📌 DSA Pattern Reference Guide

🔁 Pointer & Window Techniques

Two Pointers

- Sorted array + target sum/diff → inward-moving pointers
- Removing duplicates in-place → fast/slow pointer
- Comparing ends → often sorting + two pointers

Fast & Slow Pointers

- Cycle detection (linked list, number transformation) → Floyd's cycle
- Middle of linked list → slow.next jumps ahead

Sliding Window

- Fixed-length window → maintain sum/count (e.g., max sum subarray of size k)
- Variable-length window → shrinking window based on condition (e.g., smallest subarray > sum)

🧩 Grouping & Sorting

Merge Intervals

- If overlap/merge is mentioned → sort by start, then scan with a merged list
- Calendar/bookings → check for conflicts using sorted intervals

Cyclic Sort

- “Find missing/duplicate number from 1 to N” → place elements at correct index

In-place Linked List Reversal

- Reverse in groups → modify pointers mid-traversal
- Palindrome check / reorder list → split + reverse + merge

🧱 Data Structure Triggers

Stack

- Next Greater/Smaller Element, Undo, Valid Parentheses → classic stack
- Histograms, Stock Span → Monotonic stack variant

Monotonic Stack

- Increasing/Decreasing trends → maintain stack in order (e.g., daily temperatures)

Hash Maps

- “Count of…” → frequency map
- Subarrays with given sum/condition → prefix sum + hash map
- Detect duplicates / indices → store index mapping

🌲 Tree & Graph Traversals

Tree BFS

- Level order traversal, min/max depth, zigzag level → use queue
- Right view, left view → enqueue with level tracking

Tree DFS

- Path sum, serialize/deserialize → post-order often needed
- Recursion with depth or value constraints → classic DFS recursion

Graphs

- Nodes, edges, connections → DFS/BFS/Union Find
- Weighted graph + shortest path → Dijkstra or Bellman-Ford
- Cycles in directed/undirected → DFS with visited/recStack

Islands / Matrix traversal

- “Number of regions/clusters” → DFS/BFS flood-fill
- Directions → use `dirs = [(0,1),(1,0),(-1,0),(0,-1)]` style loop

⚙️ Heap & Binary Search Utilities

Two Heaps

- Median of stream → max heap + min heap balance

Subsets

- “All combinations/subsets/permutations” → backtracking or bitmask

Modified Binary Search

- Sorted or rotated arrays → customized mid conditions
- First/last occurrence → bounds handling

Bitwise XOR

- “Find single non-duplicate” → XOR properties
- “Swap without temp” / parity → XOR tricks

Top K Elements

- “K largest/smallest/frequent” → Min or Max heap

K-way Merge

- Merge sorted arrays/lists → min heap of next elements

🎯 Problem-Solving Strategies

Greedy

- “Minimum X to achieve Y” → sort + greedy choice
- Intervals / activity selection / jumps → greedy on end/start points

0/1 Knapsack (DP)

- “Choose items with weight/value limits” → classic knapsack DP
- Use tabulation or space optimization for constraints

Backtracking

- “All combinations with constraints” → recursive DFS + undo
- Sudoku, N-Queens → build → check → backtrack

Trie

- Prefix, autocomplete, word search → Trie insert/search
- Replace words in sentence → Trie prefix matching

🌐 Graph Variants & Advanced Structures

Topological Sort

- “Order of tasks” or “prerequisites” → DAG + indegree tracking

Union Find

- Connected components, cycle detection, Kruskal’s MST → union by rank + path compression

Ordered Set / TreeMap

- Find floor/ceiling, sliding window max/min → TreeMap or Balanced BST
- Often replaceable with `bisect` in Python or TreeSet in Java

🪛 Concurrency & Miscellaneous

Multi-thread

- Synchronization, shared resource access → locks, semaphores
- Producer-consumer, deadlock scenarios

Miscellaneous

- LRU Cache → HashMap + Doubly Linked List
- Least recent/frequent → Priority queue + HashMap