1. Fast and Slow Pointer

**Description**: This technique uses two pointers moving at different speeds to solve problems involving cycles, such as finding the middle of a list, detecting loops, or checking for palindromes.

* [Linked List Cycle II](https://leetcode.com/problems/linked-list-cycle-ii/)
* [Remove nth Node from the End of List](https://leetcode.com/problems/remove-nth-node-from-end-of-list/)
* [Find the Duplicate Number](https://leetcode.com/problems/find-the-duplicate-number/)
* [Palindrome Linked List](https://leetcode.com/problems/palindrome-linked-list/)

2. Overlapping Intervals

**Description**: Intervals are often manipulated through sorting and merging based on their start and end times.

* [Basic Merge: Merge Intervals](https://leetcode.com/problems/merge-intervals/)
* [Interval Insertion: Insert Interval](https://leetcode.com/problems/insert-interval/)
* [Minimum Number of Arrows to Burst Balloons](https://leetcode.com/problems/minimum-number-of-arrows-to-burst-balloons/)
* [Meeting Rooms ii](https://leetcode.com/problems/meeting-rooms-ii/)
* [Non-overlapping Intervals](https://leetcode.com/problems/non-overlapping-intervals/)

3. Sliding Window

**Description**: A sliding window is a subarray or substring that moves over data to solve problems efficiently in linear time.

Fixed Size

* [Maximum Sum Subarray of Size K](https://leetcode.com/problems/maximum-sum-of-distinct-subarrays-with-length-k/)
* [Number of Subarrays having Average Greater or Equal to Threshold](https://leetcode.com/problems/number-of-sub-arrays-of-size-k-and-average-greater-than-or-equal-to-threshold/)
* [Sliding Subarray Beauty](https://leetcode.com/problems/sliding-subarray-beauty/)
* [Permutation in String](https://leetcode.com/problems/permutation-in-string/)
* [Sliding Window Maximum](https://leetcode.com/problems/sliding-window-maximum/)

Variable Size

* [Longest Substring Without Repeating Characters](https://leetcode.com/problems/longest-substring-without-repeating-characters/)
* [Minimum Size Subarray Sum](https://leetcode.com/problems/minimum-size-subarray-sum/)
* [Subarray Product Less Than K](https://leetcode.com/problems/subarray-product-less-than-k/)
* [Max Consecutive Ones](https://leetcode.com/problems/max-consecutive-ones-iii/)
* [Fruits Into Baskets](https://leetcode.com/problems/fruit-into-baskets/)
* [Count Number of Nice Subarrays](https://leetcode.com/problems/count-number-of-nice-subarrays)
* [Minimum Window Substring: Minimum Window Substring](https://leetcode.com/problems/minimum-window-substring/)

4. Two Pointers

**Description**: The two pointers technique involves having two different indices move through the input at different speeds to solve various array or linked list problems.

* [Two Sum II - Input Array is Sorted](https://leetcode.com/problems/two-sum-ii-input-array-is-sorted/)
* [Dutch National Flag: Sort Colors](https://leetcode.com/problems/sort-colors/)
* [Next Permutation](https://leetcode.com/problems/next-permutation/)
* [Bag of Tokens](https://leetcode.com/problems/bag-of-tokens/)
* [Container with most water](https://leetcode.com/problems/container-with-most-water/)
* [Trapping Rain Water](https://leetcode.com/problems/trapping-rain-water/)

5. Cyclic Sort (Index-Based)

**Description**: Cyclic sort is an efficient approach to solve problems where numbers are consecutively ordered and must be placed in the correct index.

* [Missing Number](https://leetcode.com/problems/missing-number/)
* [Find Missing Numbers](https://leetcode.com/problems/find-all-numbers-disappeared-in-an-array/)
* [Set Mismatch](https://leetcode.com/problems/set-mismatch/)
* [First Missing Positive](https://leetcode.com/problems/first-missing-positive/)

6. Reversal of Linked List (In-place)

**Description**: Reversing a linked list in place without using extra space is key for problems that require in-place list manipulations.

* [Reverse Linked List](https://leetcode.com/problems/reverse-linked-list/)
* [Reverse Nodes in k-Group](https://leetcode.com/problems/reverse-nodes-in-k-group/)
* [Swap Nodes in Pairs](https://leetcode.com/problems/swap-nodes-in-pairs/)

7. Matrix Manipulation

**Description**: Problems involving 2D arrays (matrices) are often solved using row-column traversal or manipulation based on matrix properties.

* [Rotate Image](https://leetcode.com/problems/rotate-image/)
* [Spiral Matrix](https://leetcode.com/problems/spiral-matrix/)
* [Set Matrix Zeroes](https://leetcode.com/problems/set-matrix-zeroes/)
* [Game of Life](https://leetcode.com/problems/game-of-life/)

8. Breadth First Search (BFS)

**Description**: BFS explores nodes level by level using a queue. It is particularly useful for shortest path problems.

* [Shortest Path in Binary Matrix](https://leetcode.com/problems/shortest-path-in-binary-matrix/)
* [Rotten Oranges](https://leetcode.com/problems/rotting-oranges/)
* [As Far From Land as Possible](https://leetcode.com/problems/as-far-from-land-as-possible/)
* [Word Ladder: Word Ladder](https://leetcode.com/problems/word-ladder/)

9. Depth First Search (DFS)

**Description**: DFS explores as far as possible along a branch before backtracking. It's useful for graph traversal, pathfinding, and connected components.

* [Number of Closed Islands](https://leetcode.com/problems/number-of-closed-islands/)
* [Coloring a Border](https://leetcode.com/problems/coloring-a-border/)
* [DFS from boundary: Number of Enclaves](https://leetcode.com/problems/number-of-enclaves/)
* [Shortest time: Time Needed to Inform all Employees](https://leetcode.com/problems/time-needed-to-inform-all-employees/)
* [Cyclic Find: Find Eventual Safe States](https://leetcode.com/problems/find-eventual-safe-states/)

10. Backtracking

**Description**: Backtracking helps in problems where you need to explore all potential solutions, such as solving puzzles, generating combinations, or finding paths.

* [Permutation ii](https://leetcode.com/problems/permutations-ii/)
* [Combination Sum](https://leetcode.com/problems/combination-sum/)
* [Generate Parenthesis](https://leetcode.com/problems/generate-parentheses/)
* [N-Queens](https://leetcode.com/problems/n-queens/)
* [Sudoku Solver](https://leetcode.com/problems/sudoku-solver/)
* [Palindrome Partitioning](https://leetcode.com/problems/palindrome-partitioning/)
* [Word Search: Word Search](https://leetcode.com/problems/word-search/)

11. Modified Binary Search

**Description**: A modified version of binary search that applies to rotated arrays, unsorted arrays, or specialized conditions.

* [Search in Rotated Sorted Array](https://leetcode.com/problems/search-in-rotated-sorted-array-ii/)
* [Find Minimum in Rotated Sorted Array](https://leetcode.com/problems/find-minimum-in-rotated-sorted-array/)
* [Find Peak Element](https://leetcode.com/problems/find-peak-element/)
* [Minimum Time to Arrive on Time](https://leetcode.com/problems/minimum-speed-to-arrive-on-time/)
* [Capacity to Ship Packages within 'd' Days](https://leetcode.com/problems/capacity-to-ship-packages-within-d-days/)
* [Koko Eating Bananas](https://leetcode.com/problems/koko-eating-bananas)
* [Find in Mountain Array](https://leetcode.com/problems/find-in-mountain-array/)
* [Median of Two Sorted Arrays](https://leetcode.com/problems/median-of-two-sorted-arrays/)

12. Bitwise XOR

**Description**: XOR is a powerful bitwise operator that can solve problems like finding single numbers or efficiently pairing elements.

* [Missing Number](https://leetcode.com/problems/missing-number/)
* [Single Number ||](https://leetcode.com/problems/single-number-ii/)
* [Single Number III](https://leetcode.com/problems/single-number-iii/)
* [Find the Original array of Prefix XOR](https://leetcode.com/problems/find-the-original-array-of-prefix-xor/)
* [XOR Queries of a Subarray](https://leetcode.com/problems/xor-queries-of-a-subarray/)

13. Top 'K' Elements

**Description**: This pattern uses heaps or quickselect to efficiently find the top 'K' largest/smallest elements from a dataset.

* [Top K Frequent Elements](https://leetcode.com/problems/top-k-frequent-elements/)
* [Kth Largest Element](https://leetcode.com/problems/kth-largest-element-in-an-array/)
* [Ugly Number ii](https://leetcode.com/problems/ugly-number-ii/)
* [K Closest Points to Origin](https://leetcode.com/problems/k-closest-points-to-origin/)

14. K-way Merge

**Description**: The K-way merge technique uses a heap to efficiently merge multiple sorted lists or arrays.

* [Find K Pairs with Smallest Sums](https://leetcode.com/problems/find-k-pairs-with-smallest-sums/)
* [Kth Smallest Element in a Sorted Matrix](https://leetcode.com/problems/kth-smallest-element-in-a-sorted-matrix/)
* [Merge K Sorted Lists](https://leetcode.com/problems/merge-k-sorted-lists/)
* [Smallest Range: Smallest Range Covering Elements from K Lists](https://leetcode.com/problems/smallest-range-covering-elements-from-k-lists/)

15. Two Heaps

**Description**: This pattern uses two heaps (max heap and min heap) to solve problems involving tracking medians and efficiently managing dynamic data.

* [Find Median from Data Stream](https://leetcode.com/problems/find-median-from-data-stream/)
* [Sliding Window Median](https://leetcode.com/problems/sliding-window-median/)
* [IPO](https://leetcode.com/problems/ipo/)

16. Monotonic Stack

**Description**: A monotonic stack helps solve range queries by maintaining a stack of elements in increasing or decreasing order.

* [Next Greater Element II](https://leetcode.com/problems/next-greater-element-ii/)
* [Next Greater Node in Linked List](https://leetcode.com/problems/next-greater-node-in-linked-list/)
* [Daily Temperatures](https://leetcode.com/problems/daily-temperatures/)
* [Online Stock Span](https://leetcode.com/problems/online-stock-span/)
* [Maximum Width Ramp](https://leetcode.com/problems/maximum-width-ramp/)
* [Largest Rectangle in Histogram](https://leetcode.com/problems/largest-rectangle-in-histogram/)

17. **Trees**

Level Order Traversal (BFS in Binary Tree)

* [Level order Traversal](https://leetcode.com/problems/binary-tree-level-order-traversal/)
* [Zigzag Level order Traversal](https://leetcode.com/problems/binary-tree-zigzag-level-order-traversal/)
* [Even Odd Tree](https://leetcode.com/problems/even-odd-tree/)
* [Reverse odd Levels](https://leetcode.com/problems/reverse-odd-levels-of-binary-tree/)
* [Deepest Leaves Sum](https://leetcode.com/problems/deepest-leaves-sum/)
* [Add one row to Tree](https://leetcode.com/problems/add-one-row-to-tree/)
* [Maximum width of Binary Tree](https://leetcode.com/problems/maximum-width-of-binary-tree/)
* [All Nodes Distance K in Binary tree](https://leetcode.com/problems/all-nodes-distance-k-in-binary-tree/)

Tree Construction

* [Construct BT from Preorder and Inorder](https://leetcode.com/problems/construct-binary-tree-from-preorder-and-inorder-traversal/)
* [Construct BT from Postorder and Inorder](https://leetcode.com/problems/construct-binary-tree-from-inorder-and-postorder-traversal/)
* [Maximum Binary Tree](https://leetcode.com/problems/maximum-binary-tree/)
* [Construct BST from Preorder](https://leetcode.com/problems/construct-binary-search-tree-from-preorder-traversal/)

Height related Problems

* [Maximum Depth of BT](https://leetcode.com/problems/maximum-depth-of-binary-tree/)
* [Balanced Binary Tree](https://leetcode.com/problems/balanced-binary-tree/)
* [Diameter of Binary Tree](https://leetcode.com/problems/diameter-of-binary-tree/)
* [Minimum Depth of BT](https://leetcode.com/problems/minimum-depth-of-binary-tree/)
* [Binary Tree Maximum Path Sum](https://leetcode.com/problems/binary-tree-maximum-path-sum/)

Root to leaf path problems

* [Binary Tree Paths](https://leetcode.com/problems/binary-tree-paths/)
* [Path Sum ii](https://leetcode.com/problems/path-sum-ii/)
* [Sum Root to Leaf numbers](https://leetcode.com/problems/sum-root-to-leaf-numbers/)
* [Smallest string starting from Leaf](https://leetcode.com/problems/smallest-string-starting-from-leaf)
* [Insufficient nodes in root to Leaf](https://leetcode.com/problems/insufficient-nodes-in-root-to-leaf-paths/)
* [Pseudo-Palindromic Paths in a Binary Tree](https://leetcode.com/problems/pseudo-palindromic-paths-in-a-binary-tree/)

Ancestor problem

* [LCA of Binary Tree](https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree/)
* [Maximum difference between node and ancestor](https://leetcode.com/problems/maximum-difference-between-node-and-ancestor/)
* [LCA of deepest leaves](https://leetcode.com/problems/lowest-common-ancestor-of-deepest-leaves/)
* [Kth Ancestor of a Tree Node](https://leetcode.com/problems/kth-ancestor-of-a-tree-node/)

Binary Search Tree

* [Validate BST](https://leetcode.com/problems/validate-binary-search-tree/)
* [Range Sum of BST](https://leetcode.com/problems/range-sum-of-bst/)
* [Minimum Absolute Difference in BST](https://leetcode.com/problems/minimum-absolute-difference-in-bst/)
* [Insert into a BST](https://leetcode.com/problems/insert-into-a-binary-search-tree/)
* [LCA of BST](https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-search-tree/)

18. **DYNAMIC PROGRAMMING**

Take / Not take (DP)

**Description**: Solve optimization problems like selecting items with the max/min value under certain constraints.

* [House Robber ii](https://leetcode.com/problems/house-robber-ii/)
* [Target Sum](https://leetcode.com/problems/target-sum/)
* [Partition Equal Subset Sum](https://leetcode.com/problems/partition-equal-subset-sum/)
* [Ones and Zeroes](https://leetcode.com/problems/ones-and-zeroes/)
* [Last Stone Weight ii](https://leetcode.com/problems/last-stone-weight-ii/)

Infinite Supply (DP)

**Description**: Similar to the 0/1 knapsack, but items can be chosen multiple times.

* [Coin Change](https://leetcode.com/problems/coin-change/)
* [Coin Change II](https://leetcode.com/problems/coin-change-ii/)
* [Perfect Squares](https://leetcode.com/problems/perfect-squares/)
* [Minimum Cost For Tickets](https://leetcode.com/problems/minimum-cost-for-tickets/)

Longest Increasing subsequence

**Description**: It involves finding the longest subsequence of a given sequence where the elements are in ascending order

* [Longest Increasing Subsequence](https://leetcode.com/problems/longest-increasing-subsequence)
* [Largest Divisible Subset](https://leetcode.com/problems/largest-divisible-subset/)
* [Maximum Length of Pair Chain](https://leetcode.com/problems/maximum-length-of-pair-chain/)
* [Number of LIS](https://leetcode.com/problems/number-of-longest-increasing-subsequence/)
* [Longest String Chain](https://leetcode.com/problems/longest-string-chain/)

DP on Grids

**Description**: Dynamic Programming on matrices involves solving problems that can be broken down into smaller overlapping subproblems within a matrix.

* [Unique Paths ii](https://leetcode.com/problems/unique-paths-ii/)
* [Minimum Path Sum](https://leetcode.com/problems/minimum-path-sum/)
* [Triangle](https://leetcode.com/problems/triangle/)
* [Minimum Falling Path Sum](https://leetcode.com/problems/minimum-falling-path-sum/)
* [Maximal Square](https://leetcode.com/problems/maximal-square/)
* [Cherry Pickup](https://leetcode.com/problems/cherry-pickup/)
* [Dungeon Game: Dungeon Game](https://leetcode.com/problems/dungeon-game/)

DP on Strings

**Description**: It Involves 2 strings, whenever you are considering two substrings/subsequence from given two strings, concentrate on what happens when the last characters of the two substrings are same, i.e, matching.

* [Longest Common Subsequence](https://leetcode.com/problems/longest-common-subsequence/)
* [Longest Palindromic Subsequence](https://leetcode.com/problems/longest-palindromic-subsequence/)
* [Palindromic Substrings](https://leetcode.com/problems/palindromic-substrings/)
* [Longest Palindromic Substrings](https://leetcode.com/problems/longest-palindromic-substring/)
* [Edit Distance](https://leetcode.com/problems/edit-distance/)
* [Minimum ASCII Delete Sum for Two Strings](https://leetcode.com/problems/minimum-ascii-delete-sum-for-two-strings/)
* [Distinct Subsequences](https://leetcode.com/problems/distinct-subsequences/)
* [Shortest Common Supersequence](https://leetcode.com/problems/shortest-common-supersequence/)
* [Wildcard Matching](https://leetcode.com/problems/wildcard-matching/)

DP on Stocks

**Description**: It focuses on maximizing profit from buying and selling stocks over time while considering constraints.

* [Buy and Sell Stocks ii](https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii/)
* [Buy and Sell Stocks iii](https://leetcode.com/problems/best-time-to-buy-and-sell-stock-iii/)
* [Buy and Sell Stocks iv](https://leetcode.com/problems/best-time-to-buy-and-sell-stock-iv/)
* [Buy and Sell Stocks with Cooldown](https://leetcode.com/problems/best-time-to-buy-and-sell-stock-with-cooldown/)
* [Buy and Sell Stocks with Transaction fee](https://leetcode.com/problems/best-time-to-buy-and-sell-stock-with-transaction-fee/)

Partition DP (MCM)

**Description**: It Involves a sequence that needs to be divided into partitions in an optimal way. The goal is often to minimize or maximize a cost function, such as computation time, multiplications, or some other metric, by exploring all possible partitions and combining results from subproblems.

* [Partition array for Maximum Sum](https://leetcode.com/problems/partition-array-for-maximum-sum/)
* [Burst Balloons](https://leetcode.com/problems/burst-balloons/)
* [Minimum Cost to Cut a Stick](https://leetcode.com/problems/minimum-cost-to-cut-a-stick/)
* [Palindrome Partitioning ii](https://leetcode.com/problems/palindrome-partitioning-ii/)

19. **Graphs**

Topological Sort

**Description**: Topological sorting is useful for tasks that require dependency resolution (InDegree) in directed acyclic graphs (DAGs).

* [Course Schedule](https://leetcode.com/problems/course-schedule/)
* [Course Schedule II](https://leetcode.com/problems/course-schedule-ii/)
* [Sequence Reconstruction](https://leetcode.com/problems/sequence-reconstruction/)
* [Alien Dictionary](https://leetcode.com/problems/alien-dictionary/)

Union Find (Disjoint Set)

**Description**: Union-Find (or Disjoint Set) is used to solve problems involving connectivity or grouping, often in graphs.

* [Number of Operations to Make Network Connected](https://leetcode.com/problems/number-of-operations-to-make-network-connected/)
* [Redundant Connection](https://leetcode.com/problems/redundant-connection/)
* [Accounts Merge](https://leetcode.com/problems/accounts-merge/)
* [Satisfiability of Equality Equations](https://leetcode.com/problems/satisfiability-of-equality-equations/)

Graph Algorithms

**Description**: Advanced graph algorithms are used to solve complex problems involving shortest paths, minimum spanning trees, and graph cycles.

* [Kruskal's Algorithm: Minimum Cost to connect all Points](https://leetcode.com/problems/min-cost-to-connect-all-points/)
* [Dijkstra's Algorithm: Cheapest Flights Within K Stops](https://leetcode.com/problems/cheapest-flights-within-k-stops/)
* [Floyd-Warshall: Find the City with Smallest Number of Neighbours at a Threshold Distance](https://leetcode.com/problems/find-the-city-with-the-smallest-number-of-neighbors-at-a-threshold-distance/)
* [Bellman Ford: Network Delay time](https://leetcode.com/problems/network-delay-time)

20. Greedy

**Description**: Greedy algorithms make local optimal choices at each step, which lead to a global optimal solution for problems like scheduling and resource allocation.

* [Jump Game ii](https://leetcode.com/problems/jump-game-ii/)
* [Gas Station](https://leetcode.com/problems/gas-station/)
* [Bag of Tokens](https://leetcode.com/problems/bag-of-tokens/)
* [Boats to Save People](https://leetcode.com/problems/boats-to-save-people/)
* [Wiggle Subsequence](https://leetcode.com/problems/wiggle-subsequence/)
* [Car Pooling](https://leetcode.com/problems/car-pooling/)
* [Candy](https://leetcode.com/problems/candy/)

21. Design Data Structure

**Description**: It involves building custom data structures to efficiently handle specific operations, like managing data access, updates, and memory usage. Focusing on optimizing performance and resource management.

* [Design Twitter](https://leetcode.com/problems/design-twitter/)
* [Design Browser History](https://leetcode.com/problems/design-browser-history/)
* [Design Circular Deque](https://leetcode.com/problems/design-circular-deque/)
* [Snapshot Array](https://leetcode.com/problems/snapshot-array/)
* [LRU Cache](https://leetcode.com/problems/lru-cache/)
* [LFU Cache](https://leetcode.com/problems/lfu-cache/)

**Some Useful Articles on LeetCode for Better Understanding!**

Two Pointers

* [Solved all Two Pointers problems in 100 days](https://leetcode.com/discuss/study-guide/1688903/Solved-all-two-pointers-problems-in-100-days)

Sliding Window

* [Sliding Window Technique and Question Bank](https://leetcode.com/discuss/study-guide/1773891/Sliding-Window-Technique-and-Question-Bank)
* [C++ Maximum Sliding Window Cheatsheet Template!](https://leetcode.com/problems/frequency-of-the-most-frequent-element/solutions/1175088/C++-Maximum-Sliding-Window-Cheatsheet-Template/)

Greedy

* [Greedy for Beginners: Problems & Sample Solutions](https://leetcode.com/discuss/general-discussion/669996/greedy-for-beginners-problems-sample-solutions)
* [Top Greedy Questions](https://leetcode.com/discuss/interview-question/3972722/Top-Greedy-Questions-helpful-for-OA-and-Interviews)

Linked List

* [Become Master In Linked List](https://leetcode.com/discuss/study-guide/1800120/become-master-in-linked-list)
* [Must-Do LinkedList Problems on LeetCode](https://sarthak-acoustic.medium.com/must-do-linkedlist-problems-on-leetcode-19f47dc88fff)

Trees

* [Tree Question Pattern | 2021 Placement](https://leetcode.com/discuss/study-guide/1337373/Tree-question-pattern-oror2021-placement)
* [Master Tree Patterns](https://leetcode.com/discuss/study-guide/5020529/Master-Tree-Patterns/)

Binary Search

* [5 Variations of Binary Search](https://leetcode.com/discuss/interview-question/1322500/5-variations-of-Binary-search-(A-Self-Note))
* [Binary Search for Beginners: Problems & Patterns](https://leetcode.com/discuss/general-discussion/691825/Binary-Search-for-Beginners-Problems-or-Patterns-or-Sample-solutions)

Dynamic Programming (DP)

* [Dynamic Programming Patterns](https://leetcode.com/discuss/general-discussion/458695/Dynamic-Programming-Patterns)
* [DP for Beginners: Problems & Patterns](https://leetcode.com/discuss/general-discussion/662866/DP-for-Beginners-Problems-or-Patterns-or-Sample-Solutions)

Graphs

* [Graph For Beginners](https://leetcode.com/discuss/general-discussion/655708/graph-for-beginners-problems-pattern-sample-solutions/)
* [Become Master In Graph](https://leetcode.com/discuss/study-guide/2360573/become-master-in-graph)
* [Graph algorithms + problems to practice](https://leetcode.com/discuss/study-guide/1326900/graph-algorithms-problems-to-practice)

Bit Manipulation

* [Bit Manipulation Problem solving](https://leetcode.com/problems/sum-of-two-integers/solutions/84278/A-summary:-how-to-use-bit-manipulation-to-solve-problems-easily-and-efficiently/)
* [All Types of Patterns for Bits Manipulations and How to use i](https://leetcode.com/discuss/interview-question/3695233/all-types-of-patterns-for-bits-manipulations-and-how-to-use-it)