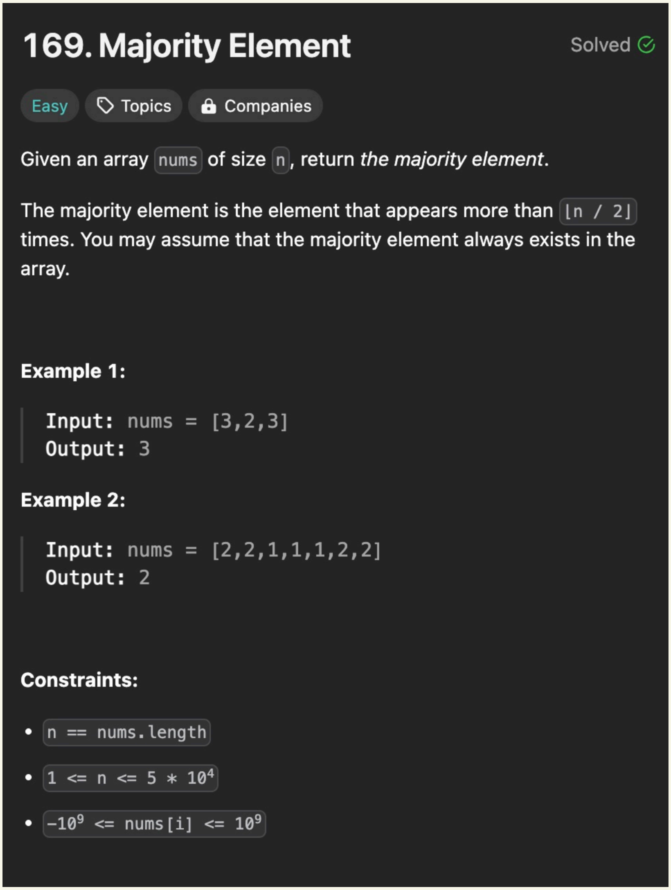

169. Majority Element

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

1) Easy Solution

→ Create a hashmap for the counts

→ Iterate over nums

→ If num is in hashmap, increase its count by 1.

→ If num is not in hashmap, add it (with a count of 1).

Here's the hashmap after iterating over nums.

i = 0 1 2 3 4 5 6

nums = [2, 2, 1, 1, 1, 2, 2]

counts = { } // empty.

i = 0

counts = { 2:1 }

i = 1

counts = { 2:2 }

i = 2

counts = { 2:2, 1:1 }

i = 3

counts = { 2:2, 1:2 }

i = 4

counts = { 2:2, 1:3 }

i = 5

counts = { 2:3, 1:3 }

i = 5

counts = { 2:4, 1:3 }

after for-loop, hashmap counts contains all nums and its counts

Final step is iterating over hashmap keys (1, 2) and getting the max count.

2) Ingenuous Solution. - Moore's Algorithm.

```
→ count = 0
→ majorityElement = 0

Iterate x over nums :
        if count == 0
            majority Element = x

        if x == majorityElement :
            count ++;
        else:
            count --;

    return majorityElement.
```

Example:

nums = [2, 2, 1, 1, 1, 2, 2]

↑1 ↑2 ↑3 ↑4 ↑5 ↑6 ↑7

count = ~~0~~ ~~1~~ ~~2~~ ~~1~~ ~~0~~ ~~1~~ ~~0~~ 1.

majEl = ~~0~~ ~~2~~ ~~1~~ 2