Complete DSA Pattern Guide with Complexity

1. Two Pointers

Concept: Use two pointers to scan from both ends of an array or string.
Use Cases: Palindrome check, reverse string, pair sum in sorted array.
Time Complexity: Best/Worst/Optimal - O(n)
Space Complexity: O(1)
Example Problems:
- Reverse String (LeetCode 344)
- Valid Palindrome (LeetCode 125)
- Two Sum II - Input Array Is Sorted (LeetCode 167)
- Remove Duplicates from Sorted Array (LeetCode 26)
- Squares of a Sorted Array (LeetCode 977)

2. Sliding Window

Concept: Fixed/variable size window moves over data to track max/min/sum, etc.
Use Cases: Max sum subarray, longest substring, etc.
Time Complexity: O(n)
Space Complexity: O(1) or O(k) depending on problem
Example Problems:
- Maximum Sum Subarray of Size K
- Longest Substring Without Repeating Characters
- Minimum Window Substring

3. Fast and Slow Pointers

Concept: Move one pointer faster than the other to detect cycles or find mid-points.
Use Cases: Linked lists, cycle detection, finding middle node.
Time Complexity: O(n)
Space Complexity: O(1)
Example Problems:
- Linked List Cycle
- Middle of the Linked List
- Happy Number

4. Merge Intervals

Concept: Sort intervals then merge overlapping ones.
Use Cases: Calendar scheduling, interval management.
Time Complexity: O(n log n)
Space Complexity: O(n) (due to sorting or output list)
Example Problems:
- Merge Intervals
- Insert Interval
- Meeting Rooms

5. Cyclic Sort

Concept: Place each element at its correct index.
Use Cases: Missing numbers, duplicates, index positioning.
Time Complexity: O(n)
Space Complexity: O(1)
Example Problems:
- Missing Number

- Find All Duplicates in an Array
- First Missing Positive

6. In-place Reversal of Linked List

Concept: Reverse nodes by changing pointers.
Use Cases: Reverse all/part of a linked list.
Time Complexity: O(n)
Space Complexity: O(1)
Example Problems:
- Reverse Linked List
- Reverse Linked List II
- Reverse Nodes in k-Group

7. Tree Traversals (DFS/BFS)

Concept: Depth or breadth-based recursive or iterative tree traversal.
Use Cases: Parsing tree structure, ordering elements.
Time Complexity: O(n)
Space Complexity: O(h) for DFS, O(w) for BFS
Example Problems:
- Binary Tree Inorder Traversal
- Binary Tree Level Order Traversal
- Zigzag Level Order Traversal

8. DFS/BFS on Graphs

Concept: Traverse unstructured data using queue/stack/recursion.
Use Cases: Connectivity, path finding, islands.
Time Complexity: O(V + E)
Space Complexity: O(V)
Example Problems:
- Number of Islands
- Clone Graph
- Pacific Atlantic Water Flow

9. Backtracking

Concept: Explore all paths and backtrack if needed.
Use Cases: Combinations, permutations, constraint satisfaction.
Time Complexity: O(2^n) or more (varies)
Space Complexity: O(n) for recursion stack
Example Problems:
- Subsets
- Permutations
- N-Queens
- Sudoku Solver

10. Binary Search

Concept: Divide and conquer search on sorted data.
Use Cases: Efficient search, bounds finding.
Time Complexity: O(log n)
Space Complexity: O(1)
Example Problems:
- Binary Search
- Search in Rotated Sorted Array
- Peak Element

11. Heap / Priority Queue

Concept: Use heap to maintain order for top-k, stream processing.
Use Cases: Top-k elements, median stream.
Time Complexity: O(log k) per operation
Space Complexity: O(k)
Example Problems:
- Kth Largest Element
- Top K Frequent Elements
- Merge K Sorted Lists

12. Greedy

Concept: Make locally optimal choices hoping for global optimum.
Use Cases: Scheduling, path minimizing.
Time Complexity: Varies (typically O(n log n))
Space Complexity: O(1) or O(n)
Example Problems:
- Activity Selection
- Fractional Knapsack
- Jump Game

13. Dynamic Programming (DP)

Concept: Solve problems by combining solutions to subproblems.
Use Cases: Optimization, counting, sequence analysis.
Time Complexity: O(n), O(n^2), O(n*m), etc. (depends on problem)
Space Complexity: O(n) or O(n*m)
Example Problems:
- Climbing Stairs
- Longest Common Subsequence
- 0/1 Knapsack
- Coin Change

14. Bit Manipulation

Concept: Operate directly on bits using bitwise operators.
Use Cases: Toggle, shift, count, XOR tricks.
Time Complexity: O(1) to O(n) (based on bits involved)
Space Complexity: O(1)
Example Problems:
- Single Number
- Number of 1 Bits
- Subsets using Bitmasking

15. Trie (Prefix Tree)

Concept: Tree-based structure for fast word/prefix storage and lookup.
Use Cases: Auto-complete, dictionary, prefix search.
Time Complexity: O(L) where L is word length
Space Complexity: O(n * L)
Example Problems:
- Implement Trie
- Word Search II
- Replace Words