Source: LeetCode)
- **Balanced Binary Tree**: Given a binary tree, determine if it is height-balanced. (Source: LeetCode)
- **Binary Tree Paths**: Given a binary tree, return all root-to-leaf paths. (Source: LeetCode)

Sorting Algorithms:

- **Merge Sort**: Implement the merge sort algorithm. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Quick Sort**: Implement the quick sort algorithm. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Heap Sort**: Implement the heap sort algorithm. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Bubble Sort**: Implement the bubble sort algorithm. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Insertion Sort**: Implement the insertion sort algorithm. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Selection Sort**: Implement the selection sort algorithm. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Counting Sort**: Implement the counting sort algorithm. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Radix Sort**: Implement the radix sort algorithm. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)

- **Bucket Sort**: Implement the bucket sort algorithm. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Topological Sort**: Implement the topological sort algorithm. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)

Graph Traversal:

- **Depth-First Search (DFS)**: Implement DFS traversal of a graph. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Breadth-First Search (BFS)**: Implement BFS traversal of a graph. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Dijkstra's Algorithm**: Implement Dijkstra's shortest path algorithm. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Bellman-Ford Algorithm**: Implement the Bellman-Ford shortest path algorithm. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Prim's Algorithm**: Implement Prim's minimum spanning tree algorithm. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Kruskal's Algorithm**: Implement Kruskal's minimum spanning tree algorithm. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Floyd-Warshall Algorithm**: Implement the Floyd-Warshall all-pairs shortest path algorithm. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Articulation Points (Cut Vertices)**: Find all articulation points in a given undirected graph. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Bridges in a Graph**: Find all bridges in a given undirected graph. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Topological Sorting**: Perform topological sorting of a directed acyclic graph (DAG). (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)

Dynamic Programming:

- **Fibonacci Series**: Find the nth Fibonacci number using dynamic programming. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Longest Common Subsequence (LCS)**: Find the length of the longest common subsequence of two given strings. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)

- **Knapsack Problem**: Solve the 0-1 knapsack problem using dynamic programming. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Coin Change Problem**: Find the number of ways to make change for a given amount using a given set of coins. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Longest Increasing Subsequence (LIS)**: Find the length of the longest increasing subsequence of a given sequence. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Matrix Chain Multiplication**: Find the optimal parenthesization of a matrix chain multiplication using dynamic programming. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Edit Distance**: Find the minimum number of operations required to convert one string into another using dynamic programming. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Maximum Subarray Sum**: Find the contiguous subarray within an array (containing at least one number) that has the largest sum using dynamic programming. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Rod Cutting Problem**: Find the maximum revenue by cutting a rod of length n into smaller rods. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Subset Sum Problem**: Determine whether there exists a subset of the given set with a given sum using dynamic programming. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)

String Manipulation:

- **Anagram Detection**: Check if two strings are anagrams of each other. (Source: "Cracking the Coding Interview" by Gayle Laakmann McDowell)
- **Longest Palindromic Substring**: Find the longest palindromic substring in a given string. (Source: LeetCode)
- **String Compression**: Compress a string by replacing consecutive repeated characters with their counts. (Source: LeetCode)
- **Valid Parentheses**: Determine if a given string of parentheses is valid. (Source: LeetCode)
- **String to Integer (atoi)**: Convert a string to an integer. (Source: LeetCode)

- **Reverse Words in a String**: Reverse the order of words in a given string. (Source: LeetCode)
- **Longest Substring Without Repeating Characters**: Find the length of the longest substring without repeating characters. (Source: LeetCode)
- **Regular Expression Matching**: Implement regular expression matching with support for '.' and '*'. (Source: LeetCode)
- **Group Anagrams**: Given an array of strings, group anagrams together. (Source: LeetCode)
- **Implement strStr()**: Return the index of the first occurrence of needle in haystack, or -1 if needle is not part of haystack. (Source: LeetCode)

Stacks and Queues:

- **Implement Stack using Queues**: Implement a stack using queues. (Source: "Cracking the Coding Interview" by Gayle Laakmann McDowell)
- **Implement Queue using Stacks**: Implement a queue using stacks. (Source: "Cracking the Coding Interview" by Gayle Laakmann McDowell)
- **Valid Parentheses**: Determine if a given string of parentheses is valid using a stack. (Source: LeetCode)
- **Largest Rectangle in Histogram**: Given n non-negative integers representing the histogram's bar height where the width of each bar is 1, find the area of the largest rectangle in the histogram. (Source: LeetCode)
- **Min Stack**: Design a stack that supports push, pop, top, and retrieving the minimum element in constant time. (Source: LeetCode)
- **Sliding Window Maximum**: Given an array 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. (Source: LeetCode)
- **Evaluate Reverse Polish Notation**: Evaluate the value of an arithmetic expression in Reverse Polish Notation. (Source: LeetCode)
- **Decode String**: Given an encoded string, return its decoded string. (Source: LeetCode)
- **Implement Stack using Linked List**: Implement a stack using a linked list. (Source: GeeksforGeeks)

- **Implement Queue using Linked List**: Implement a queue using a linked list. (Source: GeeksforGeeks)

Hashing:

- **Two Sum**: Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target. (Source: LeetCode)
- **Longest Substring Without Repeating Characters**: Given a string s, find the length of the longest substring without repeating characters. (Source: LeetCode)
- **Group Anagrams**: Given an array of strings, group anagrams together. (Source: LeetCode)
- **Valid Anagram**: Given two strings s and t, return true if t is an anagram of s, and false otherwise. (Source: LeetCode)
- **First Unique Character in a String**: Given a string s, find the first non-repeating character in it and return its index. If it does not exist, return -1. (Source: LeetCode)
- **Find All Anagrams in a String**: Given two strings s and p, return an array of all the start indices of p's anagrams in s. (Source: LeetCode)
- **Minimum Index Sum of Two Lists**: Suppose Andy and Doris want to choose a restaurant for dinner and they both have a list of favorite restaurants represented by strings. You need to help them find out their common interest with the least list index sum. (Source: LeetCode)
- **Isomorphic Strings**: Given two strings s and t, determine if they are isomorphic. Two strings are isomorphic if the characters in s can be replaced to get t. (Source: LeetCode)
- **Find the Difference**: Given two strings s and t which consist of only lowercase letters, return the letter that was added to t. (Source: LeetCode)
- **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. (Source: LeetCode)

Recursion:

- **Fibonacci Series**: Generate the nth Fibonacci number using recursion. (Source: "Cracking the Coding Interview" by Gayle Laakmann McDowell)
- **Factorial**: Compute the factorial of a given number using recursion. (Source: GeeksforGeeks)
- **Tower of Hanoi**: Solve the Tower of Hanoi problem using recursion. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)

- **Subset Generation**: Generate all subsets of a set using recursion. (Source: "Introduction to Algorithms" by Thomas H. Cormen et al.)
- **Permutations of a String**: Generate all permutations of a string using recursion. (Source: "Cracking the Coding Interview" by Gayle Laakmann McDowell)
- **Combination Sum**: 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. (Source: LeetCode)
- **Generate Parentheses**: Given n pairs of parentheses, write a function to generate all combinations of well-formed parentheses. (Source: LeetCode)
- **Subsets**: Given an integer array nums that may contain duplicates, return all possible subsets (the power set). (Source: LeetCode)
- **Palindrome Partitioning**: Given a string s, partition s such that every substring of the partition is a palindrome. Return all possible palindrome partitioning of s. (Source: LeetCode)
- **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. (Source: LeetCode)

These questions cover a wide range of topics commonly asked in technical interviews at tech companies and provide a solid foundation for DSA interview preparation. You can find solutions and explanations for these questions in various online coding platforms, programming books, and resources such as LeetCode, GeeksforGeeks, HackerRank, and "Cracking the Coding Interview" by Gayle Laakmann McDowell.

Bit Manipulation

- **Bitwise AND of Numbers Range**:
    - Given two integers left and right that represent the range [left, right], return the bitwise AND of all numbers in this range, inclusive. (Source: LeetCode)
- **Number of 1 Bits**:
    - Write a function that takes an unsigned integer and returns the number of '1' bits it has (also known as the Hamming weight). (Source: LeetCode)
- **Single Number**:

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

- **Reverse Bits**:
    - Reverse bits of a given 32 bits unsigned integer. (Source: LeetCode)
- **Counting Bits**:
    - Given a non-negative integer num, for every number i in the range 0 ≤ i ≤ num, calculate the number of 1's in their binary representation and return them as an array. (Source: LeetCode)
- **Bitwise ORs of Subarrays**:
    - We have an array arr of non-negative integers. For every (contiguous) subarray sub = [arr[i], arr[i + 1], ..., arr[j]] (with i <= j), we take the bitwise OR of all the elements in sub, obtaining a result arr[i] | arr[i + 1] | ... | arr[j]. Return the number of possible results. (Source: LeetCode)
- **Maximum XOR of Two Numbers in an Array**:
    - Given an integer array nums, return the maximum result of nums[i] XOR nums[j], where 0 ≤ i ≤ j < n. (Source: LeetCode)
- **Bitwise AND of Numbers Range**:
    - Given two integers left and right that represent the range [left, right], return the bitwise AND of all numbers in this range, inclusive. (Source: LeetCode)
- **Sum of Two Integers**:
    - Calculate the sum of two integers a and b, but you are not allowed to use the operators + and -. (Source: LeetCode)
- **Binary Watch**:
    - A binary watch has 4 LEDs on the top which represent the hours (0-11), and 6 LEDs on the bottom which represent the minutes (0-59). Each LED represents a zero or one, with the least significant bit on the right. Given a non-negative integer n which represents the number of LEDs that are currently on, return all possible times the watch could represent. (Source: LeetCode)

These questions cover a variety of Bit Manipulation topics and are commonly asked in technical interviews at tech companies. You can practice these questions on LeetCode or other coding platforms to strengthen your skills in Bit Manipulation.

Matrix

- **Search a 2D Matrix**:
    - Write an efficient algorithm that searches for a value in an m x n matrix. This matrix has the following properties: Integers in each row are sorted from left to right. The first integer of each row is greater than the last integer of the previous row. (Source: LeetCode)
- **Set Matrix Zeroes**:
    - Given an m x n matrix, if an element is 0, set its entire row and column to 0. Do it in-place. (Source: LeetCode)
- **Rotate Image**:
    - You are given an n x n 2D matrix representing an image, rotate the image by 90 degrees (clockwise). (Source: LeetCode)
- **Spiral Matrix**:
    - Given an m x n matrix, return all elements of the matrix in spiral order. (Source: LeetCode)
- **Word Search**:
    - 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. (Source: LeetCode)
- **Maximal Square**:
    - Given an m x n binary matrix filled with 0's and 1's, find the largest square containing only 1's and return its area. (Source: LeetCode)
- **Matrix Diagonal Sum**:
    - Given a square matrix mat, return the sum of the matrix diagonals. Only include the sum of all the elements on the primary diagonal and all the elements on the secondary diagonal that are not part of the primary diagonal. (Source: LeetCode)
- **Count Negative Numbers in a Sorted Matrix**:

    - Given a m x n matrix grid which is sorted in non-increasing order both row-wise and column-wise, return the number of negative numbers in grid. (Source: LeetCode)

- **Pacific Atlantic Water Flow**:
    - Given an m x n matrix of non-negative integers representing the height of each unit cell in a continent, the "Pacific ocean" touches the left and top edges of the matrix, and the "Atlantic ocean" touches the right and bottom edges. Water can only flow in four directions (up, down, left, or right) from a cell to another one with height equal or lower. Find the list of grid coordinates where water can flow to both the Pacific and Atlantic ocean. (Source: LeetCode)

- **Reshape the Matrix**:
    - In MATLAB, there is a handy function called reshape which can reshape an m x n matrix into a new one with a different size r x c while keeping its original data. You are given an m x n matrix mat and two integers r and c representing the row number and column number of the wanted reshaped matrix. (Source: LeetCode)