The **Two Pointer** technique is a versatile and efficient approach used to solve problems involving arrays, strings, or sequences. It involves using two pointers (indices) to traverse the data structure, often reducing the time complexity from **O(n^2)** to **O(n)**. Below is a comprehensive list of **Two Pointer variations** categorized by type:

---

1. Opposite Direction Two Pointers

These problems involve two pointers starting at opposite ends of the array and moving towards each other.

1. **Two Sum**
    - Problem: Given a sorted array, find two numbers that add up to a target.
    - Example:
        - Input: `nums = [2, 7, 11, 15], target = 9`
        - Output: `[2, 7]`
2. **Three Sum**
    - Problem: Given an array, find all unique triplets that add up to zero.
    - Example:
        - Input: `nums = [-1, 0, 1, 2, -1, -4]`
        - Output: `[[-1, -1, 2], [-1, 0, 1]]`
3. **Four Sum**
    - Problem: Given an array, find all unique quadruplets that add up to a target.
    - Example:
        - Input: `nums = [1, 0, -1, 0, -2, 2], target = 0`
        - Output: `[[-2, -1, 1, 2], [-2, 0, 0, 2], [-1, 0, 0, 1]]`
4. **Container With Most Water**
    - Problem: Given an array of heights, find two lines that form a container with the most water.
    - Example:
        - Input: `height = [1, 8, 6, 2, 5, 4, 8, 3, 7]`
        - Output: `49` (between indices 1 and 8)
5. **Valid Palindrome**
    - Problem: Given a string, check if it is a palindrome after removing non-alphanumeric characters and ignoring cases.
    - Example:
        - Input: `s = "A man, a plan, a canal: Panama"`
        - Output: `True`
6. **Trapping Rain Water**
    - Problem: Given an array of heights, compute how much water can be trapped after raining.
    - Example:

- Input: $height = [0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1]$
- Output: 6

---

2. Same Direction Two Pointers (Sliding Window)

These problems involve two pointers starting at the same end and moving in the same direction, often maintaining a window.

7. **Minimum Size Subarray Sum**
   - Problem: Given an array of positive integers, find the length of the smallest subarray with a sum $\geq k$.
   - Example:
     - Input: $nums = [2, 3, 1, 2, 4, 3], k = 7$
     - Output: 2 (subarray $[4, 3]$)
8. **Longest Substring Without Repeating Characters**
   - Problem: Given a string, find the length of the longest substring without repeating characters.
   - Example:
     - Input: $s = "abcabcbb"$
     - Output: 3 (substring $"abc"$)
9. **Longest Substring with At Most K Distinct Characters**
   - Problem: Given a string, find the length of the longest substring with at most k distinct characters.
   - Example:
     - Input: $s = "aabacbebebe", k = 3$
     - Output: 7 (substring $"cbebebe"$)
10. **Fruit Into Baskets**
    - Problem: Given an array of fruit types, find the maximum number of fruits you can collect with at most 2 types of fruits.
    - Example:
      - Input: $fruits = [1, 2, 1, 3, 4, 3, 5, 1, 2]$
      - Output: 4 (subarray $[1, 2, 1, 3]$)
11. **Maximum Consecutive Ones III**
    - Problem: Given a binary array, find the maximum number of consecutive 1s after flipping at most k 0s.
    - Example:
      - Input: $nums = [1, 1, 0, 0, 1, 1, 1, 0, 1, 1], k = 2$
      - Output: 7

---

3. Fast and Slow Pointers

These problems involve two pointers moving at different speeds, often used in linked lists or cyclic arrays.

12. **Linked List Cycle**
    - Problem: Given a linked list, determine if it has a cycle.
    - Example:
        - Input: $1 \text{ -> } 2 \text{ -> } 3 \text{ -> } 4 \text{ -> } 2$ (cycle back to 2)
        - Output: True
13. **Find the Duplicate Number**
    - Problem: Given an array of integers where each integer is between 1 and n, find the duplicate number.
    - Example:
        - Input: $\text{nums} = [1, 3, 4, 2, 2]$
        - Output: 2
14. **Middle of the Linked List**
    - Problem: Given a linked list, find the middle node.
    - Example:
        - Input: $1 \text{ -> } 2 \text{ -> } 3 \text{ -> } 4 \text{ -> } 5$
        - Output: 3

---

4. Two Pointers with Sorting

These problems involve sorting the array first and then using two pointers.

15. **Intersection of Two Arrays**
    - Problem: Given two arrays, find their intersection.
    - Example:
        - Input: $\text{nums1} = [1, 2, 2, 1]$, $\text{nums2} = [2, 2]$
        - Output: $[2]$
16. **Merge Sorted Arrays**
    - Problem: Given two sorted arrays, merge them into a single sorted array.
    - Example:
        - Input: $\text{nums1} = [1, 2, 3]$, $\text{nums2} = [2, 5, 6]$
        - Output: $[1, 2, 2, 3, 5, 6]$
17. **Squares of a Sorted Array**
    - Problem: Given a sorted array of integers, return the squares of the numbers in sorted order.
    - Example:
        - Input: $\text{nums} = [-4, -1, 0, 3, 10]$

- Output: [0, 1, 9, 16, 100]

---

5. Two Pointers with Greedy Approach

These problems involve using two pointers with a greedy strategy to optimize the solution.

18. **Partition Labels**
    - Problem: Given a string, partition it into as many parts as possible so that each letter appears in at most one part.
    - Example:
        - Input: s = "abacdc"
        - Output: [3, 3] (partitions "aba" and "cdc")
19. **Boats to Save People**
    - Problem: Given an array of people's weights and a boat limit, find the minimum number of boats to save everyone.
    - Example:
        - Input: people = [1, 2, 3, 4], limit = 4
        - Output: 3 (boats [1, 3], [2], [4])
20. **Remove Duplicates from Sorted Array**
    - Problem: Given a sorted array, remove duplicates in-place and return the new length.
    - Example:
        - Input: nums = [1, 1, 2, 2, 3]
        - Output: 3 (array becomes [1, 2, 3])

---

6. Miscellaneous Two Pointer Problems

21. **Backspace String Compare**
    - Problem: Given two strings, check if they are equal after processing backspace characters.
    - Example:
        - Input: s = "ab#c", t = "ad#c"
        - Output: True (both become "ac")
22. **Sort Colors**
    - Problem: Given an array of 0s, 1s, and 2s, sort them in-place.
    - Example:
        - Input: nums = [2, 0, 2, 1, 1, 0]
        - Output: [0, 0, 1, 1, 2, 2]
23. **Longest Mountain in Array**

- Problem: Given an array of integers, find the length of the longest mountain (increasing then decreasing sequence).
- Example:
  - Input: $\text{nums} = [2, 1, 4, 7, 3, 2, 5]$
  - Output: 5 (mountain $[1, 4, 7, 3, 2]$)

24. **Subarray Product Less Than K**
    - Problem: Given an array of positive integers, find the number of subarrays with a product less than k.
    - Example:
      - Input: $\text{nums} = [10, 5, 2, 6], \ k = 100$
      - Output: 8

These variations demonstrate the flexibility of the two-pointer technique. Let me know if you'd like detailed explanations or solutions for any of these!