1. **Sliding Window**:

- Description: Efficiently handles problems involving contiguous subarrays or substrings.

- Common Problems: Maximum sum subarray of size k, longest substring without repeating characters, minimum window substring.

- Techniques: Fixed-size window, dynamic window (expand/contract).

2. **Two Pointers**:

- Description: Uses two pointers to iterate through data structures, often in a bi-directional manner.

- Common Problems: Pair with target sum, sorting linked lists, palindrome checking, removing duplicates.

- Techniques: Moving pointers towards each other or in the same direction.

3. **Fast and Slow Pointers** (Tortoise and Hare):

- Description: Detects cycles within a sequence, often in linked lists.

- Common Problems: Cycle detection, finding the middle of a linked list.

- Technique: One pointer moves twice as fast as the other.

4. **Merge Intervals**:

- Description: Deals with problems involving intervals, such as merging overlapping intervals.

- Common Problems: Merging intervals, inserting intervals, meeting rooms problem.

- Techniques: Sorting intervals, then merging or inserting accordingly.

5. **Cyclic Sort**:

- Description: Efficiently sorts an array with certain properties.

- Common Problem: Find the missing number, find all missing numbers, find duplicates.

- Techniques: Placing elements at their correct positions.

6. **In-place Reversal of a Linked List**:

- Description: Involves reversing a linked list or part of it without using extra space.

- Common Problems: Reverse entire linked list, reverse sublist, rearrange linked list.

- Techniques: Iterative reversal using a few pointers.

7. **Tree BFS/DFS**:

- Description: Traverses trees or graphs either level-by-level (BFS) or depth-wise (DFS).

- Common Problems: Level order traversal, find minimum depth, zigzag traversal.

- Technique: Using queues for BFS, recursion or stacks for DFS.

8. **Two Heaps**:

- Description: Uses two heaps to maintain a stream of data efficiently.

- Common Problems: Median of a data stream, sliding window median.

- Techniques: Min-heap and max-heap to keep track of lower and upper halves.

9. **Subsets**:

- Description: Deals with problems involving combinations and permutations.

- Common Problem: Subsets, permutations, combinations, letter case permutation.

- Techniques: Iterative or recursive generation of subsets/permutations.

10. **Modified Binary Search**:

- Description: Solves search-related problems with modifications to the binary search algorithm.

- Common Problem: Order-agnostic binary search, ceiling of a number, rotation count.

- Techniques: Adjusting binary search for specific conditions.

11. **Top K Elements**:

- Description: Finds the top K elements in a dataset.

- Common Problems: Kth largest element, K closest points to origin.

- Techniques: Min-heap or max-heap, quickselect.

12. **K-way Merge**:

- Description: Merges K sorted arrays or lists.

- Common Problems: Merge K sorted lists, smallest range covering elements from K lists.

- Technique: Using a min-heap to efficiently merge.

13. **Knapsack (Dynamic Programming)**:

- Description: Solves optimization problems using dynamic programming.

- Common Problem: 0/1 knapsack, unbounded knapsack, subset sum.

- Techniques: DP table construction, space optimization.

14. **Bit Manipulation**:

- Description: Uses bitwise operations to solve problems.

- Common Problems: Single number, number of 1 bits, power of two.

- Techniques: Bitwise AND, OR, XOR operations.

15. **Graph Algorithms**:

- Description: Solves graph-related problems using standard algorithms.

- Common Problems: Shortest path (Dijkstra), connected components (Union-Find), topological sort.

- Techniques: BFS, DFS, priority queues, union-find data structure.

16. **Backtracking**:

- Description: Explores all possible solutions and backtracks when a solution is not viable.

- Common Problem: N-Queens, Sudoku solver, combination sum.

- Techniques: Recursive exploration and pruning of solution space.

By mastering these patterns and recognizing which one applies to a given problem, you can efficiently tackle a wide range of DSA challenges on LeetCode, Geeks for Geeks, and similar platforms.