Find Majority element > n/3

Problem Statement Suggest Edit

Given an array 'A' of 'N' integers, find the majority elements of the array.
A majority element in an array 'A' of size 'N' is an element that appears more than floor(N / 3) times.
Note: The floor function returns the number rounded down to the nearest integer.
Note: Return the array of majority elements in sorted order.
Example:

```
Input: 'N' = 9 'A' = [2, 2, 1, 3, 1, 1, 3, 1, 1]

Output: 1

Explanation: The frequency of '1' is 5, which is greater than floor(N
/ 3), hence '1' is the majority element.
```

Explanation of Question:

element find and store arr[] = { }

freq. > n/3

Eg:

arr[] = { 1, 1, 1, 2, 2, 2 }

n = 6

n/3 = 6/3 = 2 → freq ≥ 2

arr[] = { 1 1 1, 2 2 2 }

1 → 3 > 2

2 → 3 > 2

arr[] = { 1, 2 }

sorted Sorted order

Observation:

$n = 8 \rightarrow \left\lceil \frac{8}{3} \right\rceil = \underline{2}$

At max how many element can be there in an array?

$3 > 2$ 3

- At minimal a element would have 3 occurence so, if $3+3+3 = \underline{9} >$ size of array.
- We to assume at 3 element of 3 occurence the size of the array is 9 but it is greater than orginal size of array
- So, At max we'll have only 2 element.

eg: $99 = \left\lceil \frac{99}{3} \right\rceil = \underline{33}$

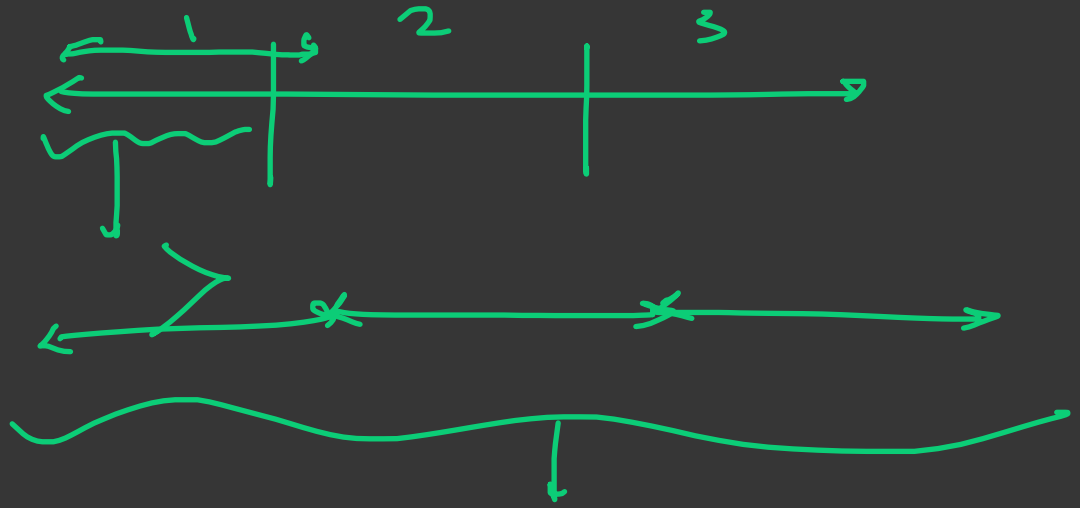
$34+34 = 68$

34

$102 > 99$ → X

Intuition behind it:

We have said that the occurrence should be more than 1/3rd of array size n. = n/3



more than 1/3rd

$\frac{1}{3} + \frac{1}{3}$

$\frac{2}{3}$

$\frac{1}{3} = 0.\overline{33}$

$= 0.43$

0.33
0.33
0.33
1

1
0.4
0.4
0.4
1.2
1

Brute force:

arr[] : { 1, 1, 1, 2, 2, 2 }

n = 6

count freq. → ans push

↳ Intuition

ans[2] : { _ , }

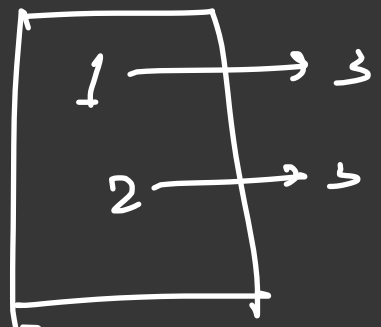
```
for (i=0, i<n, i++)
    int count=0;
    if (ans[0] != arr[i] || ans.size()==0)
        for(j=0 ; j<n, j++)
            if(arr[i]==arr[j]) count++;
        if( count > n/3 ) ans.push
```

T.C. = O(N²)

S.C. = O(1)

Better Approach:

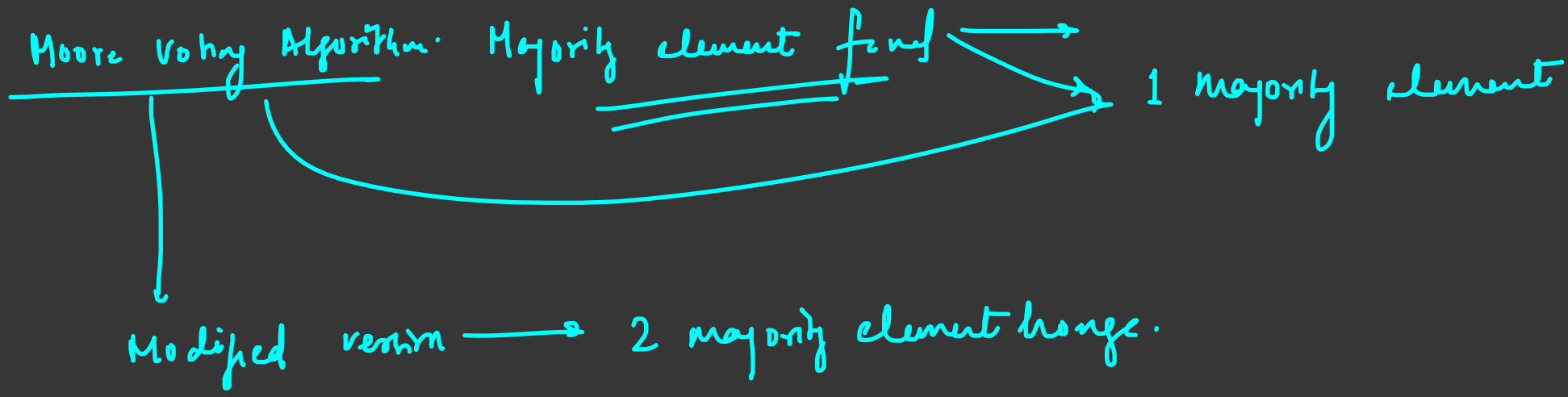
Using hashmaps to store freq




```
for (auto it: hashmap)
    if(it.second > n/3)
        ans.push_back(it.first);

return ans;
```

Optimal Approach:



So, we know that we can have only 2 elements which can be more than n/3 times.

So, Idea is to find 2 majority element in the array and check if the 2 majority elements are greater than n/3 time.

Pseudo code:

ele1, ele2, count1 = 0, count2 = 0.

for (i = 0, i < n, i++)

if(count1 == 0 && arr[i] != ele2)

ele1 = arr[i],

count1 = 1

Similar for ele2.

} Moore Voting Algo

To avoid overlapping of majority element.

Cont

T.C. :→ $O(N) + O(N) = O(2N) \approx O(N)$

S.C. : → $O(1)$ →