

3. Majority Element

MEDIUM

■ What You Will Learn

- ✓ Boyer-Moore Voting Algorithm — $O(n)$ time, $O(1)$ space
- ✓ Why it works: majority cancels all minorities
- ✓ How to verify the candidate after voting

■ Intuition First

Imagine a battle royale where each number fights. When two different numbers meet, they cancel each other out. The number that appears more than $n/2$ times will always have survivors left — because there simply aren't enough opponents to cancel all of them.

■ Boyer-Moore Voting

- candidate = current majority candidate
- count = vote count for candidate
- See same number: count++
- See different: count-- (one cancels one)
- When count=0: switch candidate to current element

■ Visual Trace on [3,2,3,1,3]

3 appears 3 times ($n/2=2.5$) — majority is 3

3	2	3	1	3
---	---	---	---	---

```
arr[0]=3: candidate=3, count=1
arr[1]=2: 2≠3, count=0 → candidate still 3 (count hit 0)
arr[2]=3: candidate=3, count=1 (reset with count=0)
arr[3]=1: 1≠3, count=0 → candidate=3, count=0
arr[4]=3: candidate=3, count=1
Final candidate = 3 ✓
```

■ Java Code

```
public static int majorityElement(int[] arr) {
    int candidate = arr[0], count = 1;
```

```
// Phase 1: Find candidate
for (int i = 1; i < arr.length; i++) {
    if (arr[i] == candidate) {
        count++;
    } else {
        count--;
        if (count == 0) {
            candidate = arr[i];
            count = 1;
        }
    }
}

// Phase 2: Verify (if majority guaranteed, skip this)
int freq = 0;
for (int x : arr) if (x == candidate) freq++;
return freq > arr.length / 2 ? candidate : -1;
}
```

■ Complexity

Case	Time	Space	What happens?
All Cases	O(n)	O(1)	Two passes at most — still O(n)

■ Common Mistakes

- Boyer-Moore only finds the CANDIDATE — verify it in a second pass if majority isn't guaranteed
- Problem says majority exists → Phase 2 verification optional, but safe to include

■ Why Does It Work?

- Majority element appears $> n/2$ times
- All other elements combined appear $< n/2$ times
- So majority can cancel at most $n/2$ opponents and still have votes left
- It always survives as the final candidate

■ Practice Problems

[Med] Find majority without Boyer-Moore — using HashMap

[Med] Find all elements appearing more than $n/3$ times

[Hard] Find majority in a stream of elements

■ Quick Revision

- Boyer-Moore: candidate+count approach
- Same: count++. Different: count-- . Zero: switch candidate
- Two-pass: find candidate, then verify
- $O(n)$ time, $O(1)$ space
- Works because majority $> n/2$, outnumbers all opponents combined