Question: https://leetcode.com/problems/majority-element/

So we need to find the element that appears more than n / 2 times.

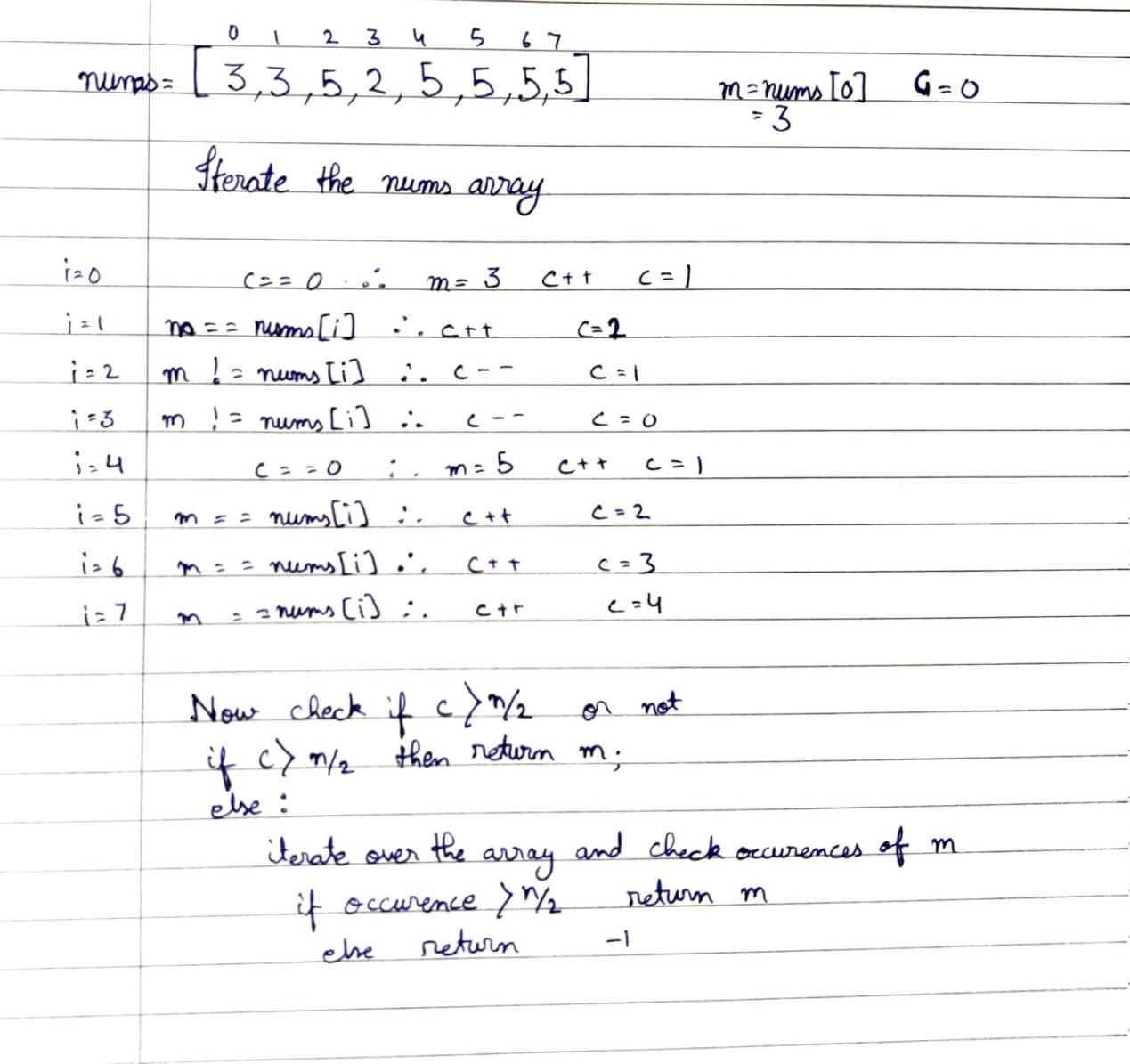
Initially what we can do is store each element as key and their occurrences as value in a Hash Map.

Then iterate over the Hash Map and look for the pair having value more than n/2.

If found then return the key else return -1.

Now this approach has a O(n) time complexity and O(n) space complexity.

But we can further improve its space complexity to O(1), by using Bayer Moore’s Voting Algorithm.



Solution:  
class Solution {

public int majorityElement(int[] nums) {

int m=nums[0], c=0;

for(int e: nums){

if(c==0){

m=e;

c++;

}else{

if(m==e)

c++;

else

c--;

}

}

if(c>(nums.length/2)){

return m;

}else{

int occur=0;

for(int el: nums){

if(m==el) occur++;

}

if(occur>(nums.length/2)){

return m;

}else{

return -1;

}

}

}

}

Github Link :<https://lnkd.in/ecwtJeaz>