# BitFlip’s Leetcode Pattern Recognition Cheat Sheet

<https://github.com/bitflipdev/guides/blob/main/Leetcode%20Pattern%20Recognition%20Guide.pdf>

# Leetcode Patterns

Sources:

* <https://www.youtube.com/watch?v=DjYZk8nrXVY&t=93s&ab_channel=AshishPratapSingh>
* <https://www.linkedin.com/pulse/20-coding-patterns-master-dsa-data-structures-algorithms-ankit-malik/>
* <https://www.linkedin.com/posts/navdeep-singh-3aaa14161_if-i-only-had-a-short-time-to-study-for-my-activity-7312473774598012928-ZhXz?utm_source=social_share_send&utm_medium=member_desktop_web&rcm=ACoAACMGebYBKY8HnG-p3doo8gpo-3Dc_TXuHFI>
* <https://gist.github.com/PrinceSinghhub/c74372270cf45df2087816a75c87bb06>

## Fast and Slow Pointers

Used in linked list or array problems, this pattern is ideal for detecting cycles or finding a midpoint.

* 141. Linked List Cycle
* 202. Happy Number
* 287. Find the Duplicate Number
* 142. Linked List Cycle II
* 19. Remove nth Node from the End of List
* 243. Palindrome Linked List

## Overlapping Intervals / Merge Intervals

Use this pattern to deal with overlapping intervals, helping to create a more organized and efficient structure. Intervals are often manipulated through sorting and merging based on their start and end times.

* 56. Merge Intervals (Basic Merge)
* 57. Insert Interval (Interval Insertion)
* 435. Non-overlapping intervals
* 731. My Calendar II
* 452. Minimum Number of Arrows to Burst Balloons

## Prefix Sum

Prefix Sums/Products are techniques that store cumulative sums or products up to each index, allowing for quick subarray range queries.

* Longest subarray with sum K (array having Positive and Negative elements)
* 303. Range Sum Query - Immutable
* 525. Contiguous Array
* 560. Subarray Sum Equals K
* 1991. Find the Middle Index in Array
* 238. Product of Array Except Self
* 152. Maximum Product Subarray
* 2270. Number of Ways to Split Array
* 304. Range Sum Query 2D

## Sliding Window

This pattern is used to track a subset of data within a larger dataset. It's particularly useful in array or string problems when you need to maintain a 'window' of elements satisfying a certain condition.

### Fixed Size

* 643. Maximum Average Subarray I
* 424. Longest Repeating Character Replacement
* Maximum Sum Subarray of Size K
* 1343. Number of Sub-arrays of Size K and Average Greater or Equal to Threshold
* 187. Repeated DNA Sequences
* 567. Permutation in String
* 2653. Sliding Subarray Beauty
* 239. Sliding Window Maximum

### Variable Size

* 3. Longest Substring without Repeating Characters
* 76. Minimum Window Substring
* 209. Minimum Size Subarray Sum
* 713. Subarray Product Less Than K
* 485. Max Consecutive Ones
* 904. Fruits Into Baskets
* 1248. Count Number of Nice Subarrays

## Two Pointers

This technique is commonly applied on sorted arrays or linked lists to find pairs or reverse elements. It is an ideal strategy when managing elements with pair relationships.

* 167. Two Sum II - Input Array is Sorted
* 15. 3 Sum
* 11. Container with most water
* 75. Sort Colors (Dutch National Flag)
* 31. Next Permutation
* 948. Bag of Tokens
* 11. Container with Most Water
* 42. Trapping Rain Water

## 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
* Find Missing Numbers
* Set Mismatch
* First Missing Positive

## Linked List in-place reversal

Reversing a linked list in place without using extra space is key for problems that require in-place list manipulations. 206. Reverse Linked List

* 92. Reverse Linked List II
* 24. Swap Nodes in Pairs
* 206. Reverse Linked List
* 25. Reverse Nodes in k-Group
* 24. Swap Nodes in Pairs

## Matrix Traversal

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

* 733. Flood Fill
* 200. Number of Islands
* 130. Surrounded Regions
* 48. Rotate Image
* 54. Spiral Matrix
* 73. Set Matrix Zeroes
* 289. Game of Life

## 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
* 994. Rotting Oranges
* 127. Word Ladder
* As Far From Land as Possible

## 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
* Coloring a Border
* DFS from Boundary: Number of Enclaves
* Shortest Time: Time Needed to Inform All Employees
* Cyclic Find: Find Eventual Safe States

## Backtracking

Backtracking helps in problems where you need to explore all potential solutions, such as solving puzzles, generating combinations, or finding paths.46. Permutations

* 78. Subsets
* 51. N-Queens
* Permutation II
* 39. Combination Sum
* 22. Generate Parenthesis
* 51. N-Queens
* 37. Sudoku Solver
* 131. Palindrome Partitioning
* 79. Word Search

## Modified Binary Search

An adaptation of the binary search for situations where a standard binary search doesn't apply. A modified version of binary search that applies to rotated arrays, unsorted arrays, or specialized conditions.

* 33. Search in Rotated Sorted Array
* 153. Find Minimum in Rotated Sorted Array
* 240. Search a 2D Matrix II
* 875. Koko Eating Bananas
* 162. Find Peak Element
* 540. Single Element in a Sorted Array
* 1870. Minimum Speed to Arrive on Time
* 1011. Capacity to Ship Packages Within 'd' Days
* 1095. Find in Mountain Array
* 4. Median of Two Sorted Arrays

## Bitwise XOR

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

* Missing Number
* Single Number II
* Single Number III
* Find the Original Array of Prefix XOR
* XOR Queries of a Subarray

## Top ‘K’ Elements OR min/max Heap

This pattern is beneficial for problems that require identifying the top or bottom 'k' elements in a set.

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

* 215. Kth Largest element in an array
* 347. Top K Frequent Elements
* 264. Ugly Number II
* 973. K Closest Points to Origin

## K-way Merge

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

* 373. Find K Pairs with Smallest Sums
* 378. Kth Smallest Element in a Sorted Matrix
* 23. Merge K Sorted Lists
* 632. Smallest Range Covering Elements from K Lists

## 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
* Sliding Window Median
* IPO

## Monotonic Stack

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

* 496. Next Greater Element I
* 503. Next Greater Element II
* 739. Daily Temperatures
* 84. Largest Rectangle in Histogram
* 1019. Next Greater Node in Linked List
* 901. Online Stock Span
* 962. Maximum Width Ramp
* 84. Largest Rectangle in Histogram

## Trees

### Depth First Search

This pattern allows you to traverse a tree or graph using depth as the main factor.

* 100. Same Binary Tree (DFS or BFS)
* 257. Binary Tree Paths
* 133. Clone Graph
* 113. Path Sum II
* 210. Course Schedule II
* 124. Binary Tree Maximum Path Sum
* 107. Binary Tree Level Order Traversal II
* 230. Kth Smallest Element in a BST
* 297. Serialize and Deserialize Binary Tree (DFS or BFS)
* 129. Sum Root to Leaf Numbers
* 988. Smallest String Starting from Leaf
* 1080. Insufficient Nodes in Root to Leaf
* 1457. Pseudo-Palindromic Paths in a Binary Tree
* 124. Binary Tree Maximum Path Sum

### Breadth First Search (Level Order Traversal)

Perfect for traversing a tree level-by-level, providing a comprehensive overview of all nodes.

* 102. Binary Tree Level Order Traversal
* 103. Binary Tree Zigzag Level Order Traversal
* 1609. Even Odd Tree
* 2415. Reverse Odd Levels of Binary Tree
* 1302. Deepest Leaves Sum
* 623. Add One Row to Tree
* 662. Maximum Width of Binary Tree
* 863. All Nodes Distance K in Binary Tree

### Tree Construction

* 105. Construct BT from Preorder and Inorder
* 106. Construct BT from Postorder and Inorder
* 654. Maximum Binary Tree
* 1008. Construct BST from Preorder

### Height Related Problems

* Maximum Depth of BT
* Balanced Binary Tree
* Diameter of Binary Tree
* Minimum Depth of BT

### Ancestor Problem

* LCA of Binary Tree
* Maximum Difference Between Node and Ancestor
* LCA of Deepest Leaves
* Kth Ancestor of a Tree Node

### Binary Search Tree

* Validate BST
* Range Sum of BST
* Minimum Absolute Difference in BST
* Insert into a BST
* LCA of BST

## Dynamic Programming

### Take / Not Take (DP)

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

* 70. Climbing Stairs
* House Robber II
* Target Sum
* 416: Partition Equal Subset Sum
* Ones and Zeroes
* Last Stone Weight II

### Infinite Supply (DP)

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

* 322: Coin Change
* Coin Change II
* Perfect Squares
* 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.

* 300: Longest Increasing Subsequence
* Largest Divisible Subset
* Maximum Length of Pair Chain
* Number of LIS
* 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
* Minimum Path Sum
* Triangle
* Minimum Falling Path Sum
* Maximal Square
* Cherry Pickup
* Dungeon Game

### DP on Strings

Description: It involves two strings, focusing on what happens when the last characters of the two substrings are the same.

* 1143: Longest Common Subsequence
* Longest Palindromic Subsequence
* Palindromic Substrings
* Longest Palindromic Substrings
* Edit Distance
* Minimum ASCII Delete Sum for Two Strings
* Distinct Subsequences
* Shortest Common Supersequence
* 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
* Buy and Sell Stocks III
* Buy and Sell Stocks IV
* Buy and Sell Stocks with Cooldown
* Buy and Sell Stocks with Transaction Fee

### Partition DP (MCM - Matrix Chain Multiplication)

Description: It involves a sequence that needs to be divided into partitions in an optimal way.

* Partition Array for Maximum Sum
* 312: Burst Balloons
* Minimum Cost to Cut a Stick
* Palindrome Partitioning II

## Graphs

### Topological Sort

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

* Course Schedule
* Course Schedule II
* Strange Printer II
* Sequence Reconstruction
* 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
* Redundant Connection
* Accounts Merge
* 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
* Dijkstra's Algorithm: Cheapest Flights Within K Stops
* Floyd-Warshall: Find the City with Smallest Number of Neighbours at a Threshold Distance
* Bellman Ford: Network Delay Time

## 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
* Gas Station
* Bag of Tokens
* Boats to Save People
* Wiggle Subsequence
* Car Pooling
* Candy

## Design Data Structure

Description: It involves building custom data structures to efficiently handle specific operations, like managing data access, updates, and memory usage.

* Design Twitter
* Design Browser History
* Design Circular Deque
* Snapshot Array
* LRU Cache
* LFU Cache

**Floyd's Cycle Detection Algorithm**: Ideal for finding cycles in data structures such as linked lists or arrays.

**Kadane’s Algorithm (Dynamic Programming)**: It's an optimal solution for the maximum subarray problem.