# LeetCode Patterns for Interviews

This document lists common LeetCode patterns that are frequently encountered in technical interviews. Understanding these patterns can help you approach a wide variety of problems.

## 1. Sliding Window

This pattern is used for problems that involve finding a subarray or substring in an array or string that satisfies certain conditions.

- **Problems:**
  - Maximum Sum Subarray of Size K (easy) (Note: This is a general max subarray, a fixed size K version is a good starting point for sliding window)
  - Longest Substring with K Distinct Characters (medium)
  - Smallest Subarray with a given sum (easy)
  - Longest Substring Without Repeating Characters (medium)
  - Minimum Window Substring (hard)

## 2. Two Pointers

This pattern uses two pointers to iterate through an array or list, often from opposite ends or with one pointer moving faster than the other.

- **Problems:**
  - Pair with Target Sum (easy) (Can be solved with hashmap too, but two pointers work if sorted)
  - Remove Duplicates (easy)
  - Squares of a Sorted Array (easy)
  - 3Sum (medium)
  - Container With Most Water (medium)

## 3. Fast & Slow Pointers (Floyd's Tortoise and Hare)

This pattern is often used in problems involving linked lists or arrays where you need to detect cycles or find a specific position.

- **Problems:**
  - Linked List Cycle (easy)
  - Middle of the Linked List (easy)
  - Happy Number (easy)
  - Palindrome Linked List (easy)
  - Linked List Cycle II (medium) (Find the start of the cycle)

## 4. Merge Intervals

This pattern involves merging overlapping intervals.

- **Problems:**
  - Merge Intervals (medium)
  - Insert Interval (medium)
  - Interval List Intersections (medium)
  - Non-overlapping Intervals (medium)
  - Meeting Rooms II (medium)

## 5. Cyclic Sort

This pattern is used for problems where you have an array containing numbers in a given range (e.g., 1 to N) and you need to sort it or find missing/duplicate numbers.

- **Problems:**
  - Cyclic Sort (easy concept, often part of medium problems) (Conceptual link, as LeetCode might not have a direct "Cyclic Sort" problem but uses the pattern)
  - Find the Missing Number (easy)
  - Find all Duplicates in an Array (medium)
  - Find the Duplicate Number (medium)
  - Find the First K Missing Positive Numbers (hard) (Related concept)

## 6. In-place Reversal of a LinkedList

This pattern involves reversing parts of a linked list without using extra space.

- **Problems:**
  - Reverse a LinkedList (easy)
  - Reverse a Sub-list (medium)
  - Reverse Nodes in k-Group (hard)

## 7. Tree Breadth-First Search (BFS)

This pattern is used to traverse a tree level by level.

- **Problems:**
  - Binary Tree Level Order Traversal (medium)
  - Zigzag Level Order Traversal (medium)
  - Average of Levels in Binary Tree (easy)
  - Minimum Depth of Binary Tree (easy)
  - Connect Level Order Siblings (medium)

## 8. Tree Depth-First Search (DFS)

This pattern is used to traverse a tree by exploring as far as possible along each branch before backtracking.

- **Problems:**
  - Path Sum (easy)
  - Path Sum II (medium)
  - Binary Tree Maximum Path Sum (hard)
  - Diameter of Binary Tree (easy)
  - Sum of Path Numbers (medium)

## 9. Two Heaps

This pattern uses two heaps (min-heap and max-heap) to solve problems involving finding medians, or partitioning elements.

- **Problems:**
  - Find Median from Data Stream (hard)
  - Sliding Window Median (hard)
  - IPO (hard)

## 10. Subsets

This pattern deals with generating all possible subsets (or permutations, combinations) of a given set.

- **Problems:**
  - Subsets (medium)
  - Subsets II (with duplicates) (medium)
  - Permutations (medium)
  - Permutations II (with duplicates) (medium)
  - Combinations (medium)

## 11. Modified Binary Search

This pattern applies binary search to problems that don't involve a simple sorted array, but where the search space can be divided.

- **Problems:**
  - Search in Rotated Sorted Array (medium)
  - Find Minimum in Rotated Sorted Array (medium)
  - Median of Two Sorted Arrays (hard)
  - Search in a Sorted Array of Unknown Size (medium)

## 12. Top 'K' Elements

This pattern involves finding the top 'K' largest/smallest elements, most frequent elements, etc. Often solved using heaps or quickselect.

- **Problems:**
  - Top K Frequent Elements (medium)
  - Kth Largest Element in an Array (medium)
  - K Closest Points to Origin (medium)
  - Sort Characters By Frequency (medium)

## 13. K-way Merge

This pattern is used for problems that involve merging multiple sorted lists or arrays.

- **Problems:**
  - Merge K Sorted Lists (hard)
  - Kth Smallest Element in a Sorted Matrix (medium) (Can also be solved with binary search)
  - Find K Pairs with Smallest Sums (medium)

## 14. Dynamic Programming

Dynamic Programming (DP) is a broad pattern that involves breaking down a problem into smaller overlapping subproblems and storing the results of subproblems to avoid redundant computations.

- **General Categories & Problems:**
  - **0/1 Knapsack:**
    - Partition Equal Subset Sum (medium)
    - Target Sum (medium)
  - **Unbounded Knapsack:**
    - Coin Change (medium)
    - Combination Sum IV (medium)
  - **Longest Common Subsequence/Substring:**
    - Longest Common Subsequence (medium)
    - Edit Distance (medium)
  - **Fibonacci Style / Sequences:**
    - Climbing Stairs (easy)
    - House Robber (medium)
    - Unique Paths (medium)
  - **Palindromic Subsequence/Substring:**
    - Longest Palindromic Substring (medium)

        - Palindromic Substrings (medium)

## 15. Topological Sort

This pattern is used for problems involving directed acyclic graphs (DAGs) where you need to find a linear ordering of nodes such that for every directed edge from node A to node B, node A comes before node B in the ordering.

- **Problems:**
    - Course Schedule (medium)
    - Course Schedule II (medium)
    - Alien Dictionary (hard)
    - Minimum Height Trees (medium)

## 16. Trie (Prefix Tree)

This pattern uses a tree-like data structure to efficiently store and retrieve strings based on their prefixes.

- **Problems:**
    - Implement Trie (Prefix Tree) (medium)
    - Word Search II (hard)
    - Design Add and Search Words Data Structure (medium)
    - Maximum XOR of Two Numbers in an Array (medium)

## 17. Backtracking

This pattern is used for problems that require exploring all possible solutions by incrementally building a candidate solution and abandoning a candidate ("backtracking") as soon as it is determined that the candidate cannot possibly be completed to a valid solution.

- **Problems:**
    - Letter Combinations of a Phone Number (medium)
    - Generate Parentheses (medium)
    - N-Queens (hard)
    - Word Search (medium) (Often uses DFS/Backtracking)

**Note:** Some problems can be solved using multiple patterns, and the categorization here is based on common approaches. The difficulty levels (easy, medium, hard) are as per LeetCode.