MOORE'S VOTING ALGORITHM

Think of this array as a collection of votes from the voters for different political parties.
Now as we know the party which has >50% votes can form the government.

So , our above question is analogous to this situation.
Now , if we are certain that one party has > 50% votes . Then , if :

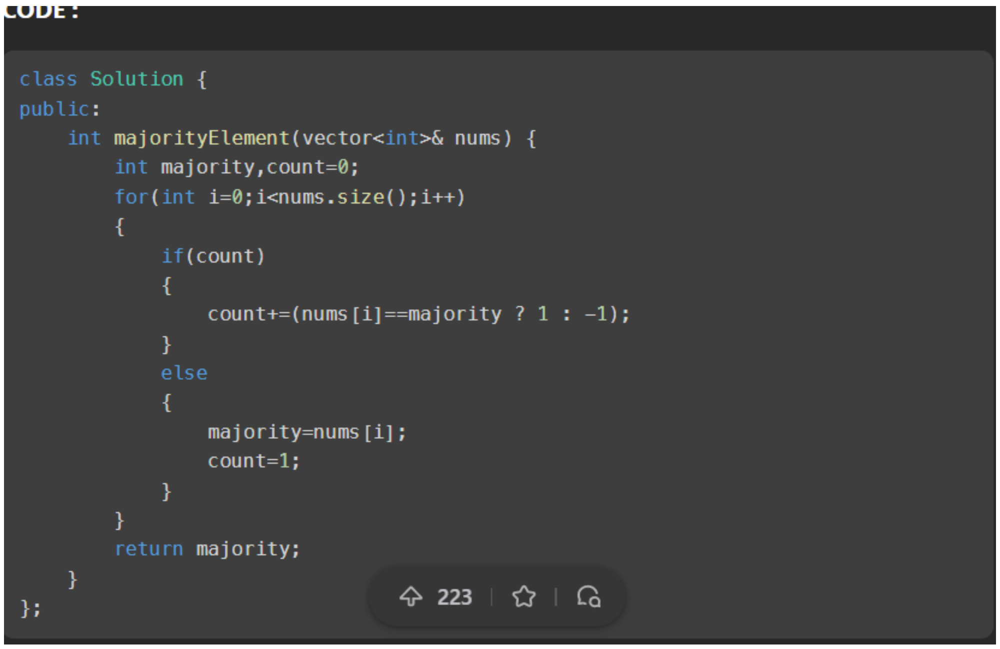
- We **increment a count variable every time we see the vote from the majority party** and **decrement it whenever a vote from some other party is occured** , we can guarantee that ,
  count>0.

Using the above logic ,

1. Create a count=0 and a majority variable that stores the current majority element.
2. Traverse the array , and if count =0 , then store the current element as the majority element and increment the count.
3. Else , if the current element is equal to the current majority element , increment count , else decrement it.
4. At the end return the majority element.

This approach works because ,

1. We simply know that the majority element has a frequency greater than half of the total elements.
2. So , the value of count > 0.
3. But whenever it becomes 0 , it means that till now the majority element has either not occured , or if it has then the its frequency is equal to the sum of frequency of rest of the others. So , the next element will be the majority element till now . But eventually by the end , the final answer will always be the majority element of the array , as when the "count" becomes 0 for the final time , the next element will be the majority element.


CODE:

```cpp
class Solution {
public:
    int majorityElement(vector<int>& nums) {
        int majority,count=0;
        for(int i=0;i<nums.size();i++)
        {
            if(count)
            {
                count+=(nums[i]==majority ? 1 : -1);
            }
            else
            {
                majority=nums[i];
                count=1;
            }
        }
        return majority;
    }
};
```