Blind 75 LeetCode - Complete DSA Interview Preparation Guide

Can You Complete Blind 75 in 2 Days?

Realistic Assessment: Completing all 75 problems with proper understanding in 2 days is **extremely challenging** but here's a strategic approach:

Day 1 Focus (12-14 hours)

- **Arrays & Strings** (25 problems) - 6-8 hours
- **Linked Lists** (6 problems) - 2 hours
- **Trees** (14 problems) - 4 hours

Day 2 Focus (12-14 hours)

- **Dynamic Programming** (11 problems) - 6 hours
- **Graphs** (8 problems) - 4 hours
- **Remaining categories** - 4 hours

⚡ 2-Day Strategy

1. **Focus on patterns, not memorization**
2. **Study optimized solutions directly**
3. **Practice coding 2-3 key problems per pattern**
4. **Review Amazon-specific frequent problems**

Complete Blind 75 Problem List by Category

Array (10 Problems)

1. **Two Sum** (Easy) - Hash Map Pattern
2. **Best Time to Buy and Sell Stock** (Easy) - Greedy Pattern
3. **Contains Duplicate** (Easy) - Hash Set Pattern
4. **Product of Array Except Self** (Medium) - Prefix Sum Pattern
5. **Maximum Subarray** (Medium) - Kadane's Algorithm
6. **Maximum Product Subarray** (Medium) - DP Pattern
7. **Find Minimum in Rotated Sorted Array** (Medium) - Binary Search

8. **Search in Rotated Sorted Array** (Medium) - Binary Search
9. **3Sum** (Medium) - Two Pointers
10. **Container With Most Water** (Medium) - Two Pointers

String (10 Problems)

11. **Longest Substring Without Repeating Characters** (Medium) - Sliding Window
12. **Longest Repeating Character Replacement** (Medium) - Sliding Window
13. **Minimum Window Substring** (Hard) - Sliding Window
14. **Valid Anagram** (Easy) - Hash Map
15. **Group Anagrams** (Medium) - Hash Map
16. **Valid Parentheses** (Easy) - Stack
17. **Valid Palindrome** (Easy) - Two Pointers
18. **Longest Palindromic Substring** (Medium) - DP/Expand Around Centers
19. **Palindromic Substrings** (Medium) - DP
20. **Encode and Decode Strings** (Medium) - String Manipulation

Linked List (6 Problems)

21. **Reverse Linked List** (Easy) - Two Pointers
22. **Detect Cycle in a Linked List** (Easy) - Floyd's Algorithm
23. **Merge Two Sorted Lists** (Easy) - Two Pointers
24. **Merge k Sorted Lists** (Hard) - Divide & Conquer/Heap
25. **Remove Nth Node From End of List** (Medium) - Two Pointers
26. **Reorder List** (Medium) - Two Pointers + Reverse

Tree (14 Problems)

27. **Maximum Depth of Binary Tree** (Easy) - DFS/BFS
28. **Same Tree** (Easy) - DFS
29. **Invert/Flip Binary Tree** (Easy) - DFS
30. **Binary Tree Maximum Path Sum** (Hard) - DFS
31. **Binary Tree Level Order Traversal** (Medium) - BFS
32. **Serialize and Deserialize Binary Tree** (Hard) - DFS/BFS
33. **Subtree of Another Tree** (Easy) - DFS
34. **Construct Binary Tree from Preorder and Inorder** (Medium) - DFS
35. **Validate Binary Search Tree** (Medium) - DFS
36. **Kth Smallest Element in a BST** (Medium) - DFS/Inorder

37. **Lowest Common Ancestor of BST** (Easy) - BST Properties
38. **Implement Trie (Prefix Tree)** (Medium) - Trie
39. **Add and Search Word** (Medium) - Trie + DFS
40. **Word Search II** (Hard) - Trie + DFS

Dynamic Programming (11 Problems)

41. **Climbing Stairs** (Easy) - Basic DP
42. **Coin Change** (Medium) - DP
43. **Longest Increasing Subsequence** (Medium) - DP
44. **Longest Common Subsequence** (Medium) - DP
45. **Word Break Problem** (Medium) - DP
46. **Combination Sum** (Medium) - Backtracking/DP
47. **House Robber** (Medium) - DP
48. **House Robber II** (Medium) - DP
49. **Decode Ways** (Medium) - DP
50. **Unique Paths** (Medium) - DP
51. **Jump Game** (Medium) - Greedy/DP

Graph (8 Problems)

52. **Clone Graph** (Medium) - DFS/BFS
53. **Course Schedule** (Medium) - Topological Sort
54. **Pacific Atlantic Water Flow** (Medium) - DFS
55. **Number of Islands** (Medium) - DFS/BFS
56. **Longest Consecutive Sequence** (Medium) - Union Find/Hash Set
57. **Alien Dictionary** (Hard) - Topological Sort
58. **Graph Valid Tree** (Medium) - DFS/Union Find
59. **Number of Connected Components** (Medium) - DFS/Union Find

Interval (5 Problems)

60. **Insert Interval** (Medium) - Merge Intervals
61. **Merge Intervals** (Medium) - Sorting
62. **Non-overlapping Intervals** (Medium) - Greedy
63. **Meeting Rooms** (Easy) - Sorting
64. **Meeting Rooms II** (Medium) - Heap/Sorting

Heap (3 Problems)

65. **Merge k Sorted Lists** (Hard) - Min Heap
66. **Top K Frequent Elements** (Medium) - Heap
67. **Find Median from Data Stream** (Hard) - Two Heaps

Matrix (4 Problems)

68. **Set Matrix Zeroes** (Medium) - Matrix Manipulation
69. **Spiral Matrix** (Medium) - Matrix Traversal
70. **Rotate Image** (Medium) - Matrix Rotation
71. **Word Search** (Medium) - DFS/Backtracking

Binary (5 Problems)

72. **Sum of Two Integers** (Medium) - Bit Manipulation
73. **Number of 1 Bits** (Easy) - Bit Manipulation
74. **Counting Bits** (Easy) - Bit Manipulation/DP
75. **Missing Number** (Easy) - Bit Manipulation/Math

Top 20 Most Important Problems (Amazon Focused)

Must-Know for Amazon Interviews:

1. **Two Sum** - Foundation hash map problem
2. **Maximum Subarray (Kadane's)** - Classic DP/Greedy
3. **Merge Intervals** - Common Amazon pattern
4. **Number of Islands** - Graph traversal fundamentals
5. **Reverse Linked List** - Linked list basics
6. **Valid Parentheses** - Stack fundamentals
7. **Binary Tree Level Order Traversal** - BFS template
8. **Climbing Stairs** - DP introduction
9. **Best Time to Buy and Sell Stock** - Greedy pattern
10. **3Sum** - Two pointers technique
11. **Longest Substring Without Repeating** - Sliding window
12. **Course Schedule** - Topological sort
13. **Word Search** - DFS/Backtracking
14. **Top K Frequent Elements** - Heap usage
15. **Validate BST** - Tree property validation

16. **House Robber** - Classic DP pattern
17. **Coin Change** - DP optimization
18. **Clone Graph** - Graph algorithms
19. **Merge k Sorted Lists** - Divide & conquer
20. **Serialize/Deserialize Binary Tree** - Tree serialization

Key Patterns & Templates

Pattern 1: Two Pointers

```
def two_pointers(arr):
    left, right = 0, len(arr) - 1
    while left < right:
        if condition:
            # Process and move pointers
            left += 1
        else:
            right -= 1
```

Pattern 2: Sliding Window

```
def sliding_window(s):
    left = 0
    window = {}
    max_len = 0

    for right in range(len(s)):
        # Expand window
        window[s[right]] = window.get(s[right], 0) + 1

        # Contract window if needed
        while condition_violated:
            window[s[left]] -= 1
            if window[s[left]] == 0:
                del window[s[left]]
            left += 1

        max_len = max(max_len, right - left + 1)

    return max_len
```

Pattern 3: DFS Template

```
def dfs(root):
    if not root:
        return
```

```
    # Process current node

    # Recurse on children
    dfs(root.left)
    dfs(root.right)
```

Pattern 4: BFS Template

```
def bfs(root):
    if not root:
        return

    queue = [root]
    while queue:
        node = queue.pop(0)
        # Process node

        if node.left:
            queue.append(node.left)
        if node.right:
            queue.append(node.right)
```

Pattern 5: Dynamic Programming

```
def dp_solution(n):
    # Base cases
    if n <= 1:
        return n

    # DP array
    dp = [0] * (n + 1)
    dp[0], dp[1] = 0, 1

    # Fill DP table
    for i in range(2, n + 1):
        dp[i] = dp[i-1] + dp[i-2]  # Recurrence relation

    return dp[n]
```

Amazon-Specific Interview Tips

What Amazon Tests:

1. **Problem-solving approach** - Think out loud
2. **Code quality** - Clean, readable code
3. **Edge cases** - Handle null, empty inputs
4. **Optimization** - Time/space complexity analysis

5. **Testing** - Walk through examples

Amazon Leadership Principles in Coding:

- **Ownership** - Take responsibility for your solution
- **Dive Deep** - Understand the problem thoroughly
- **Think Big** - Consider scalability
- **Bias for Action** - Start coding after clarifying
- **Customer Obsession** - Consider user experience

Time Management (45-60 min interview):

- **5 min** - Problem clarification & examples
- **10 min** - Approach discussion (brute force → optimized)
- **20 min** - Coding the solution
- **10 min** - Testing & edge cases
- **5 min** - Complexity analysis & improvements

⚡ 2-Day Crash Course Schedule

Day 1 (14 hours total)

Morning (6 hours): Arrays & Strings

- 7:00-9:00 AM: Array fundamentals (Two Sum, Best Time to Buy/Sell, Contains Duplicate)
- 9:00-11:00 AM: Advanced arrays (Product Except Self, Maximum Subarray, 3Sum)
- 11:00-1:00 PM: String patterns (Valid Anagram, Valid Parentheses, Sliding Window problems)

Afternoon (4 hours): Linked Lists

- 2:00-4:00 PM: Basic operations (Reverse, Cycle Detection, Merge Two Lists)
- 4:00-6:00 PM: Advanced problems (Merge K Lists, Remove Nth Node, Reorder List)

Evening (4 hours): Trees

- 7:00-9:00 PM: Tree traversals (DFS, BFS, Level Order)
- 9:00-11:00 PM: BST problems & Tree construction

Day 2 (14 hours total)

Morning (6 hours): Dynamic Programming

- 7:00-9:00 AM: Basic DP (Climbing Stairs, House Robber, Coin Change)
- 9:00-11:00 AM: String DP (Word Break, Decode Ways)
- 11:00-1:00 PM: Advanced DP (LIS, LCS, Unique Paths)

Afternoon (4 hours): Graphs

- 2:00-4:00 PM: Graph traversal (Number of Islands, Clone Graph)
- 4:00-6:00 PM: Topological Sort (Course Schedule, Alien Dictionary)

Evening (4 hours): Final Review

- 7:00-9:00 PM: Heap problems & remaining categories
- 9:00-11:00 PM: Mock interview practice with top 20 problems

Success Probability Analysis

If you follow this 2-day plan:

- **Arrays/Strings**: 80% mastery (most important for Amazon)
- **Trees/DP**: 60% mastery (good foundation)
- **Graphs**: 50% mastery (basic understanding)
- **Overall interview success**: **65-75%** (assuming strong fundamentals)

Recommendations:

1. **If possible, take 3-4 days** for better retention
2. **Focus heavily on patterns** rather than memorizing solutions
3. **Practice explaining your approach** out loud
4. **Do at least 5 mock interviews** with the top problems
5. **Review Amazon's leadership principles** and prepare examples

Additional Resources

Must-Have Resources:

- **LeetCode Premium** - For Amazon-tagged problems
- **NeetCode** - Pattern-based explanations
- **AlgoExpert** - Comprehensive video solutions
- **InterviewBit** - Topic-wise practice

- **Pramp/InterviewCake** - Mock interviews

Books:

- "Cracking the Coding Interview" by Gayle McDowell
- "Elements of Programming Interviews" by Aziz, Lee, Prakash

Remember: **Quality over quantity**. It's better to deeply understand 40-50 problems than to superficially know all 75. Focus on the patterns and problem-solving approaches that will help you tackle new problems during the interview.

Good luck with your Amazon interview!