MOORE'S VOTING ALGORITHM

☆ In this problem, we are supposed to find the majority element, i.e. element which occurs > N/2 times

Brute force solution is to use two for loops, keep track of each element. Better solution involves the use of hashmap to improve time complexity but also adds to space complexity

Optimal Solution is given by Moore's Voting Algorithm. In Moore's Voting Algorithm, we iterate through the array and maintain a count. Whenever the element occurs we add to count and whenever it doesn't we subtract. When count hits 0, we restart this process with the next element implying that the previous one got cancelled out.
At the end, if we have a non zero count, we conclude that if a majority exists it is this one else it doesnt.

Pseudocode :

```
majorityElement(arr, N) {
    c = 0
    el = -1
    for(int i = 0 → N) {
        if(c == 0) {
            c = 1
            el = arr[i]
        } else if(arr[i] == el) {
            c++
        } else {
            c--
        }
    }
    if(c == 0) return -1
    if(c != 0) {
        c = 0
        for(int i = 0 → N) {
            if(arr[i] == el) {
                c++
            }
        }
        if(c > N/2) {
            return el
        } else return -1
    }
}
```