Majority Element-11 , N = length of array A majority Element is, having a frequency > N 3 p We use same approach - "Boyers Moore Voting Algo" But with additional components * previously if two distinct Elements Occurred, we Cancelled them * Now, if three distinct Elements Occurs, we will ancel them Lete Split the Solution into 2 phases: 1) Identifying potential Candidates @ Verifying the Candidates

1) Identifying potential Candidates

(2) Use a variables to Store potential Condidates & their Counts

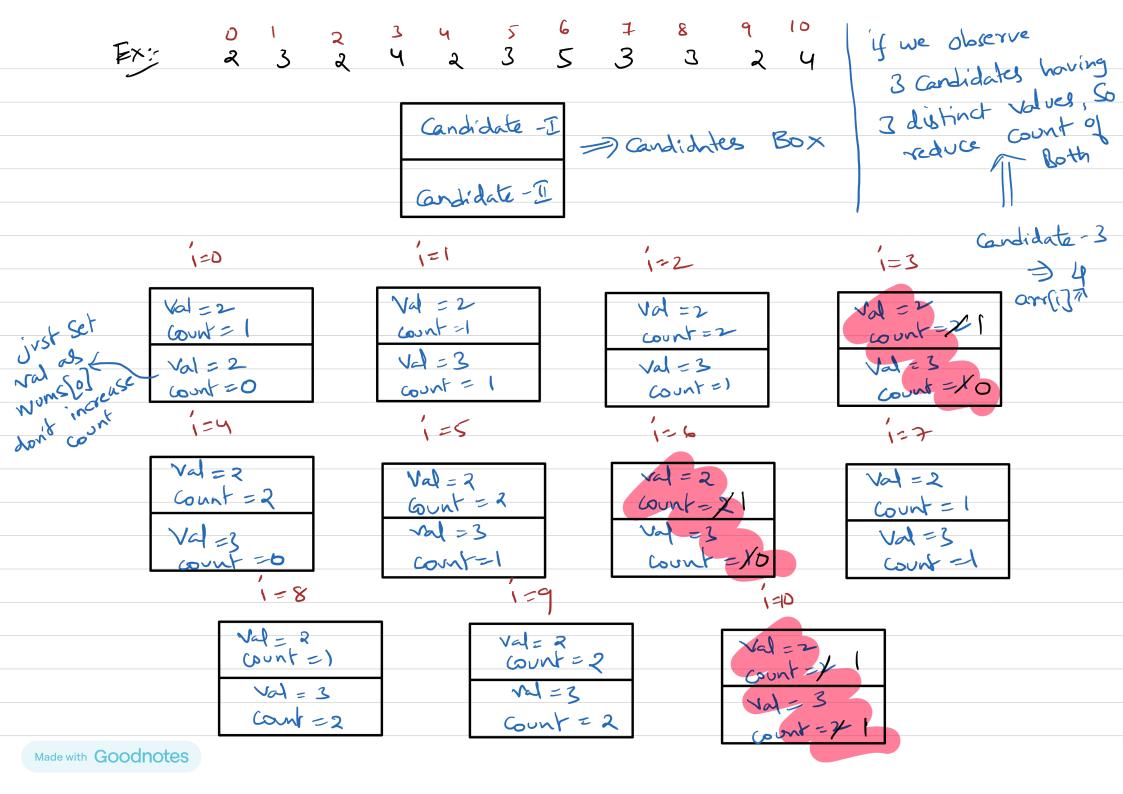
- * If one counter is zero, Update it with current Element and Set frey to 1
 - · If current element matches with one of the Candidates, increment its count
 - · Otherwise, decrement the count

2 Verifying the Candidates

* Initialise counts for the two andidates

* Traverse array, to count every condidate occurance

* Check if any condidates count is (>2)



```
* At last, for valve = 2,3 \Rightarrow we are having counts >0

So, these are possible Candidates for majority Elements

thence, travel through the array, and check whether \{2,3\} are

appearing (72) times or not,
```

which ever element from 22,34 appears (27) times that element is a majority Element

```
public List<Integer> majorityElement(int[] nums) {
    List<Integer> ans = new ArrayList<>();
    int n = nums.length;
                                         phase I
    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];
        else if(cnt2 == 0){
           val2 = nums[i];
           cnt2++;
        else if(val1 != nums[i] && val2 != nums[i]){
Goodnotes
```

```
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;
}
```