# 169. Majority Element : Moore's voting algorithm

Given an array `nums` of size `n`, return *the majority element*.

The majority element is the element that appears more than `⌊n / 2⌋` times. You may assume that the majority element always exists in the array.

**Example 1:**

```
Input: nums = [3,2,3]
Output: 3
```

**Example 2:**

```
Input: nums = [2,2,1,1,1,2,2]
Output: 2
```

**Constraints:**

- `n == nums.length`
- `1 <= n <= 5 * 10<sup>4</sup>`
- `-2<sup>31</sup> <= nums[i] <= 2<sup>31</sup> - 1`

**Follow-up:** Could you solve the problem in linear time and in `O(1)` space?

```
class Solution:
    def majorityElement(self, nums: List[int]) -> int:
        val = nums[0]
        count = 1
        for i in range(1,len(nums)):
            if val==nums[i]:
                count+=1
            else:
                count-=1
            if count==0:
                val = nums[i]
                count+=1
        # ref = len(nums)//2
        # ans = 0
        # for ele in nums:
        #     if ele==val:
```

```
        #          ans+=1
        # return val if ans>ref else
        return val
```