DSA PATTERN CHEATSHEET 2025

1. Prefix Sum

Concept: Precompute a prefix sum array where prefix[i] stores the sum of elements from index 0 to i. This enables quick sum queries over any subarray.

LeetCode Problems:

- Range Sum Query - Immutable (LeetCode #303)
- Contiguous Array (LeetCode #525)
- Subarray Sum Equals K (LeetCode #560)

2. Two Pointers

Concept: Use two pointers (either moving towards or away from each other) to efficiently search or process elements in an array.

LeetCode Problems:

- Two Sum II — Sorted Array (LeetCode #167)
- 3Sum (LeetCode #15)
- Container With Most Water (LeetCode #11)
- Trapping Rain Water (LeetCode #42)

3. Sliding Window

Concept: Maintain a dynamic window (subarray or substring) that slides over the input while updating required values efficiently.

LeetCode Problems:

- Longest Substring Without Repeating Characters (LeetCode #3)
- Minimum Window Substring (LeetCode #76)
- Sliding Window Maximum (LeetCode #239)
- Longest Repeating Character Replacement (LeetCode #424)

4. Fast & Slow Pointers (Tortoise and Hare)

Concept: Use two pointers moving at different speeds to detect cycles or find specific elements in linked lists.

LeetCode Problems:

- Linked List Cycle (LeetCode #141)
- Find the Duplicate Number (LeetCode #287)
- Happy Number (LeetCode #202)
- Reorder List (LeetCode #143)

---

5. Linked List In-Place Reversal

Concept: Reverse sections of a linked list in place by adjusting pointers without extra memory.

LeetCode Problems:

- Reverse Linked List (LeetCode #206)
- Reverse Linked List II (LeetCode #92)
- Swap Nodes in Pairs (LeetCode #24)
- Rotate List (LeetCode #61)

---

6. Monotonic Stack

Concept: Use a stack to maintain a sequence of increasing/decreasing elements to solve problems related to the "next greater/smaller" elements.

LeetCode Problems:

- Next Greater Element I (LeetCode #496)
- Daily Temperatures (LeetCode #739)
- Largest Rectangle in Histogram (LeetCode #84)
- Online Stock Span (LeetCode #901)

---

7. Top K Elements (Heap)

Concept: Use heaps (priority queues) or quick-select to efficiently find the k largest/smallest elements.

LeetCode Problems:

- [Kth Largest Element in an Array (LeetCode #215)]
- [Top K Frequent Elements (LeetCode #347)]
- [Find K Pairs with Smallest Sums (LeetCode #373)]

---

8. Overlapping Intervals

Concept: Merge or process overlapping intervals in a sorted list.

LeetCode Problems:

- Merge Intervals (LeetCode #56)
- Insert Interval (LeetCode #57)
- Non-Overlapping Intervals (LeetCode #435)

---

9. Modified Binary Search

Concept: Apply binary search variations on sorted, rotated, or complex datasets.

LeetCode Problems:

- Search in Rotated Sorted Array (LeetCode #33)
- Find Minimum in Rotated Sorted Array (LeetCode #153)
- Search a 2D Matrix II (LeetCode #240)

---

10. Binary Tree Traversal

Concept: Visit all nodes in a tree using different orders.

LeetCode Problems:

- Binary Tree Inorder Traversal (LeetCode #94)
- Binary Tree Zigzag Level Order Traversal (LeetCode #103)
- Binary Tree Paths (LeetCode #257)

---

11. Depth-First Search (DFS)

Concept: Explore as far as possible along each branch before backtracking.

LeetCode Problems:

- Clone Graph (LeetCode #133)
- Path Sum II (LeetCode #113)
- Course Schedule II (LeetCode #210)

---

12. Breadth-First Search (BFS)

Concept: Explore all nodes at the current depth before moving deeper.

LeetCode Problems:

- Binary Tree Level Order Traversal (LeetCode #102)
- Rotting Oranges (LeetCode #994)
- Word Ladder (LeetCode #127)

---

13. Matrix Traversal

Concept: Navigate through matrices using BFS, DFS, or pattern-based traversal.

LeetCode Problems:

- Set Matrix Zeroes (LeetCode #73)
- Number of Islands (LeetCode #200)
- Spiral Matrix (LeetCode #54)

---

14. Backtracking

Concept: Explore all possible choices recursively, undoing changes when necessary.

LeetCode Problems:

- Combination Sum (LeetCode #39)
- Sudoku Solver (LeetCode #37)
- Permutations (LeetCode #46)

15. Dynamic Programming (DP)

Concept: Break a problem into smaller overlapping subproblems, store the results to avoid redundant computations (memoization or tabulation).

Types of DP Approaches:

- **Top-down (Memoization):** Solve recursively and store results.
- **Bottom-up (Tabulation):** Solve iteratively using a DP table.
- **State Transition:** Define dp[i] meaningfully and derive recurrence relations.

LeetCode Problems:

- Climbing Stairs (LeetCode #70) → **Basic DP**
- House Robber (LeetCode #198) → **1D DP**
- Longest Palindromic Substring (LeetCode #5) → **String DP**
- Unique Paths (LeetCode #62) → **Grid DP**
- Coin Change (LeetCode #322) → **Unbounded Knapsack**
- Edit Distance (LeetCode #72) → **2D DP**
- Longest Increasing Subsequence (LeetCode #300) → **LIS Pattern**