DSA Patterns Cheat Sheet

---

1. Prefix / Running / Cumulative Sum

- **Idea:** Keep cumulative totals while iterating.
- **Clues:** Subarray sum, range queries, cumulative total.
- **Typical Problems:** Subarray sum = k, max subarray sum, 2D grid sum.

2. Sliding Window

- **Idea:** Maintain a window (fixed/variable) to check conditions.
- **Clues:** Longest/shortest contiguous substring/array.
- **Typical Problems:** Longest substring with K distinct chars, max sum subarray.

3. Two Pointers

- **Idea:** Two indices moving from start/end to optimize search.
- **Clues:** Pairs, triplets, sorted array problems.
- **Typical Problems:** Two sum in sorted array, container with most water.

4. Fast & Slow Pointers

- **Idea:** Two pointers at different speeds.
- **Clues:** Linked list cycles, middle element, repeated elements.
- **Typical Problems:** Detect cycle, find middle of list.

5. Binary Search / Search Space

- **Idea:** Divide-and-conquer on sorted/monotonic array or condition.
- **Clues:** Sorted array, optimization, monotonicity.
- **Typical Problems:** Search in rotated array, minimize max load.

6. Hashmap / Frequency Counting

- **Idea:** Store counts or last seen indices.
- **Clues:** Count occurrences, find duplicates.
- **Typical Problems:** Two sum, longest substring without repeats.

7. Stack / Monotonic Stack

- **Idea:** Maintain order/history efficiently.
- **Clues:** Next greater/smaller, largest rectangle, valid parentheses.
- **Typical Problems:** Next greater element, histogram problem.

8. Queue / Deque

- **Idea:** Order-based processing; deque for sliding window min/max.
- **Clues:** Sliding window, streaming max/min.
- **Typical Problems:** Sliding window maximum, first non-repeating char.

9. Heap / Priority Queue

- **Idea:** Maintain dynamic min/max efficiently.
- **Clues:** Kth largest/smallest, merging sorted structures.
- **Typical Problems:** Kth largest element, merge K sorted arrays.

10. DFS / BFS (Graph or Tree)

- **Idea:** Systematic node exploration.
- **Clues:** Connected components, path counting, shortest path.
- **Typical Problems:** Number of islands, shortest path in unweighted graph.

11. Backtracking / Recursion

- **Idea:** Explore all possibilities with undo.
- **Clues:** Generate combinations, subsets, sequences.
- **Typical Problems:** N-Queens, permutations, word search.

12. Dynamic Programming

- **Idea:** Use previous results to build solutions.
- **Clues:** Overlapping subproblems, optimal substructure.
- **Typical Problems:** Fibonacci, 0-1 Knapsack, Longest increasing subsequence.

13. Greedy / Interval Scheduling

- **Idea:** Make locally optimal choice.
- **Clues:** Optimize count, select intervals.
- **Typical Problems:** Activity selection, min intervals to remove.

14. Union-Find / Disjoint Set

- **Idea:** Track connected components efficiently.
- **Clues:** Connectivity, cycles in graphs.
- **Typical Problems:** Detect cycle, Kruskal's MST.

15. Bit Manipulation

- **Idea:** Use bits to optimize space/time.
- **Clues:** Powers of two, XOR properties.
- **Typical Problems:** Single number, subsets using bits.

16. 2D / Matrix Patterns

- **Idea:** Extend 1D patterns to 2D.
- **Clues:** Grid sums, obstacles, paths.
- **Typical Problems:** Unique paths, max rectangle in binary matrix.

17. Difference Array / Range Update

- **Idea:** Efficiently update ranges using start/end markers.
- **Clues:** Multiple range updates.
- **Typical Problems:** Increment subarrays, flight bookings.

Tip for Recognition: - Look for **keywords in problem**: sum, range, contiguous, pair, sorted, graph/tree, optimization, count. - Map keywords → likely pattern → apply solution approach. - After solving 3–5 problems per pattern, recognition becomes natural.

PDF Export: This document is formatted and ready for PDF conversion for easy reference.