Problem Solving Structured Approach

1. Arrays

- **Sliding Window**:
  - Maximum sum subarray of size K (e.g., LeetCode 53)
  - Longest substring without repeating characters (e.g., LeetCode 3)
- **Two Pointers**:
  - Remove duplicates (e.g., LeetCode 26)
  - Pair sum (e.g., LeetCode 167)
- **Binary Search**:
  - Find min in rotated sorted array (e.g., LeetCode 153)
  - Search in rotated sorted array (e.g., LeetCode 33)
- **Sorting and Greedy**:
  - Meeting rooms (e.g., LeetCode 252)
  - Minimum moves to equal array elements (e.g., LeetCode 453)
- **Prefix Sum**:
  - Subarray sum equals K (e.g., LeetCode 560)
  - Product of array except self (e.g., LeetCode 238)

2. Strings

- **Two Pointers**:
  - Valid palindrome (e.g., LeetCode 125)
  - Reverse vowels of a string (e.g., LeetCode 345)
- **Sliding Window**:
  - Longest substring with at most K distinct characters (e.g., LeetCode 340)
  - Minimum window substring (e.g., LeetCode 76)
- **Dynamic Programming**:
  - Longest palindromic substring (e.g., LeetCode 5)
  - Edit distance (e.g., LeetCode 72)

- **Hashing**:
    - Group anagrams (e.g., LeetCode 49)
    - Longest substring without repeating characters (e.g., LeetCode 3)
- **Backtracking**:
    - Generate parentheses (e.g., LeetCode 22)
    - Word search (e.g., LeetCode 79)

3. Recursion and Backtracking

- **Subset Generation**:
    - Subsets (e.g., LeetCode 78)
    - Permutations (e.g., LeetCode 46)
- **Combination Generation**:
    - Combination sum (e.g., LeetCode 39)
    - Letter combinations of a phone number (e.g., LeetCode 17)
- **Sudoku/Constraint Problems**:
    - Sudoku solver (e.g., LeetCode 37)
    - N-Queens problem (e.g., LeetCode 51)
- **Recursive Backtracking**:
    - Word search (e.g., LeetCode 79)
    - Rat in a maze (classic problem)

4. Stacks

- **Monotonic Stack**:
    - Next greater element (e.g., LeetCode 496)
    - Largest rectangle in histogram (e.g., LeetCode 84)
- **Balanced Parentheses**:
    - Valid parentheses (e.g., LeetCode 20)
    - Minimum add to make parentheses valid (e.g., LeetCode 921)
- **Infix/Postfix Expressions**:

    - Evaluate reverse polish notation (e.g., LeetCode 150)
    - Convert infix to postfix (classic problem)
- **Stack for Tracking Indices**:
    - Daily temperatures (e.g., LeetCode 739)
    - Stock span problem (classic problem)

5. Queues

- **Sliding Window**:
    - Maximum of each sliding window (e.g., LeetCode 239)
- **BFS Traversal**:
    - Level order traversal (e.g., LeetCode 102)
    - Shortest path in binary matrix (e.g., LeetCode 1091)
- **Deque for Double-Ended Processing**:
    - Sliding window maximum (optimized, e.g., LeetCode 239)
- **Circular Queue**:
    - Implement circular queue (e.g., LeetCode 622)

6. Priority Queues

- **Top K Elements**:
    - K closest points (e.g., LeetCode 973)
    - K largest elements (e.g., LeetCode 215)
- **Sliding Window Maximum**:
    - Maximum of each sliding window (e.g., LeetCode 239)
- **Sorting with Heaps**:
    - Sort characters by frequency (e.g., LeetCode 451)
    - Reorganize string (e.g., LeetCode 767)

7. Linked Lists

- **Two Pointers**:
    - Remove N-th node from end (e.g., LeetCode 19)

- **Reversal**:
    - Reverse a linked list (e.g., LeetCode 206)
    - Reverse nodes in k-group (e.g., LeetCode 25)
- **Cycle Detection**:
    - Detect cycle (e.g., LeetCode 141)
    - Find cycle start (e.g., LeetCode 142)
- **Merge**:
    - Merge two sorted lists (e.g., LeetCode 21)
    - Merge K sorted lists (e.g., LeetCode 23)

8. Trees

- **DFS/BFS Traversals**:
    - Preorder traversal (e.g., LeetCode 144)
    - Level-order traversal (e.g., LeetCode 102)
- **Recursion**:
    - Maximum depth of binary tree (e.g., LeetCode 104)
    - Path sum (e.g., LeetCode 112)
- **Binary Search Tree (BST)**:
    - Validate BST (e.g., LeetCode 98)
    - Lowest common ancestor (e.g., LeetCode 235)
- **Backtracking on Trees**:
    - All root-to-leaf paths (e.g., LeetCode 257)
    - Flatten binary tree (e.g., LeetCode 114)

9. Dynamic Programming (DP)

- **0/1 Knapsack**:
    - Subset sum (e.g., LeetCode 416)
    - Partition equal subset sum (e.g., LeetCode 416)
- **Memoization**:

    - Fibonacci sequence (e.g., classic problem)
    - Climbing stairs (e.g., LeetCode 70)
- **Matrix DP**:
    - Unique paths (e.g., LeetCode 62)
    - Minimum path sum (e.g., LeetCode 64)
- **Sequence DP**:
    - Longest increasing subsequence (e.g., LeetCode 300)
    - Longest common subsequence (e.g., LeetCode 1143)
- **Subarray/Subsequence DP**:
    - Maximum sum subarray (e.g., LeetCode 53)
    - Palindromic subsequences (e.g., LeetCode 1312)

10. Intervals

- **Merge Intervals**:
    - Merge intervals (e.g., LeetCode 56)
    - Insert interval (e.g., LeetCode 57)
- **Sorting + Greedy**:
    - Non-overlapping intervals (e.g., LeetCode 435)
    - Meeting rooms II (e.g., LeetCode 253)
- **Two Pointers**:
    - Employee free time (e.g., LeetCode 759)

11. Bit Manipulation

- **Bitwise Operations**:
    - Single number (e.g., LeetCode 136)
    - Counting bits (e.g., LeetCode 338)
- **Masking**:
    - Subset sum problems (custom problems)

- **XOR Patterns**:
    - Missing number (e.g., LeetCode 268)

12. Graphs

- **BFS/DFS**:
    - Connected components in a graph (e.g., LeetCode 200)
    - Shortest path in an unweighted graph (e.g., LeetCode 1091)
- **Topological Sort**:
    - Course schedule (e.g., LeetCode 207)
    - Alien dictionary (e.g., LeetCode 269)
- **Dijkstra's Algorithm**:
    - Network delay time (e.g., LeetCode 743)
- **Union-Find**:
    - Number of connected components (e.g., LeetCode 323)
    - Redundant connection (e.g., LeetCode 684)