Majority Element - II

A majority element is, having a (frequency > $\frac{N}{3}$)

N = length of array

* We use same approach - "Boyers Moore Voting Algo"

But with additional components

* previously if two distinct elements occured, we cancelled them
* Now, if three distinct elements occurs, we will cancel them

Lets split the solution into 2 phases:

1. Identifying potential candidates
2. Verifying the candidates

① Identifying potential Candidates

* Use 2 variables to store potential candidates & their counts

* Traverse the array,
  - If one counter is zero, Update it with current element and set freq to 1
  - If current element matches with one of the candidates, increment its count
  - Otherwise, decrement the count

② Verifying the Candidates

* Initialise counts for the two candidates
* Traverse array, to count every candidate occurance
* Check if any candidates count is $(> \frac{n}{3})$

Ex:

0	1	2	3	4	5	6	7	8	9	10
2	3	2	4	2	3	5	3	3	2	4

Candidate -I

Candidate -II

⟹ Candidates Box

if we observe 3 Candidates having 3 distinct values, So reduce count of Both

Candidate - 3 ⇒ 4 arr[i]

i=0

Val = 2
Count = 1

Val = 2
Count = 0

just set val as nums[0] dont increase count

i=1

Val = 2
Count = 1

Val = 3
Count = 1

i=2

Val = 2
Count = 2

Val = 3
Count = 1

i=3

Val = 2
~~Count = 1~~

Val = 3
~~Count = 0~~

i=4

Val = 2
Count = 2

Val = 3
Count = 0

i=5

Val = 2
Count = 2

Val = 3
Count = 1

i=6

Val = 2
~~Count = 1~~

Val = 3
~~Count = 0~~

i=7

Val = 2
Count = 1

Val = 3
Count = 1

i=8

Val = 2
Count = 1

Val = 3
Count = 2

i=9

Val = 2
Count = 2

Val = 3
Count = 2

i=10

Val = 2
~~Count = 1~~

Val = 3
~~Count = 1~~

* At last, for value = 2, 3 ⇒ we are having counts > 0

So, these are possible candidates for majority elements

Hence, travel through the array, and check whether {2, 3} are appearing ($> \frac{n}{3}$) times or not,

which ever element from {2, 3} appears ($> \frac{n}{3}$) times that element is a majority element

```java
class Solution {
    public List<Integer> majorityElement(int[] nums) {
        List<Integer> ans = new ArrayList<>();
        int n = nums.length;

        int val1 = nums[0];
        int cnt1 = 1;
        int val2 = nums[0];
        int cnt2 = 0;

        for(int i = 1;i<n;i++){
            if(nums[i] == val1)cnt1++;
            else if(nums[i] == val2)cnt2++;
            else if(cnt1 == 0){
                val1 = nums[i];
                cnt1++;
            }
            else if(cnt2 == 0){
                val2 = nums[i];
                cnt2++;
            }
            else if(val1 != nums[i] && val2 != nums[i]){
                cnt1--;
                cnt2--;
            }
        }
```

Phase - I

```java
        int candidate1 = 0;
        int candidate2 = 0;

        for(int i = 0;i<n;i++){
            if(val1 == nums[i])candidate1++;
            else if(val2 == nums[i])candidate2++;
        }
        if(candidate1 > n/3){
            ans.add(val1);
        }
        if(candidate2 > n/3){
            ans.add(val2);
        }

        return ans;
    }
}
```

Phase - II