50 DSA Problems to Secure a 7-15 LPA Job

Table of Contents

1. Introduction

This document outlines 50 carefully selected Data Structures and Algorithms (DSA) problems that are essential for cracking interviews at top companies offering salaries in the range of 7-15 LPA. These problems cover various topics, ensuring a comprehensive preparation.

2. Array Problems

1. Find the Largest Sum Contiguous Subarray (Kadane's Algorithm)
    - Example: Input: [-2, 1, -3, 4, -1, 2, 1, -5, 4] Output: 6 (Subarray: [4, -1, 2, 1])
2. Rotate an Array by K Steps
    - Example: Input: nums = [1,2,3,4,5,6,7], k = 3 Output: [5,6,7,1,2,3,4]
3. Merge Intervals
    - Example: Input: [[1,3],[2,6],[8,10],[15,18]] Output: [[1,6],[8,10],[15,18]]
4. Find the Duplicate Number
    - Example: Input: [1,3,4,2,2] Output: 2
5. Maximum Product Subarray
    - Example: Input: [2,3,-2,4] Output: 6
6. Find the Missing and Repeating Number
    - Example: Input: n = 5, arr[] = {1, 3, 3, 5, 4} Output: Missing = 2, Repeating = 3
7. Subarray with Given Sum
    - Example: Input: arr = [1,2,3,7,5], sum = 12 Output: [2,4]
8. Longest Consecutive Sequence
    - Example: Input: [100,4,200,1,3,2] Output: 4
9. Trapping Rain Water
    - Example: Input: [0,1,0,2,1,0,1,3,2,1,2,1] Output: 6
10. Next Permutation
    - Example: Input: [1,2,3] Output: [1,3,2]

3. String Problems

11. Longest Palindromic Substring
    - Example: Input: "babad" Output: "bab" or "aba"
12. Reverse Words in a String
    - Example: Input: "the sky is blue" Output: "blue is sky the"
13. Longest Common Prefix
    - Example: Input: ["flower","flow","flight"] Output: "fl"
14. Group Anagrams
    - Example: Input: ["eat","tea","tan","ate","nat","bat"] Output: [["bat"],["nat","tan"],["ate","eat","tea"]]
15. Check for Valid Parentheses
    - Example: Input: "()[]{}" Output: true
16. Implement ATOI
    - Example: Input: "42" Output: 42
17. String to Integer Conversion
    - Example: Input: "-123" Output: -123
18. Longest Repeating Subsequence
    - Example: Input: "AABEBCDD" Output: "ABD"
19. KMP Algorithm for Pattern Searching
    - Example: Text: "abxabcabcaby", Pattern: "abcaby" Output: 6
20. Minimum Window Substring
    - Example: Input: s = "ADOBECODEBANC", t = "ABC" Output: "BANC"

4. Linked List Problems

21. Reverse a Linked List
    - Example: Input: 1 -> 2 -> 3 -> 4 -> 5 Output: 5 -> 4 -> 3 -> 2 -> 1
22. Detect and Remove a Loop in a Linked List
    - Example: Input: Linked List with a loop Output: Loop removed
23. Merge Two Sorted Linked Lists
    - Example: Input: 1 -> 2 -> 4, 1 -> 3 -> 4 Output: 1 -> 1 -> 2 -> 3 -> 4 -> 4
24. Flatten a Multilevel Doubly Linked List
    - Example: Input: Nested Linked List Output: Single Flattened List
25. Find the Intersection Point of Two Linked Lists
    - Example: Input: List A: 4 -> 1 -> 8 -> 4 -> 5, List B: 5 -> 0 -> 1 -> 8 -> 4 -> 5 Output: 8
26. Remove N-th Node from the End of the List
    - Example: Input: 1 -> 2 -> 3 -> 4 -> 5, n = 2 Output: 1 -> 2 -> 3 -> 5
27. Add Two Numbers Represented by Linked Lists
    - Example: Input: 7 -> 5 -> 9, 8 -> 4 Output: 5 -> 0 -> 0 -> 1
28. Clone a Linked List with Random Pointers
    - Example: Input: Original List with random pointers Output: Cloned List
29. Sort a Linked List
    - Example: Input: 4 -> 2 -> 1 -> 3 Output: 1 -> 2 -> 3 -> 4 30.Check if a

Linked List is Palindrome

- Example: Input: 1 -> 2 -> 2 -> 1 Output: true

5. Stack and Queue Problems

31. Implement Stack Using Queues
    - Example: Operations: Push, Pop, Top Output: Mimic Stack behavior
32. Implement Queue Using Stacks
    - Example: Operations: Enqueue, Dequeue Output: Mimic Queue behavior
33. Next Greater Element
    - Example: Input: [4,5,2,25] Output: [5,25,25,-1]
34. LRU Cache Implementation
    - Example: Operations: Set, Get Output: Cache results
35. Min Stack
    - Example: Operations: Push, Pop, Top, GetMin Output: Min value of stack
36. Evaluate Reverse Polish Notation
    - Example: Input: ["2","1","+","3","*"] Output: 9
37. Circular Queue Implementation
    - Example: Operations on Circular Queue Output: Maintain FIFO order
38. Sliding Window Maximum
    - Example: Input: nums = [1,3,-1,-3,5,3,6,7], k = 3 Output: [3,3,5,5,6,7]
39. Celebrity Problem
    - Example: Input: Matrix representing acquaintances Output: Celebrity index

40. Largest Rectangle in Histogram

    - Example: Input: [2,1,5,6,2,3] Output: 10

---

6. Binary Tree and Binary Search Tree Problems

41. Inorder, Preorder, Postorder Traversals
    - Example: Input: Binary Tree Output: Various traversal orders
42. Level Order Traversal
    - Example: Input: Binary Tree Output: Level order traversal as a list
43. Diameter of a Binary Tree
    - Example: Input: Binary Tree Output: Diameter of the tree
44. Lowest Common Ancestor in a Binary Tree
    - Example: Input: Binary Tree, Two Nodes Output: Lowest common ancestor
45. Validate a Binary Search Tree
    - Example: Input: Binary Tree Output: True if it's a BST
46. Serialize and Deserialize a Binary Tree
    - Example: Input: Binary Tree Output: Serialized and Deserialized tree
47. Zigzag Level Order Traversal
    - Example: Input: Binary Tree Output: Zigzag traversal order
48. Kth Smallest Element in a BST
    - Example: Input: BST, k = 3 Output: 3rd smallest element
49. Maximum Path Sum in a Binary Tree
    - Example: Input: Binary Tree Output: Maximum path sum
50. Construct a Binary Tree from Preorder and Inorder Traversal
    - Example: Input: Preorder and Inorder arrays Output: Constructed Binary Tree

7. Graph Problems

1. Breadth-First Search (BFS)
   - Example: Input: Graph and a starting node Output: BFS traversal
2. Depth-First Search (DFS)
   - Example: Input: Graph and a starting node Output: DFS traversal
3. Detect Cycle in a Directed Graph
   - Example: Input: Directed graph Output: True/False if cycle exists
4. Detect Cycle in an Undirected Graph
   - Example: Input: Undirected graph Output: True/False if cycle exists
5. Dijkstra’s Shortest Path Algorithm
   - Example: Input: Graph, source node Output: Shortest distances from source

8. Dynamic Programming Problems

1. 0/1 Knapsack Problem
    - Example: Input: Weights, Values, Capacity Output: Maximum value possible
2. Longest Increasing Subsequence
    - Example: Input: [10,9,2,5,3,7,101,18] Output: 4
3. Longest Common Subsequence
    - Example: Input: "abcde", "ace" Output: "ace"
4. Edit Distance
    - Example: Input: Words "horse", "ros" Output: 3
5. Partition Equal Subset Sum
    - Example: Input: [1,5,11,5] Output: true (Partition exists)

9. Searching and Sorting Problems

1. *Binary Search*
  - *Example*: Input: arr = [1, 3, 5, 7, 9], target = 5 Output: 2 (Index of the target)

2. *Merge Sort*
  - *Example*: Input: [5, 2, 9, 1, 5, 6] Output: [1, 2, 5, 5, 6, 9]

3. *Quick Sort*
  - *Example*: Input: [10, 7, 8, 9, 1, 5] Output: [1, 5, 7, 8, 9, 10]

4. *Find First and Last Position of an Element in a Sorted Array*
  - *Example*: Input: nums = [5,7,7,8,8,10], target = 8 Output: [3,4]

5. *Kth Smallest Element*
  - *Example*: Input: arr = [7, 10, 4, 3, 20, 15], k = 3 Output: 7

6. *Search in Rotated Sorted Array*
  - *Example*: Input: nums = [4,5,6,7,0,1,2], target = 0 Output: 4

7. *Count Inversions in an Array*
   - *Example*: Input: [8, 4, 2, 1] Output: 6

8. *Heap Sort*
   - *Example*: Input: [12, 11, 13, 5, 6, 7] Output: [5, 6, 7, 11, 12, 13]

9. *Counting Sort*
   - *Example*: Input: [4, 2, 2, 8, 3, 3, 1] Output: [1, 2, 2, 3, 3, 4, 8]

10. *Radix Sort*
   - *Example*: Input: [170, 45, 75, 90, 802, 24, 2, 66] Output: [2, 24, 45, 66, 75, 90, 170, 802]

10. Backtracking Problems

1. *N-Queens Problem*
   - *Example*: Input: n = 4 Output: All arrangements of 4 queens on a 4x4 chessboard.

2. *Sudoku Solver*
   - *Example*: Input: Partially filled 9x9 board Output: Completed board.

3. *Word Search*
   - *Example*: Input: board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "ABCCED" Output: true

4. *Permutations of a String*
   - *Example*: Input: "ABC" Output: ["ABC", "ACB", "BAC", "BCA", "CAB", "CBA"]

5. *Subsets*
   - *Example*: Input: nums = [1,2,3] Output: [[],[1],[2],[1,2],[3],[1,3],[2,3],[1,2,3]]

6. *Combination Sum*
   - *Example*: Input: candidates = [2,3,6,7], target = 7 Output: [[7],[2,2,3]]

7. *Rat in a Maze*
   - *Example*: Input: A maze grid Output: All possible paths from start to finish.

8. *Palindrome Partitioning*
   - *Example*: Input: "aab" Output: [["a","a","b"],["aa","b"]]

9. *Knight's Tour Problem*
   - *Example*: Input: n = 8 Output: Sequence of moves for a knight to visit all cells of an 8x8 board exactly once.

10. *Solve the M-Coloring Problem*
   - *Example*: Input: Graph and m colors Output: Possible coloring of graph nodes.

11. Greedy Algorithm Problems

1. *Activity Selection Problem*
  - *Example*: Input: Start times = [1, 3, 0, 5, 8, 5], End times = [2, 4, 6, 7, 9, 9] Output: Maximum number of non-overlapping activities.

2. *Fractional Knapsack Problem*
  - *Example*: Input: Weights and values of items, capacity Output: Maximum value in the knapsack.

3. *Huffman Encoding*
  - *Example*: Input: Characters and frequencies Output: Huffman tree and codes.

4. *Minimum Spanning Tree (Prim's Algorithm)*
  - *Example*: Input: Graph Output: MST and its weight.

5. *Minimum Spanning Tree (Kruskal's Algorithm)*
  - *Example*: Input: Graph Output: MST and its weight.

6. *Job Sequencing Problem*
  - *Example*: Input: Jobs with deadlines and profits Output: Maximum profit sequence of jobs.

7. *Greedy Coloring of a Graph*
  - *Example*: Input: Graph Output: Minimum number of colors needed to color the graph.

8. *Optimal File Merge Pattern*
  - *Example*: Input: File sizes [20, 30, 10, 5] Output: Minimum cost to merge files.

9. *Dijkstra's Shortest Path Algorithm*
  - *Example*: Input: Graph, source Output: Shortest paths to all nodes.

10. *Gas Station Problem*
  - *Example*: Input: Gas = [1,2,3,4,5], Cost = [3,4,5,1,2] Output: 3 (Index to start the circuit).

Conclusion

This compilation of problems spans all major DSA topics, providing a solid foundation for cracking interviews at top tech companies. Each problem is carefully chosen to help you understand key concepts and apply them effectively. With consistent practice and a deep understanding of these problems, you'll be well-equipped to tackle any coding challenge.