Must-Learn DSA Patterns

The CORE 40 Patterns for Interviews & Competitions

---

1. ARRAY & STRING (8 Patterns)

Essential

1. **Two Pointers (Opposite Direction)** - Palindrome, Container with most water
2. **Two Pointers (Same Direction)** - Remove duplicates, Move zeros
3. **Sliding Window (Fixed Size)** - Maximum average subarray
4. **Sliding Window (Variable Size)** - Longest substring without repeating
5. **Kadane's Algorithm** - Maximum subarray sum
6. **Prefix Sum** - Range sum queries
7. **Dutch National Flag** - Sort colors (3-way partition)
8. **Fast & Slow Pointers** - Find duplicate in array

---

2. HASH MAP/SET (3 Patterns)

Essential

9. **Frequency Counter** - Count occurrences, valid anagram
10. **Two Sum Pattern** - Finding pairs with target
11. **Subarray Sum with Prefix** - Subarray sum equals K

---

3. LINKED LIST (4 Patterns)

Essential

12. **Fast & Slow Pointers** - Detect cycle, find middle
13. **Reverse Linked List** - Reverse entire or in groups
14. **Merge Two Sorted Lists** - Merge operation
15. **Remove Nth Node from End** - Two pointers with gap

---

4. STACK & QUEUE (3 Patterns)

4. STACK & QUEUE (3 Patterns)

Essential

16. **Valid Parentheses** - Matching brackets
17. **Monotonic Stack** - Next greater/smaller element
18. **Min/Max Stack** - Stack with O(1) min/max query

---

5. BINARY SEARCH (4 Patterns)

Essential

19. **Basic Binary Search** - Search in sorted array
20. **Search in Rotated Array** - Modified binary search
21. **First/Last Occurrence** - Find boundaries
22. **Binary Search on Answer** - Minimize/maximize with validation

---

6. TREE PATTERNS (6 Patterns)

Essential

23. **DFS Traversal** - Inorder, Preorder, Postorder
24. **BFS (Level Order)** - Level-wise traversal
25. **Lowest Common Ancestor (LCA)** - Find common ancestor
26. **Height/Depth Calculation** - Recursive measurement
27. **Path Sum** - Root to leaf paths
28. **Construct Tree from Traversals** - Rebuild tree

---

7. BINARY SEARCH TREE (2 Patterns)

Essential

29. **Validate BST** - Check BST property
30. **Kth Smallest/Largest** - Inorder traversal

---

8. RECURSION & BACKTRACKING (5 Patterns)

Essential

31. **Subsets** - Generate all subsets (2^n)
32. **Permutations** - All arrangements (n!)
33. **Combinations** - Choose K from N
34. **Combination Sum** - Target sum with candidates
35. **Word Search (Grid Backtracking)** - 2D DFS

---

9. DYNAMIC PROGRAMMING (10 Patterns)

Essential - 1D DP

36. **Fibonacci/Climbing Stairs** - Basic 1D DP
37. **House Robber** - Non-adjacent selection
38. **Longest Increasing Subsequence (LIS)** - $O(n^2)$ and O(n log n)

Essential - 2D DP

39. **Longest Common Subsequence (LCS)** - String matching
40. **Edit Distance** - Transform strings
41. **Unique Paths** - Grid navigation
42. **0/1 Knapsack** - Include/exclude decisions

Essential - Advanced DP

43. **Coin Change** - Minimum coins (unbounded knapsack)
44. **Word Break** - Dictionary-based splitting
45. **Maximum Subarray** - Kadane's (can also be DP)

---

10. GRAPH PATTERNS (8 Patterns)

Essential - Traversal

46. **DFS (Depth First Search)** - Connected components, cycle detection
47. **BFS (Breadth First Search)** - Shortest path (unweighted)
48. **Number of Islands** - Grid DFS/BFS

Essential - Algorithms

49. **Topological Sort** - Course schedule (Kahn's algorithm)
50. **Dijkstra's Algorithm** - Shortest path (weighted)
51. **Union-Find (Disjoint Set)** - Dynamic connectivity
52. **Detect Cycle** - Directed and undirected graphs
53. **Bipartite Check** - Two-coloring

---

11. HEAP/PRIORITY QUEUE (3 Patterns)

Essential

54. **Top K Elements** - Kth largest/smallest
55. **Merge K Sorted Lists** - Min heap
56. **Find Median from Stream** - Two heaps (max + min)

---

12. INTERVALS (3 Patterns)

Essential

57. **Merge Intervals** - Combine overlapping
58. **Insert Interval** - Add and merge
59. **Non-overlapping Intervals** - Minimum removals (greedy)

---

13. BIT MANIPULATION (3 Patterns)

Essential

60. **XOR Properties** - Single number, missing number
61. **Count Set Bits** - Brian Kernighan's algorithm
62. **Power of Two** - n & (n-1) == 0

---

14. TRIE (2 Patterns)

Essential

63. **Implement Trie** - Insert, search, startsWith
64. **Word Search II** - Backtracking + Trie

15. GREEDY (3 Patterns)

Essential

65. **Jump Game** - Can reach end
66. **Meeting Rooms II** - Minimum rooms (interval scheduling)
67. **Activity Selection** - Non-overlapping intervals

---

16. ADVANCED BUT CRUCIAL (3 Patterns)

Essential

68. **Sliding Window Maximum** - Deque pattern
69. **Trapping Rain Water** - Two pointers or stack
70. **Longest Palindromic Substring** - Expand around center

Priority Learning Order

Phase 1: ABSOLUTE MUST (Master First) - 20 Patterns

Focus: 80% of interview problems

1. Two Pointers
2. Sliding Window
3. Hash Map (Frequency Counter)
4. Fast & Slow Pointers
5. Binary Search (Basic)
6. DFS on Trees
7. BFS on Trees
8. Backtracking (Subsets, Permutations)
9. 1D DP (Fibonacci, House Robber)
10. 2D DP (LCS, Unique Paths)
11. Knapsack (0/1)
12. DFS on Graphs

13. BFS on Graphs
14. Union-Find
15. Top K (Heap)
16. Merge Intervals
17. Valid Parentheses
18. Monotonic Stack
19. Kadane's Algorithm
20. Prefix Sum

Phase 2: VERY IMPORTANT - 15 Patterns

Focus: Advanced interview problems

21. Topological Sort
22. Dijkstra's Algorithm
23. LCA (Lowest Common Ancestor)
24. Trie
25. Two Heaps (Median)
26. Binary Search on Answer
27. Edit Distance
28. Word Break
29. Coin Change
30. LIS (Longest Increasing Subsequence)
31. Sliding Window Maximum
32. Trapping Rain Water
33. XOR Patterns
34. Greedy (Jump Game)
35. Construct Tree from Traversals

Phase 3: COMPETITIVE EDGE - 5 Patterns

Focus: Hard interview problems

36. Segment Tree Basics

37. Bit Manipulation Advanced
38. Game Theory (Minimax)
39. Bitmask DP
40. Advanced String (KMP basics)

Weekly Practice Plan

Week 1-2: Arrays & Strings

- Two Pointers (both types)
- Sliding Window
- Prefix Sum
- Kadane's

Week 3-4: Hash & Search

- Hash Map patterns
- Binary Search (all types)
- Fast & Slow pointers

Week 5-6: Lists & Stacks

- Linked List operations
- Stack patterns
- Monotonic Stack

Week 7-8: Trees

- DFS & BFS
- LCA
- Construct trees

Week 9-10: Backtracking

- Subsets & Permutations
- Combinations
- Grid backtracking

Week 11-14: Dynamic Programming

Week 11-14: Dynamic Programming

- 1D DP (Week 11)
- 2D DP (Week 12)
- Knapsack variants (Week 13)
- Advanced DP (Week 14)

Week 15-16: Graphs

- DFS & BFS applications
- Topological Sort
- Union-Find
- Dijkstra's

Week 17-18: Heaps & Intervals

- Top K problems
- Two heaps
- Merge intervals

Week 19-20: Polish & Practice

- Trie
- Greedy
- Hard problems
- Mock interviews

Problem Count Target

Minimum to be interview-ready:

- **Easy**: 50 problems
- **Medium**: 150 problems
- **Hard**: 30 problems
- **Total**: 230 problems

Distribution by pattern:

- Arrays & Strings: 40

- Hash Maps: 15
- Linked Lists: 15
- Stacks: 10
- Binary Search: 15
- Trees: 30
- Backtracking: 20
- DP: 50
- Graphs: 25
- Heaps: 10
- Others: 10

Success Metrics

After 3 months:

✅ Solve 80% of Easy problems
✅ Solve 40% of Medium problems
✅ Recognize 30+ patterns instantly

After 6 months:

✅ Solve 95% of Easy problems
✅ Solve 70% of Medium problems
✅ Solve 20% of Hard problems
✅ Recognize all 40 core patterns

After 12 months:

✅ Solve 100% of Easy problems
✅ Solve 85% of Medium problems
✅ Solve 40% of Hard problems
✅ Ready for FAANG interviews

Key Takeaway

These 40 patterns cover 95% of all interview questions.

Master these in order, practice consistently, and you'll be ready for any coding interview or competition!

Daily commitment: 1-2 hours = 6 months to mastery