Majority Element (n/3)

Problem Statement:

Given an integer array nums, return all elements that appear more than floor(n/3) times.

You may return the answer in any order.

Brute Force Approach:

- Idea: For each element, count its frequency by looping through the entire array.
- Implementation: Use nested loops.
- Time Complexity: O(n^2)
- Space Complexity: O(1)
- Note: Not optimal for large arrays.

Approach 1: Using Hash Map / Frequency Count

- Idea: Use a hash map (or unordered_map in C++) to store frequencies.
- Steps:
  1. Traverse the array and count the frequency of each element.
  2. Collect elements with frequency > floor(n/3).
- Time Complexity: O(n)
- Space Complexity: O(n)

Approach 2: Boyer-Moore Voting Algorithm (Optimized)

- Idea: There can be at most two majority elements more than n/3.
- Phase 1: Candidate selection
  - Maintain two candidate variables and counts.
  - Traverse and update based on matching, zero counts, or decrement.
- Phase 2: Count verification
  - Recount candidates to verify if they occur more than floor(n/3) times.

- Time Complexity: O(n)

- Space Complexity: O(1)

Edge Cases:

1. Empty array

2. All elements same (e.g., [1,1,1,1])

3. No element appears > n/3 times (e.g., [1,2,3,4,5])

4. Multiple valid majority elements

5. Array with only one or two elements