
⛓ Patterns for Arrays & Strings

1. **Sliding Window**
   - **Use when:** You need to find the optimal (min/max/longest) contiguous subarray/substring.
   - **Problems to Master:**
     - Maximum Sum Subarray of Size K
     - Longest Substring Without Repeating Characters
     - Minimum Window Substring
2. **Two Pointers**
   - **Use when:** Working with sorted arrays to find pairs/triplets, or for problems involving palindromes.
   - **Problems to Master:**
     - Two Sum II - Input Array Is Sorted
     - 3Sum
     - Container With Most Water
3. **Prefix Sum**
   - **Use when:** You need to answer multiple queries about the sum of a given range [i, j].
   - **Problems to Master:**
     - Range Sum Query - Immutable
     - Subarray Sum Equals K
4. **Merge Intervals** *(New Pattern)* or overlapping interva
   - **Use when:** Dealing with problems involving overlapping intervals or scheduling. The key is almost always to sort the intervals by their start time first.
   - **Problems to Master:**
     - Merge Intervals
     - Insert Interval
     - Meeting Rooms II and Non-overlapping Intervals
5. **Hashing / Frequency Counter**
   - **Use when:** You need to count items, find duplicates, or check for anagrams. It provides O(1) average time lookups.
   - **Problems to Master:**
     - Two Sum
     - Valid Anagram
     - Group Anagrams
6. **Kadane's Algorithm**
   - **Use when:** The problem asks for the maximum subarray sum in a 1D array.
   - **Problems to Master:**
     - Maximum Subarray

        - Maximum Product Subarray
7. **Cyclic Sort** *(New Pattern)*
    - **Use when:** The problem involves an array containing numbers in a specific range (e.g., 1 to N). It's a powerful in-place technique for finding missing or duplicate numbers in O(1) space.
    - **Problems to Master:**
        - Missing Number
        - Find all Duplicates in an Array
        - Find the First Missing Positive

🔗 Patterns for Linked Lists

8. **Fast & Slow Pointers (Floyd's Cycle-Finding)**
    - **Use when:** You need to detect cycles, find the middle of a linked list, or find the start of a cycle.
    - **Problems to Master:**
        - Linked List Cycle
        - Middle of the Linked List
        - Find the Duplicate Number
9. **In-place Reversal**
    - **Use when:** The problem requires reversing a linked list (or a part of it) with O(1) space complexity.
    - **Problems to Master:**
        - Reverse a Linked List
        - Reverse Nodes in k-Group

🌳 Patterns for Trees & Graphs 🌲

9.9 Binary Tree Traversa

pre , inorder, post , level order

Problems to master
257. Binary tree Path
230. Kth Smallest Element in a BST
124. Binary Tree maximum Path sum
107. Binary Tree level Order Traversal

10. **Breadth-First Search (BFS)**
    - **Use when:** Finding the shortest path in an unweighted graph or for level-order traversal. Uses a queue.
    - **Problems to Master:**
        - Binary Tree Level Order Traversal
        - Rotting Oranges
        - Word Ladder
11. **Depth-First Search (DFS)**
    - **Use when:** Checking for path existence, finding connected components, or solving maze-like problems. Uses recursion or a stack.

11.9 Matrix traversal pattern

Problems:
733. Flood fill
200. Numbers of Island
130. Surrounding Regions

- **Problems to Master:**
  - Number of Islands
  - Path Sum II
  - Clone Graph

12. **Topological Sort**
    - **Use when:** The problem involves dependencies, prerequisites, or a required order of tasks in a Directed Acyclic Graph (DAG).
    - **Problems to Master:**
      - Course Schedule
      - Course Schedule II
      - Alien Dictionary
13. **Union-Find (Disjoint Set Union - DSU)**
    - **Use when:** You need to check for connectivity in an undirected graph or find redundant connections efficiently.
    - **Problems to Master:**
      - Number of Connected Components
      - Redundant Connection
      - Number of Provinces
14. **Dijkstra's Algorithm / A* Search**
    - **Use when:** You need to find the shortest path from a single source in a **weighted graph** with non-negative edge weights.
    - **Problems to Master:**
      - Network Delay Time
      - Path with Maximum Probability
15. **Trie (Prefix Tree)**
    - **Use when:** The problem involves string prefixes, auto-completion, or dictionary lookups.
    - **Problems to Master:**
      - Implement Trie (Prefix Tree)
      - Word Search II

💡 Patterns for Heaps

16. **Top 'K' Elements**
    - **Use when:** You need to find the "Kth largest/smallest" element or the "Top K" items from a collection. or Top K most frequent
    - **Problems to Master:**
      - Kth Largest Element in an Array
      - Top K Frequent Elements
17. **Two Heaps**

- **Use when:** You need to find the median of a data stream or partition data into two balanced halves.
- **Problems to Master:**
  - Find Median from Data Stream
  - Sliding Window Median

18. **Merge K Sorted Lists**
    - **Use when:** You need to merge k sorted lists, arrays, or streams into a single sorted list.
    - **Problems to Master:**
      - Merge k Sorted Lists
      - Find K Pairs with Smallest Sums

🧩 Patterns for DP & Backtracking

19. **0/1 Knapsack**
    - **Use when:** You have to make a choice for each item (take it or not) to optimize some value given a constraint.
    - **Problems to Master:**
      - Partition Equal Subset Sum
      - Target Sum
20. **Longest Common Subsequence/Substring** and Longest Increasing Subsequence
    - **Use when:** You need to compare two sequences to find their longest shared sequence.
    - **Problems to Master:**
      - Longest Common Subsequence
      - Longest Palindromic Substring
21. **Fibonacci-style Problems**
    - **Use when:** The solution for state n depends on solutions for states n-1, n-2, etc.
    - **Problems to Master:**
      - Climbing Stairs
      - House Robber
22. **Coin Change**
    - **Use when:** You need to find combinations or permutations that sum up to a target, often with an infinite supply of elements.
    - **Problems to Master:**
      - Coin Change
      - Combination Sum IV
23. **Subsets / Permutations / Combinations (Backtracking)**
    - **Use when:** You need to generate all possible solutions (subsets, permutations, etc.).
    - **Problems to Master:**

23.9 MCM

    - Subsets
    - Permutations
    - Combination Sum

⚙️ Miscellaneous Patterns

24. **Standard Binary Search** *(New Pattern)*
    - **Use when:** You need to efficiently find an element in a **sorted** collection.
    - **Problems to Master:**
        - Binary Search
        - Find First and Last Position of Element in Sorted Array
        - Search in Rotated Sorted Array
25. **Binary Search on Answer**
    - **Use when:** You can guess an answer and verify if it's feasible in a monotonic way (i.e., if an answer x works, any answer x' > x also works).
    - **Problems to Master:**
        - Split Array Largest Sum
        - Koko Eating Bananas
26. **Bit Manipulation**
    - **Use when:** You need fast, memory-efficient operations on numbers, often for finding unique elements or handling sets of flags.
    - **Problems to Master:**
        - Single Number
        - Number of 1 Bits
        - Counting Bits
27. **Monotonic Stack/Queue**
    - **Use when:** You need to find the next/previous greater/smaller element for each item in an array.
    - **Problems to Master:**
        - Next Greater Element I
        - Daily Temperatures