Three Sum - Notes

Problem Statement:

Given an integer array nums, return all the unique triplets [nums[i], nums[j], nums[k]] such that:

- i != j, i != k, and j != k

- nums[i] + nums[j] + nums[k] == 0

Approach 1: Brute Force

- Explanation:

Iterate through each triplet using three nested loops to find the sum of three elements that equal zero.

- Time Complexity: O(n^3)

- Space Complexity: O(1)

Approach 2: Sorting + Two Pointers

- Explanation:

Sort the array, then for each element nums[i], use two pointers to find the other two elements that sum to z

- Fix one element, and use two pointers to find the other two elements.

- Time Complexity: O(n^2)

- Space Complexity: O(1) (excluding the space used by the input array)

Approach 3: HashMap

- Explanation:

For each number in the array, calculate the complement (0 - nums[i]), and check if the complement exists u

- Time Complexity: O(n^2)

- Space Complexity: O(n)

Corner / Edge Cases Handled:

- Duplicate values (ensure unique triplets by skipping over duplicates after sorting).

- Array with fewer than three elements (no valid triplet).

- Handling negative numbers.

- Target sum of zero (all solutions should sum to zero).