GRIND90: Week 1

Arrays, Two Pointers & Sliding Window

Weekly Goal: Master the Two Pointers and Sliding Window patterns. For every problem, solve it, then write down the time and space complexity and the specific pattern used.

Day 1: Monday - Introduction to Two Pointers

125. Valid Palindrome (Easy)

Key Idea: Use two pointers starting from the beginning and the end of the string. Move them inwards, skipping non-alphanumeric characters.

283. Move Zeroes (Easy)

Key Idea: Use two pointers starting at the same position. One pointer (`slow`) keeps track of where the next non-zero element should go, and the other (`fast`) scans the array.

167. Two Sum II - Input Array Is Sorted (Medium)

Key Idea: This is the classic "opposite ends" two-pointer problem. If the sum is too small, which pointer should you move to increase the sum? If it's too large?

Day 2: Tuesday - Advanced Two Pointers

15. 3Sum (Medium)

> **Key Idea:** Sort the array first. Then, iterate through the array with a main pointer (`i`), and for each element, use the two-pointer technique (from `i+1` to the end) to find the other two numbers that sum to `-nums[i]`. Remember to handle duplicates.

> ### 11. Container With Most Water (Medium)
> **Key Idea:** Start with pointers at the two widest points. The area is limited by the shorter of the two lines. To find a potentially larger area, you must move the pointer of the shorter line inwards.

> ### 75. Sort Colors (Medium)
> **Key Idea:** This is the Dutch National Flag problem. Use three pointers: `low`, `mid`, and `high`. Iterate with `mid` and swap elements based on whether they are 0, 1, or 2, moving `low` and `high` accordingly.

Day 3: Wednesday - Introduction to Sliding Window

> ### 121. Best Time to Buy and Sell Stock (Easy)
> **Key Idea:** Use two pointers, `left` (buy) and `right` (sell). If `prices[right]` is less than `prices[left]`, you've found a new, better day to buy, so move your `left` pointer to `right`.

> ### 643. Maximum Average Subarray I (Easy)
> **Key Idea:** This is a perfect example of a *fixed-size* sliding window. Calculate the sum of the first `k` elements. Then, slide the window one element at a time, subtracting the element that's leaving and adding the new element.

> ### 3. Longest Substring Without Repeating Characters (Medium)
> **Key Idea:** Use a sliding window with a hash set to keep track of characters currently inside the window. If you encounter a character that's already in the set, you must shrink the window

from the left until the duplicate is removed.

Day 4: Thursday - Dynamic Sliding Window

> **209. Minimum Size Subarray Sum (Medium)**
> **Key Idea:** Expand the window by adding elements from the right. When the window's sum meets the target, record the length and then shrink the window from the left until the condition is no longer met.

> **904. Fruit Into Baskets (Medium)**
> **Key Idea:** This is a sliding window problem that translates to "find the longest subarray with at most two distinct elements." Use a hash map to track the count of fruit types in your current window.

> **424. Longest Repeating Character Replacement (Medium)**
> **Key Idea:** The window is valid if `window_length - count_of_most_frequent_char <= k`. Maintain a frequency map of characters in your window. If the window becomes invalid, shrink it from the left.

Day 5: Friday - Review & Mixed Problems

> **567. Permutation in String (Medium)**
> **Pattern Hint:** This is a fixed-size sliding window problem. How can you efficiently check if the characters in your window match the character counts of the first string?

> **438. Find All Anagrams in a String (Medium)**
> **Pattern Hint:** Very similar to the problem above, but you need to record all occurrences. This reinforces the fixed-size window pattern with frequency maps.

> **76. Minimum Window Substring (Hard)**
> **Challenge Problem:** A classic, difficult sliding window problem. It

combines frequency maps and a dynamic window. Attempt it for at least an hour. If you get stuck, watch a detailed explanation and then code it yourself.

Day 6: Saturday - Pattern Review & Consolidation

18. 4Sum (Medium)
Key Idea: A natural extension of 3Sum. Sort the array, then use nested loops to fix the first two numbers and apply the two-pointer technique on the rest of the array. Pay close attention to avoiding duplicate quadruplets.

1004. Max Consecutive Ones III (Medium)
Key Idea: A classic sliding window problem. The window is valid if the number of zeros within it is less than or equal to `k`. Expand the window to the right, and if it becomes invalid, shrink it from the left.

1423. Maximum Points You Can Obtain from Cards (Medium)
Key Idea: This is a clever problem that can be reframed. Instead of taking `k` cards from the ends, think about finding a subarray of size `n-k` with the minimum sum. The maximum score will be the `total_sum - min_sum_of_subarray`.

Sunday Plan

Active Rest. Review notes from the week. Verbally explain each problem and solution to solidify your understanding. Do not solve new problems today.