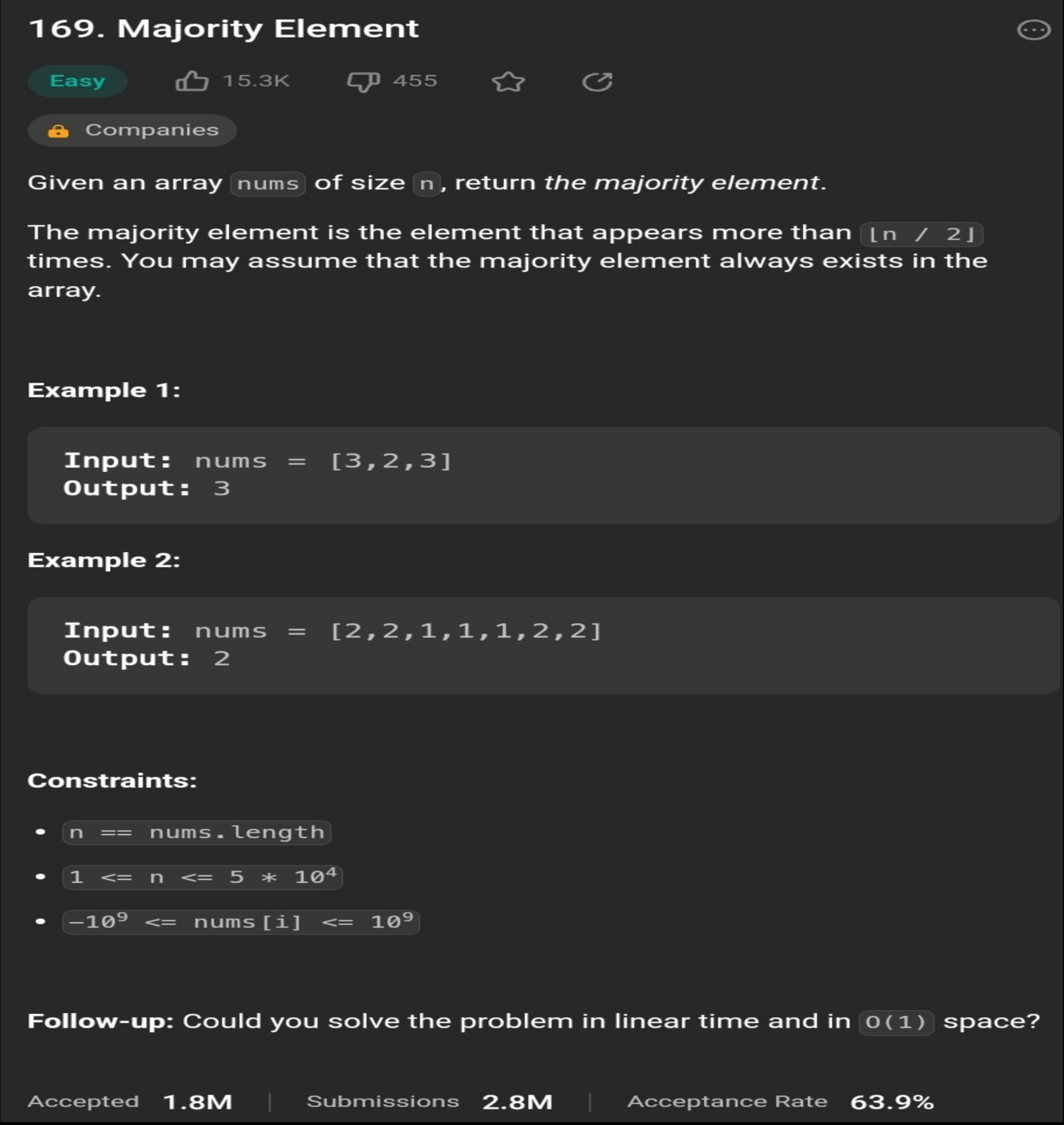


169. Majority Element

Easy 15.3K 455

Companies

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`
- $1 <= n <= 5 * 10^4$
- $-10^9 <= nums[i] <= 10^9$

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

Accepted 1.8M | Submissions 2.8M | Acceptance Rate 63.9%

Approach 1:

Using sorting.

→ Sort the array.

→ now return nums[n/2].

$O(n\log n)$

Approach 2:

Approach 2:

Using hash map

→ Traverse the array and store the frequency of elements i.e. keys in an unordered_map.

→ now return the key for which the it→second > n/2.

T: O(n)
S: O(n)

Approach 3:

Moore's voting algorithm

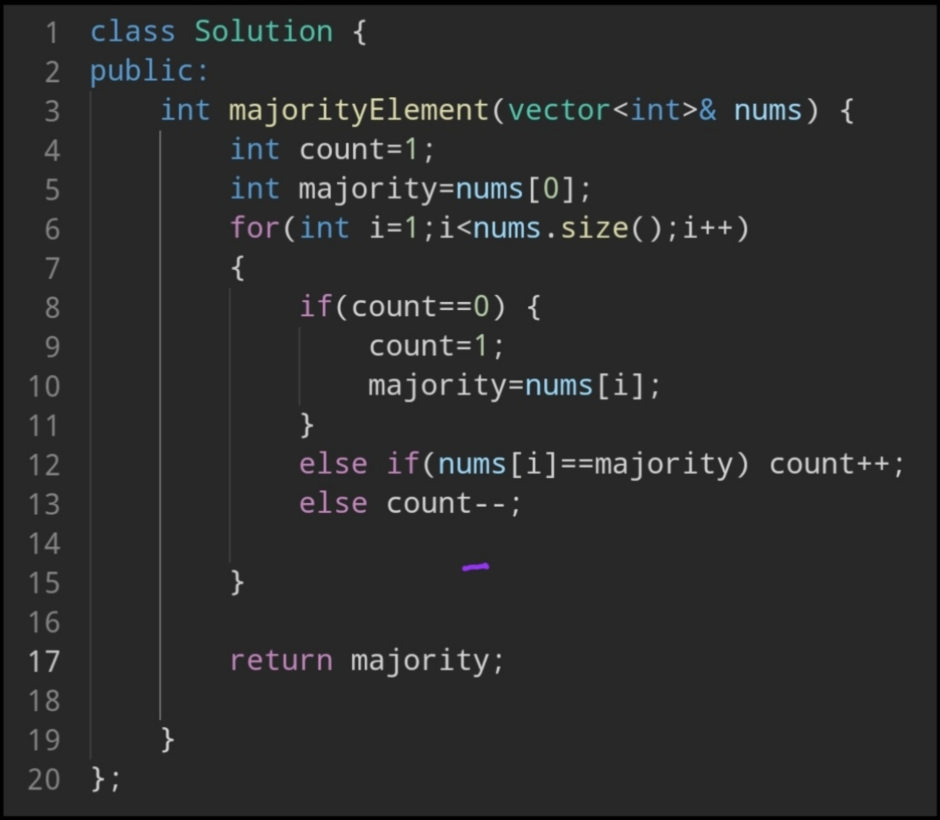
As we are guaranteed that, there is always an element in the array which occurs for more than n/2 times, we use voting algo.

0	1	2	3	4	5	6	7	8	9
9	5	6	9	9	5	9	8	9	9

count = 1 majority = 9
nums[1] ≠ majority So count--
count = 0 majority = 9
As count = 0, change majority = 5
Count = 1 majority = 5
nums[2] ≠ majority So count--
Count = 0 majority = 5
As count = 0, change majority = 6
count = 1 majority = 6
nums[3] ≠ majority So count--
count = 0 majority = 6
As count = 0, change majority = 9
count = 1 majority = 9

nums[4] = majority So count++
count = 2 majority = 9
nums[5] ≠ majority So count--
count = 1 majority = 9
nums[6] = majority So count++
Count = 2 majority = 9
nums[7] ≠ majority So count--
count = 1 majority = 9
nums[8] = majority So count++
count = 2 majority = 9
nums[9] = majority So count++
count = 3 majority = 9

So return majority = 9



```cpp
class Solution {
public:
    int majorityElement(vector<int>& nums) {
        int count=1;
        int majority=nums[0];
        for(int i=1;i<nums.size();i++)
        {
            if(count==0) {
                count=1;
                majority=nums[i];
            }
            else if(nums[i]==majority) count++;
            else count--;

        }

        return majority;

    }
};
```

T: O(n)
S: O(1)