**DSA Patterns by Data Structure & Topic**

## 🧠 1. Arrays & Strings

### 📌 Core Patterns:

* **Sliding Window**
  + Fixed size window (e.g., max sum subarray of size K)
  + Dynamic size window (e.g., longest substring without repeating characters)
* **Two Pointers**
  + Opposite ends (e.g., two sum sorted, reverse string)
  + Fast and slow pointers (e.g., detect cycle in linked list)
* **Prefix Sum / Difference Array**
  + Efficient range sum or range update problems
* **Sorting + Searching**
  + When sorting makes logic easier (e.g., 3Sum, merging intervals)
* **Greedy Choices**
  + When making local optimal choices leads to global optimum (e.g., jump game, gas station)

## 🔗 2. Linked List

### 📌 Core Patterns:

* **Fast and Slow Pointers**
  + Detect cycle, find middle node, palindrome check
* **Reversal Pattern**
  + Reverse entire list or in k-group
* **Dummy Node Trick**
  + Simplifies insertion/deletion (e.g., remove Nth node from end)
* **Merging Lists**
  + Merge K sorted lists, merge two sorted lists
* **In-place Updates**
  + Modify list without extra space

## 🟩 3. Stack / Monotonic Stack

### 📌 Core Patterns:

* **Next Greater / Smaller Element**
  + Nearest greater to right/left (NGTR, NGL, NSR, NSL)
* **Valid Parentheses**
  + Balance checking using stack
* **Histogram Problems**
  + Largest rectangle in histogram, maximal rectangle
* **Stack with Extra Info**
  + Min stack, frequency stack, custom design

## 🕦 4. Queue / Deque

### 📌 Core Patterns:

* **Sliding Window Max/Min**
  + Using deque for O(n) solution
* **Monotonic Queue**
  + For window-based optimization problems
* **BFS Level Order**
  + Tree or graph level traversal

## 🌲 5. Binary Tree / BST

### 📌 Core Patterns:

* **DFS**
  + Preorder, Inorder, Postorder (recursive + iterative)
* **BFS / Level Order**
  + Use queue to traverse level-wise
* **Diameter / Height Pattern**
  + Bottom-up recursion (e.g., diameter of tree)
* **Top View / Bottom View / Zigzag View**
  + Use BFS + index tracking
* **Lowest Common Ancestor**
  + DFS approach, parent pointers, or binary lifting (advanced)
* **BST Specific**
  + Inorder is sorted, properties for search/insert/delete

## 🧲 6. HashMap / Set

### 📌 Core Patterns:

* **Counting Frequency**
  + Majority Element, Anagrams, Valid Sudoku
* **Prefix Sum with HashMap**
  + Subarray with sum K, longest equal subarray
* **Caching**
  + LRU Cache using hashmap + doubly linked list
* **Union-Find with HashMap**
  + Advanced dynamic connectivity

## 🔀 7. Recursion / Backtracking

### 📌 Core Patterns:

* **Generate All Combinations / Permutations**
  + N-Queens, subsets, permutations
* **Decision Tree Traversal**
  + Try all paths with pruning (sudoku, word search)
* **Palindrome Partitioning**
  + Recursively split and check partitions
* **Memoization (Top-Down DP)**
  + Avoid recomputation using cache

## 📀 8. Binary Search

### 📌 Core Patterns:

* **Classic Binary Search**
  + On sorted array
* **Binary Search on Answer**
  + Search space is answer range (e.g., min max distance, kth smallest)
* **Search in Rotated Array**
  + Modified conditions
* **First/Last Occurrence**
  + Boundary finding using BS

## 📊 9. Heap / Priority Queue

### 📌 Core Patterns:

* **Top K Problems**
  + Use min heap of size K
* **Merging K Sorted Lists**
  + Heap of head pointers
* **Median Maintenance**
  + Two heaps (max-left, min-right)
* **Scheduling / Greedy Selection**
  + Task scheduler, meeting rooms

## 🧲 10. Dynamic Programming (DP)

### 📌 Core Patterns:

* **0/1 Knapsack**
  + Pick or not pick item
* **Unbounded Knapsack**
  + Use item multiple times
* **Longest Common Subsequence / Substring**
  + Grid-based recurrence
* **DP on Strings**
  + Edit distance, palindrome partition
* **DP on Trees**
  + DFS + memo
* **Bitmask DP**
  + State compression using bitmasks

## 🌐 11. Graphs

### 📌 Core Patterns:

* **DFS / BFS**
  + Connected components, islands, flood fill
* **Topological Sort**
  + Course schedule (Kahn’s algorithm / DFS)
* **Dijkstra’s Algorithm**
  + Shortest path with priority queue
* **Bellman-Ford / Floyd-Warshall**
  + For negative weights / all pairs
* **Union Find / DSU**
  + Kruskal’s MST, cycle detection
* **Tarjan’s / Kosaraju’s**
  + Strongly connected components (advanced)

## ⚙️ 12. Tries

### 📌 Core Patterns:

* **Autocomplete / Prefix Search**
* **Word Dictionary with Wildcards**
* **Replace Words**
* **Suffix Trie (Advanced)**
* **Bitwise Trie**
  + Max XOR of two numbers

## ⚡️ Bonus: Problem Solving Patterns

### 📌 General Techniques:

* **Greedy**
  + Always choose best immediate option
* **Meet in the Middle**
  + Divide set into halves to optimize
* **Bit Manipulation**
  + XOR tricks, subsets
* **Two DFS / Multi-pass**
  + First DFS to gather info, second to propagate