Majority Element Problem (Boyer–Moore & HashMap Solutions)

Problem Statement: Given an array of size n, find the element that appears more than ■n/2■ times. If no such element exists, return -1.

■ Approach 1: Boyer–Moore Voting Algorithm - Idea: Keep track of a potential candidate and its frequency. - If count becomes 0, change the candidate. - At the end, verify if the candidate is the majority element. - Time Complexity: O(n) - Space Complexity: O(1)

```
class Solution:
    def majorityElement(self, arr):
        n = len(arr)
        count = 0
        element = arr[0]

        for num in arr:
            if count == 0:
                element = num
                count = 1
            elif num == element:
                count += 1
            else:
                count -= 1

        # Verify candidate
        count = 0
        for num in arr:
            if num == element:
                count += 1

        if count > n // 2:
            return element
        return -1
```

■ Approach 2: HashMap (Dictionary in Python) - Count occurrences of each element using a dictionary. - If an element's frequency > n/2, return it. - If no element satisfies the condition, return -1. - Time Complexity: O(n) - Space Complexity: O(n)

```
class Solution:
    def majorityElement(self, arr):
        n = len(arr)
        freq = {}

        for num in arr:
            freq[num] = freq.get(num, 0) + 1
            if freq[num] > n // 2:
                return num

        return -1
```