Array Problem Solving Patterns

list of patterns that need to solve in array's to get good

To get good at solving array-based problems, it's important to master a variety of patterns that are commonly encountered in technical interviews and competitive programming. Here's a list of essential array problem-solving patterns:

1. Sliding Window Pattern

- Used when dealing with a contiguous subset of elements in an array (e.g., subarrays, substrings).
- **Example problems:**
    - Maximum sum subarray of size `` `k` ``.
    - Longest substring without repeating characters.
    - Smallest subarray with a given sum.

2. Two Pointers Pattern

- Useful for problems that require searching for pairs or triplets with specific properties.
- **Example problems:**
    - Pair with a target sum (sorted array).
    - Removing duplicates from a sorted array.
    - Triplets with a zero sum.

3. Fast and Slow Pointers (Tortoise and Hare)

- Typically used in cycle detection or problems that require detecting loops.
- **Example problems:**
    - Detect cycle in a linked list.
    - Find the starting point of a cycle in a circular array.

4. Prefix Sum Pattern

- Often used to find the sum of elements in a subarray efficiently.
- **Example problems:**
    - Subarray with a target sum.
    - Maximum sum subarray (Kadane’s algorithm).

5. Merge Intervals Pattern

- Useful for problems that involve overlapping intervals.
- **Example problems:**
    - Merge overlapping intervals.
    - Insert interval into a list of non-overlapping intervals.

    - Meeting room scheduling.

6. Cyclic Sort

- Efficient for problems where the array contains numbers in a range.
- **Example problems:**
    - Find the missing number.
    - Find all duplicate numbers in an array.
    - Find the smallest missing positive integer.

7. In-place Reversal of Array or Linked List

- For problems that involve reversing or rotating arrays or linked lists without extra space.
- **Example problems:**
    - Reverse a subarray.
    - Rotate an array.
    - Reverse a linked list in-place.

8. Greedy Algorithm (for Arrays)

- Used when locally optimal choices lead to a global solution.
- **Example problems:**
    - Jump Game.
    - Minimum number of platforms.
    - Maximize profit from stock prices.

9. Dutch National Flag Problem

- A three-way partitioning algorithm useful for sorting arrays with limited values.
- **Example problems:**
    - Sort an array of 0s, 1s, and 2s.
    - Segregate even and odd numbers in an array.

10. Binary Search on Arrays

- Efficient search for sorted arrays.
- **Example problems:**
    - Find the position of an element in a sorted array.
    - Search in a rotated sorted array.
    - Find the peak element in an array.

11. Matrix Manipulation (2D Arrays)

- Techniques for working with matrices.
- **Example problems:**
    - Rotate a matrix by 90 degrees.
    - Spiral order traversal.
    - Search for a target in a sorted 2D matrix.

12. Subsets and Subarrays

- Problems involving generating all subsets, subarrays, or combinations.

- **Example problems:**
    - Generate all subsets of an array (Power Set).
    - Generate all subarrays of an array.
    - Find subarrays with a given sum.

13. Monotonic Stack/Queue

- Used to solve problems related to ranges or spans.
- **Example problems:**
    - Find the next greater element.
    - Largest rectangle in a histogram.
    - Trapping rainwater problem.

14. Backtracking

- Useful for problems that require exploring all possible solutions.
- **Example problems:**
    - Permutations of an array.
    - Subset sum problem.
    - Combination sum.

15. Divide and Conquer

- Often used for problems that can be divided into subproblems and solved recursively.
- **Example problems:**
    - Merge sort.
    - Count inversions in an array.
    - Maximum subarray sum using divide and conquer.

16. Kadane's Algorithm (Maximum Sum Subarray)

- Used to find the largest sum of a contiguous subarray.
- **Example problems:**
    - Maximum sum subarray.
    - Maximum product subarray (modified version).

17. Counting and Hashing

- Helpful for counting occurrences of elements or efficiently checking for duplicates.
- **Example problems:**
    - Find the majority element.
    - Find duplicate elements in an array.
    - Find anagrams in an array.

18. Top K Elements

- Used for finding the top or smallest K elements in an array.
- **Example problems:**
    - Find the Kth largest/smallest element.
    - Top K frequent elements.
    - K closest points to the origin.

19. Union-Find (Disjoint Set)

- A technique for solving problems involving connected components in graphs or arrays.
- **Example problems:**
    - Number of connected components.
    - Find redundant connections in a network.

20. Dynamic Programming on Arrays

- Useful for optimization problems where decisions are based on previous subproblems.
- **Example problems:**
    - Longest increasing subsequence.
    - Coin change problem.
    - Maximum sum of non-adjacent elements.

Practice Approach:

- Start with understanding the problem clearly.
- Identify which pattern is likely to help you solve it.
- Implement the pattern.
- Practice variations of the problem to solidify your understanding.

By mastering these patterns, you'll be well-prepared for array-related challenges in technical interviews and coding competitions!

ChatGPT can make mistakes. Check important info.