Majority Element - 1

A majority Element is

↳ A element with frequency $> \frac{N}{2}$ → where 'N' is size of array

arr[6] : {1 2 1 6 1 1} ⇒ frequency of '1' = $4 > \frac{6}{2}$

So, 1 is a majority Element

arr[9] : {3 4 4 8 4 9 4 3 4} ⇒ frequency of '4' = $5 > \frac{9}{2}$

So, '4' is a majority Element

Note : There always exists a majority Element, Code accordingly

//idea - 1 → Use nested loops , TC : $O(n^2)$

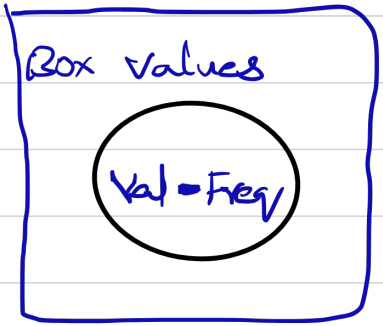
//idea - 2 → Use Hashmap to Count , TC : $O(n)$ SC : $O(n)$

//optimal approach

Boyer Moore's Voting Algorithm

If we cancel out 2 distinct Elements one by one, the Uncancelled (or) remaining Element is going to be Our majority Element

Ex: arr[11] : { 3 3 4 6 1 3 2 5 3 3 3 }
(indices: 0 1 2 3 4 5 6 7 8 9 10)



i=0 → 3-1

i=1 → 3-2

i=2 → 3-1 (Cancel one 3 with 4)

i=3 → 3-0 (Cancel one 3 with 6)

i=4 → 1-1

Cancel one 1 with 3

i=5 → 1-0

i=6 → 2-1

i=7 → 2-0 (Cancel one 2 with 5)

i=8 → 3-1

i=9 → 3-2

i=10 → 3-3

```java
class Solution {
    public int majorityElement(int[] nums) {
        int n = nums.length;
        int freq = 1;
        int val = nums[0];

        for(int i = 1;i<n;i++){
            if(freq == 0){
                val = nums[i];
                freq = 1;
            }
            else if(val == nums[i])freq++;
            else freq--;
        }
        return val;
    }
}
```

i=0

See,
1st Box filled with 0th idx val & freq = 1

if freq → 0
then we have to assign new majority element

if opposite elements cancel out by reducing frequency

for understanding See Box ③ & ④

From Box-3, the freq of 3 became '0', so in Box ④ we assigned 1 as our new majority Element with freq '1'