Majority Element - I

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) → freq of `1` = $\left(4 > \frac{6}{2}\right)$

So, 1 is a majority Element

arr[9] : (3, 4, 4, 8, 4, 9, 4, 3, 4) → freq of `4` = $\left(5 > \frac{9}{2}\right)$

So, 4 is a majority Element

Note :

There always exists a majority element, so code accordingly

//idea-1

use nested loops : $O(n^2)$

//idea-2

use hashmap to Count : Tc : $O(n)$, SC : $O(n)$

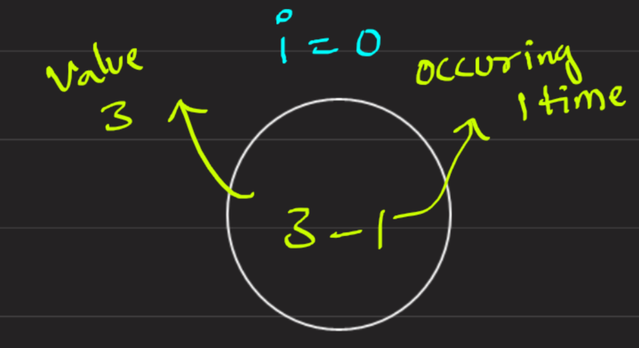
As Time Complexity → $O(n)$ is good, but we have an algorithm that can reduce space to $O(1)$

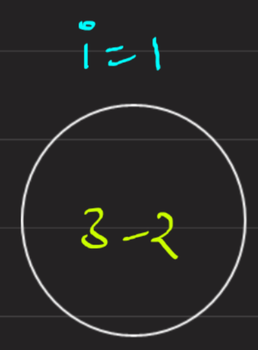
Boyer Moore's Voting Algorithm

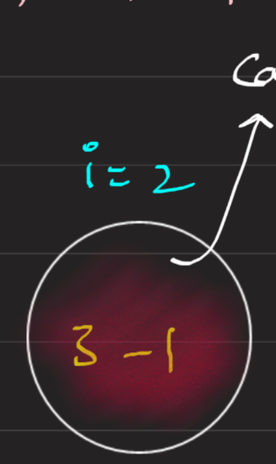
If we cancel out 2 distinct elements one by one, the uncancelled (or) remaining element is going to be our majority element

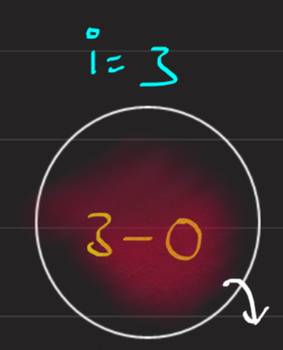
Ex : arr[11] = (3, 3, 4, 6, 1, 3, 2, 5, 3, 3, 3)
0 1 2 3 4 5 6 7 8 9 10

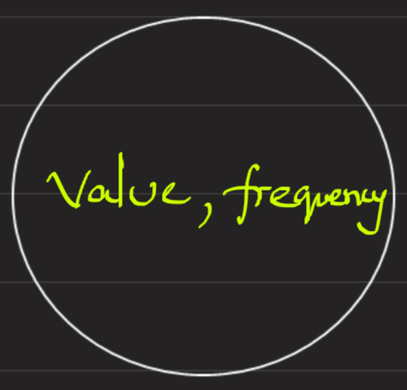
if a value that is different from Box value, we reduce box frequency Count

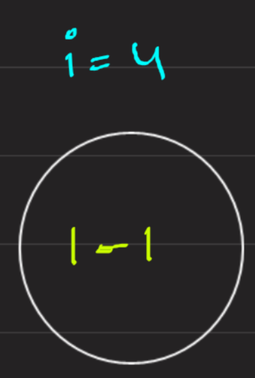


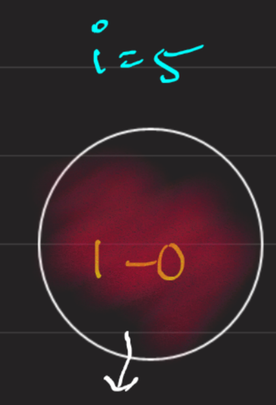


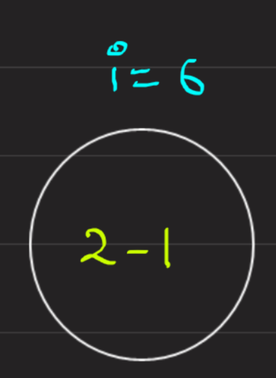


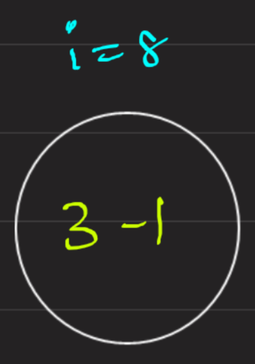


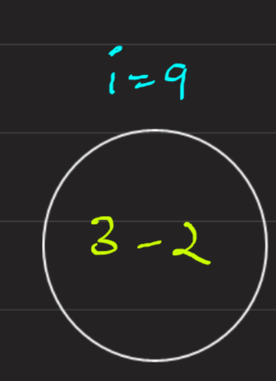


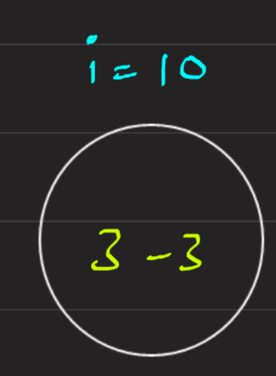













```java
public int majorityElement(int[] nums) {
    int n = nums.length;
    int freq = 1;
    int val = nums[0];

    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;
}
```

} → At start, whatever is there in 0th index, store it

} → whenever freq goes to 0, update with new value,

→ if same as box value comes, increase freq

if different value comes, decrement count (distinct from box value)

* for understand, see at (i==3) & (i==4)

from (i==3) → the freq become '0', so in (i==4) we assigned 1 as our new majority element with freq = 1