MAJORITY

NO. 169

MAJORITY ELEMENT

PROBLEM DESCRIPTION

- Given an array of size n, find the majority element. The majority element is the element that appears more than ⌊ n/2 ⌋ times.
- You may assume that the array is non-empty and the majority element always exist in the array.

IDEAS

- Sorting
- Use reference count
- Time complexity O(n)
- if incoming number equal to majority, ref count increases by 1
- Otherwise, ref count decreases by 1
- if ref count < 0, then change majority number

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/169_Majority_Element

NO. 229

MAJORITY ELEMENT II

PROBLEM DESCRIPTION

- Given an integer array of size n, find all elements that appear more than ⌊ n/3 ⌋ times. The algorithm should run in linear time and in O(1) space.

IDEAS

- Calculate the target reference count
  - Use “set” to remove the redundant number
  - Use Python built-in function “count”
    - If count is greater than target reference count, add to return list
- **Performance sucks!!**

IDEAS

- “count” could be an expensive operation
  - How to reduce the calling times of “count” ?
- There are at most 2 major numbers
  - Because major number needs to appear at least ⌊ n/3 ⌋ times
- Create major1, major2, ref_count1, ref_count2, and start voting!

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/229_Majority_Element_II