SCALER

Interview Problems

TABLE OF CONTENTS

< Question > : Given a binary array [ ]. We can almost replace a single 0 with 1. Find the maximum consecutive 1's we can get in the array[ ] after the replacement.

$1 \leq N \leq 10^3$

Example : [1 1 0 1 1 0 1 1 1] ans = 6

Example : [0 1 1 1 0 1 1 0 1 1 0] ans = 6

Example : [1 1 1 1 1] ans = 5

Example : [0 0 0 0 0] ans = 1

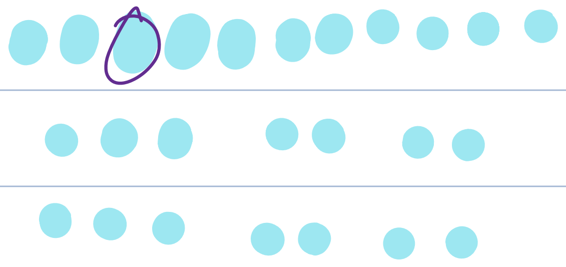
$$\underbrace{\qquad\qquad}_{l}\ 0\ \underbrace{\qquad\qquad}_{r}$$

[BF Idea] - For each 0, I counted how many 1's on the left, 1's on the right.

$l \quad r \quad \Rightarrow \quad l + r + 1$

[0 1 1 1 0 1 1 0 1 1 0]

0 1 2 3 4 5 6 7 8 9 10



for any index max how many visit $\Rightarrow 3$

$\underline{1} \rightarrow 3$ visits

$N \Rightarrow 3N$ visits

TC: $O(N)$

< / > *Code*

```
1)    Count the number of ones
           if ones == arr.length
              ans = arr.length
   ans = 0
   for ( i : 0 -> n-1 ) {
         if( nums[i] == 1 )
               Continue
         else {
              l=0        r=0
              j = i+1
              while ( j<n && nums[j] ==1 ){
                     r++
                     j++
              }
            j =  i-1
            while (j>=0 && nums[j]==1) {
                     l++
                     j--
            }
```

ans = max(ans, l+r+1)

}

< Question > : Given a binary array [ ]. We can swap a single 0 with 1. Find the maximum consecutive 1's we can get in the array[ ] after almost 1 swap.

1 $\overset{}{\cancel{0}}$ 1 1 0 $\cancel{1}$ l+r+1

    1   0

1 1 $\cancel{0}$ 1 $\cancel{1}$ l+r = Ones

    1   0

arr[ ] → [1 1 0 1 1 1 0 1]

1 1 $\cancel{0}$ 1 1 0 $\cancel{1}$

    1   0

arr[ ] → [0 1 1 0 1 1 1 0]

0 1 2 3 4 5 6 7

$1\ 1\ 1\ \underset{1}{\cancel{0}}\ 1\ \underset{0}{\cancel{1}}\ 0\ 0\ 0$

</> Code

```
1)   Count the number of ones
          if ones == arr.length
               ans = arr.length
   ans = 0
   for ( i : 0 -> n-1 ) {
          if( nums[i] == 1)
                    Continue
          else {
               l=0        r=0
               j = i+1
               while( j<n && nums[j] ==1){
```

```
            r++
            j++
    }
</> Code
    j =  i-1
    while (j>=0 && nums(j)==1) {
            l++
            j--
    }
```

```
    if ( l+r == total_ones)
            ans = max(ans, l+r)
    else
            ans = max( ans, l+r+1)
}
```

Majority Element

< Question > : Given array [N]. Find the majority element

$\downarrow$

Elements which occurs more than N/2 times.

- *You can assume that majority element always exists.*

2, 1, 4 $\times$

1 ≤ N ≤ 10^3

3, 3, 4 $\Rightarrow$ 3

3, 3, 2, 1 $\times$ no of occurrences $> N/2$

arr[ ] → [3 , 4 , 3 , 6 , 1 , 3 , 2 , 5 , 3 , 3 , 3] 3 $6 > 11/2$

arr[ ] → [4 , 6 , 5 , 3 , 4 , 5 , 6 , 4 , 4 , 4] $\times$ $5 > 10/2$

1, 2, 3

elem	1	1	$\boxed{3}$
Count	1	0	1

Observations :

At max, how many majority elements are possible?

Can there be 2 majorities? No

$f_1 > N/2$

$f_2 > N/2$

Max 1 majority

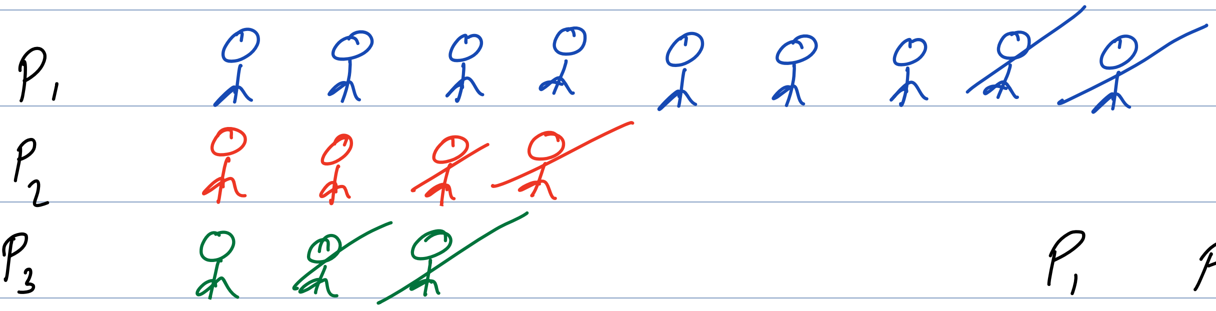
Brute force: sort & count

↳ nlogn

arr[ ] → [3 , 4 , 3 , 6 , 1 , 3 , 2 , 5 , 3 , 3 , 3]

$1\ 2\ \underbrace{3\ 3\ 3\ 3\ 3\ 3}_{6}\ 4\ 5\ 6$

① ①



P_1	P_2	P_3
9	4	3
8	3	3

Obs: remove 2 distinct party

seats, majority is 7 3 2
same 7 2 1

arr[ ] → [3 , 4 , 3 , 6 , 1 , 3 , 2 , 5 , 3 , 3 , 3]

int elem	3	3	3	3	1	1	2	2	3	3	3
int count	1	0	1	0	1	0	1	0	1	2	3

x

$6 > 11/2$

< / > Code

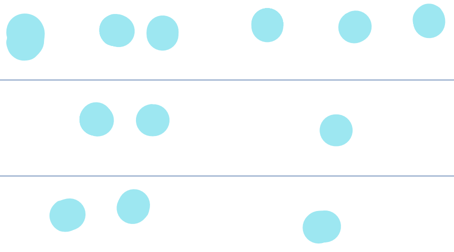
```
elem = a[0]
count = 1
for( i: 1 → n-1 ){
        if( a[i] == elem )
                count ++
        else {
                if (count > 0)
                        count --
                else {
                        elem = a[i] , count = 1
                }
        }
}
```

```
// Check if elem is majority
count_of_elem = 0
for( i : 0 -> n-1 ) {
    if( a[i] == elem)
        count_of_elem++
}
if (count_of_elem > N/2 )
    return elem
else
    // No majority
```

TC: O(N)

Omao	1	2	3
27	10	12	1

0 1 1 0 1 0



< Question > : Given array [N] [M]. $arr[i][j] \geq 0$

Make all elements in a row and column zero if arr[i][j] $= 0$

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 0 \\ 9 & 2 & 0 & 4 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 0 \\ 9 & 2 & 13 & 4 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 2 & 3 & -1 \\ -1 & -1 & -1 & 0 \\ 9 & 2 & 13 & -1 \end{bmatrix}$$

$$\downarrow$$

$$\begin{matrix} 1 & 2 & 3 & 0 \\ 0 & 0 & 0 & 0 \\ 9 & 2 & 13 & 0 \end{matrix}$$

Approach :

</> Code

```
for( i : 0 → n-1 ) {
        convert = false
        for( j : 0 → m-1 ) {
                if (arr[i][j] == 0)
                        convert = true
        }
        if( convert == true ) {
                for( j : 0 → m-1 ) {
                        if (arr[i][j] != 0)
                                arr[i][j] = -1
                }
        }
}

for( j : 0 → m-1 ) {
        convert = false
        for( i : 0 → n-1 ) {
                if (arr[i][j] == 0)
                        convert = true
        }
        }
        if( convert == true) {
            for(i : 0 → n-1) {
                if (arr(i)(j) != 0)
                    arr (i)(j) = -1
            }
        }
}


for (i : 0 → n-1) {
    for(j : 0 → m-1) {
        if( arr(i)(j) == -1)
            arr (i)(j) = 0
    }
}
```

TC : $O(n^2)$