**✅ 1. ARRAYS**

**🔸 Patterns Used:**

* **Two Pointers**  
  → When: Array is **sorted**, you need to find pairs/sums, or remove duplicates  
  → Problems: Two Sum II, Remove Duplicates, Three Sum  
  → Identify: You have two ends or you want to find something that adds to X.
* **Sliding Window**  
  → When: You deal with **subarrays** of **size K** or **variable size** that meet a condition  
  → Problems: Max Sum Subarray of size K, Longest Substring Without Repeat  
  → Identify: You track a **window** over array and shift it to maintain a constraint.
* **Prefix Sum**  
  → When: Range sums, frequent sum queries  
  → Problems: Subarray Sum Equals K, Range Sum Queries  
  → Identify: You're asked about sum of elements between index i and j.
* **Binary Search on Answer**  
  → When: You must **minimize/maximize** something like minimum time or capacity  
  → Problems: Capacity to Ship Packages, Koko Eating Bananas  
  → Identify: Binary search not on array values, but on possible answers.

**✅ 2. STRINGS**

**🔸 Patterns Used:**

* **Two Pointers**  
  → When: Palindromes, reverse operations  
  → Problems: Valid Palindrome, Reverse String
* **Sliding Window**  
  → When: Longest/shortest substring with condition  
  → Problems: Longest Substring Without Repeating Characters, Anagram substring
* **Hashing (freq maps)**  
  → When: Anagrams, char frequency, grouping  
  → Problems: Group Anagrams, Valid Anagram
* **Trie (Advanced)**  
  → When: Prefix-based queries  
  → Problems: Word Search, Implement Trie

**✅ 3. LINKED LIST**

**🔸 Patterns Used:**

* **Two Pointers (Fast & Slow)**  
  → When: Detect cycle, find middle, reverse  
  → Problems: Linked List Cycle, Middle of Linked List
* **Reverse Pointer Techniques**  
  → Problems: Reverse Linked List, Merge Two Sorted Lists

**✅ 4. HASHMAP / SET**

**🔸 Patterns Used:**

* **Hashing**  
  → When: Unordered access, frequency, presence check in O(1)  
  → Problems: Two Sum, Isomorphic Strings, First Unique Character
* **Prefix Sum + Hashing**  
  → When: Store cumulative sum and its index  
  → Problems: Subarray Sum Equals K

**✅ 5. STACK / DEQUE**

**🔸 Patterns Used:**

* **Monotonic Stack**  
  → When: Next Greater/Smaller element, histogram  
  → Problems: Daily Temperatures, Largest Rectangle in Histogram
* **Sliding Window Maximum**  
  → Use: Monotonic Deque  
  → Problems: Max Sliding Window

**✅ 6. HEAP / PRIORITY QUEUE**

**🔸 Patterns Used:**

* **Min/Max Heap**  
  → When: You need top K elements, or merge K sorted arrays  
  → Problems: Kth Largest Element, Merge K Lists
* **Greedy + Heap**  
  → When: Process by priority  
  → Problems: Task Scheduler, Reorganize String

**✅ 7. BINARY SEARCH**

**🔸 Patterns Used:**

* **Classic Binary Search**  
  → When: Find index in sorted array  
  → Problems: Search Insert Position
* **First/Last Occurrence (lower\_bound / upper\_bound)**  
  → Problems: Count Occurrences, Range Queries
* **Binary Search on Answer**  
  → Problems: Minimum Time to Complete Jobs

**✅ 8. TREES**

**🔸 Patterns Used:**

* **DFS / Recursion**  
  → When: Traversals, height, path sum  
  → Problems: Max Depth, Balanced Tree
* **BFS**  
  → When: Level Order, shortest path  
  → Problems: Level Order Traversal
* **Inorder Traversal**  
  → When: BST-related problems  
  → Problems: Validate BST

**✅ 9. GRAPHS**

**🔸 Patterns Used:**

* **DFS / BFS**  
  → Problems: Number of Islands, Clone Graph
* **Topological Sort**  
  → Problems: Course Schedule
* **Union Find (DSU)**  
  → Problems: Graph Connectivity, Kruskal’s MST

**✅ 10. DYNAMIC PROGRAMMING**

**🔸 Patterns Used:**

* **0/1 Knapsack**
* **Fibonacci (Top-Down, Bottom-Up)**
* **DP on Strings**  
  → LCS, Edit Distance
* **DP on Grids**  
  → Unique Paths, Min Path Sum

Identify:

If the problem asks for "maximum/minimum ways", or optimal solution with overlapping subproblems — it’s **DP**.

**✅ 11. BACKTRACKING**

**🔸 Patterns Used:**

* **DFS + Backtrack**  
  → When: Explore all paths, undo decisions  
  → Problems: N-Queens, Sudoku, Subsets, Permutations

**🔑 How to Identify the Pattern:**

| **Clue in Problem** | **Pattern** |
| --- | --- |
| “Find all subarrays of size K” | Sliding Window |
| “Check if sorted” | Two Pointers |
| “Find two elements that sum to target” | Hashing / Two Pointers |
| “Track top K or min/max” | Heap |
| “Range sum queries” | Prefix Sum |
| “Avoid recomputation” | DP |
| “Check connectivity / components” | Graph / DFS / DSU |
| “Repeated decisions, backtracking needed” | Backtracking |
| “Substring/Anagram/Char freq” | Hashing + Sliding Window |
| “Minimize/maximize answer” | Binary Search on Answer |