SCALER

Interview Problems

TABLE OF CONTENTS

Benefits of Constraints:

1) Data structure.
2) Time complexity.
3) Algorithm.

$N <= 10^6$ $O(N)$, $O(n \log n)$

$N <= 10^3$ $O(N^2)$

< *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.

A = [1, 1, 0, 1, 1, 0, 1]

1 ≤ N ≤ 10^3

Ans = 5

A = [0]

Ans = 1

Quiz 1:

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

Ans = 6

Quiz 2: [0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0]

4 | 6 | 5 | 3

Ans = 6

[1, 1, 1, 1,] → 5

ans = ~~0~~ 6

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

- l = 0, r = 3, l+r+1 ⇒ 4
- l = 3, r = 2, l+r+1 ⇒ 6
- l = 2, r = 2, l+r+1 ⇒ 5
- l = 2, r = 0, l+r+1 ⇒ 3

Sol:

For every zero check max consecutive 1's, we can have by updating this 0 with 1.

Count of 1's on left and right side.

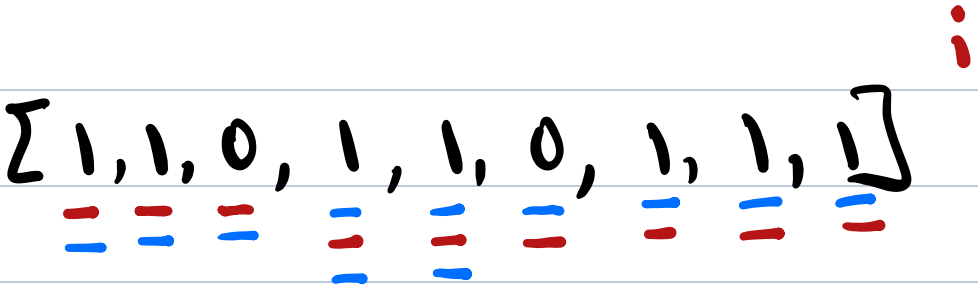
```
ans=0 ; totalOnes=0
for(i=0;i<N;i++)
{
    if(A[i]==1)
    {
        totalOnes++;
    }
}
```

O(N)

```
if (totalOne == N)
{
    return N;
}

for(i=0; i<N; i++)
{
    if(A[i]==0)
    {
        l=0;
        for(j=i-1; j>=0; j--)
        {
            if(A[j]==1)
            {
                l++;
            }
            else
```

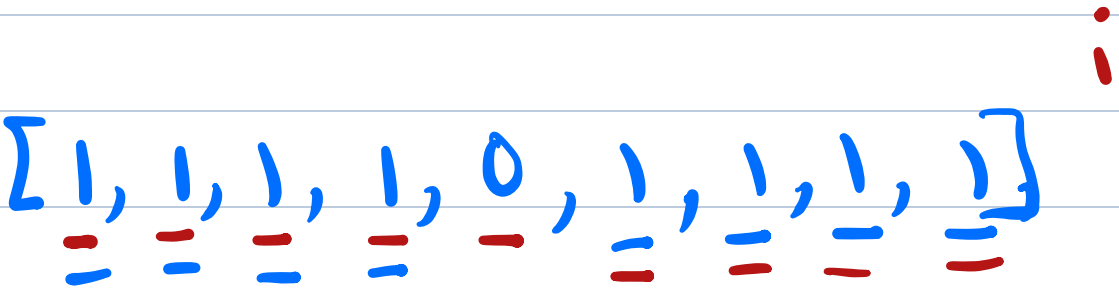
< / > Code

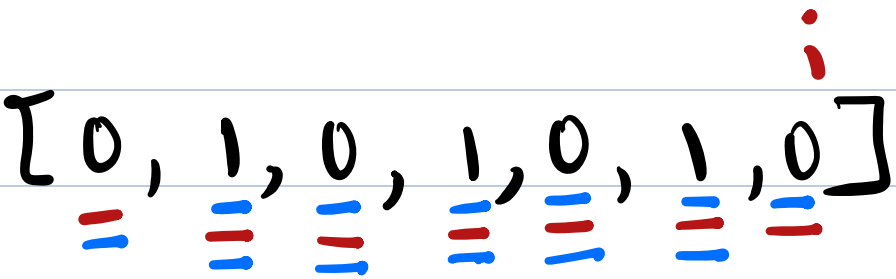
```
        }
    }
    break;
}

r = 0
for (j = i+1; j < N; j++)
{
    if (A[j] == 1)
    {
        r++;
    }
    else
    {
        break;
    }
}

if ( (l+r+1) > ans)
{
    ans = l+r+1;
}
}
}

return ans;
```

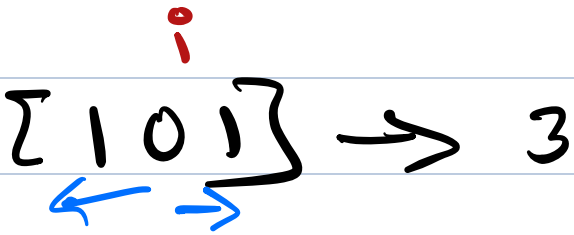


TC = 3N + N = O(N)

SC = O(1)







l = 1
r = 1

l + r + 1 => 3

< *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.

6

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

Ans = 6

0 1 2 3 4 5

A[] = { 1, 1, 0, 1, 1, 1 }

Ans = 5

Interview Problems SCALER

```
ans=0; totalOnes=0
for(i=0;i<N;i++)
{
    if(A[i]==1)
    {
        totalOnes++;
    }
}
if (totalOne == N)
{
    return N;
```

O(N)

```
for(i=0; i< n; i++)
{
    if(A[i] == 0)
    {
        l=0;
        for(j=i-1; j>=0; j--)
        {
            if(A[j]==1)
            {
                l++;
            }
            else
            {
                break;
            }
        }
```

Interview Problems

SCALER

< / > Code

< / > Code

```
    r=0
    for(j=i+1; j< n; j++)
    {
        if(A[j]==1)
        {
```

```
        r++;
    }
    else
    {
        break;
    }
    if(ltr == totalOnes)
    {
        return ltr;
    }
    if( (ltr+1) > ans)
    {
        ans = ltr+1;
    }
}
}
return ans;
```

TC : O(N)

SC : O(1)

Majority Element

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

↓

Elements which occurs more than N/2 times.

freq. > $\frac{N}{2}$

A = [2 , 1 , 4] (0, 1, 2) → No majority

A = [2 , 2 , 4] → > $\frac{N}{2}$

freq(2) > $\frac{3}{2}$

Ans = 2

2 > 1

```
      0  1  2  3  4  5  6  7  8  9  10
A[] = [3, 4, 3, 6, 1, 3, 2, 5, 3, 3, 3]
```

$N = 11 \quad \frac{N}{2} = \frac{11}{2} = 5$

Ans = 3

freq(3) > 5

6 > 5

```
      0  1  2  3  4  5  6  7  8  9
A[] = [4, 6, 5, 3, 4, 5, 6, 4, 4, 4]
```

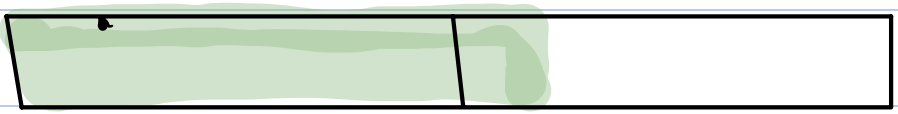
$N = 10 \quad \frac{N}{2} = 5$

freq(4) > 5

5 ≯ 5

No majority element.

majority ele $>= \frac{n}{2} + 1$



There will be only one majority ele.

Brute force:

Iterate and find freq. of each element

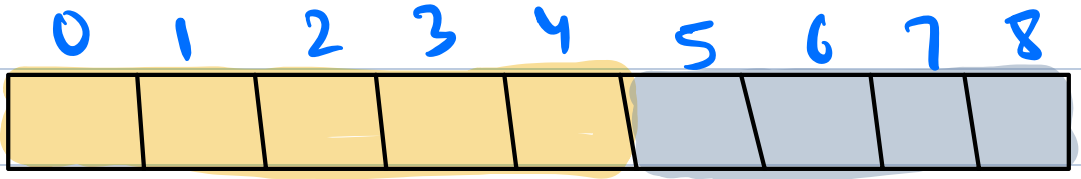
$TC = O(N^2)$

// Slightly better : Sorting $O(n \log n)$
// HashMap TC: $O(N)$ SC = $O(N)$

Break : 10:35 pm

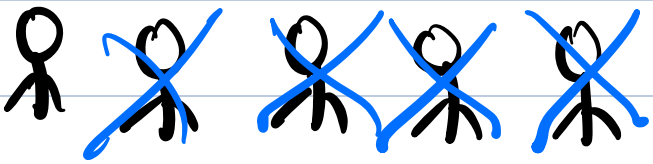
Moore's voting Algo:

If we remove any two distinct ele, the majority element still stay the same.

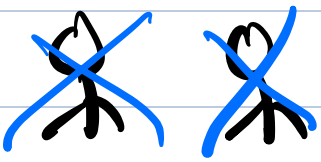


Total = 9

min req. = 5

Junaid 

Lakshit 

Karan 

Chandan 

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

Maj = 3 3 X X 3

Count = X 0 X 0 X 0 X 0 X X 3

[3, 3, 4, 3, 5, 7, 3].

Maj = no 3 3

Count = 0 X X X X 0 1

[3, 3, 4, 3, 3, 5, 7]

Maj = 3

Cont = X X X X 3 2 1

i

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

Maj = 0 0 0

Count = 1 0 1 0 1 0

Edge Case

i

1, 1, 2, 3, 4, 5, 6

Maj = 1 4 6

Count = 1 1 1 0 1 0

```
maj = -1;
count = 0;

for(i=0; i<N; i++)
{
    if(count==0)
    {
        maj = A[i];
        count = 1;
    }
    else
    {
        if(A[i]==maj) count++;
        else count--;
    }
}
if(count==0) maj = -1
```

TC: O(N)
SC: O(1)

```
count-maj = 0
for(i=0; i<N; i++)
{
    if(A[i]==maj) count-maj++
}
if(count-maj > N/2) return maj;
else return -1;
```

A[i][j] >= 0

< *Question* > : Given array [N] [M].

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

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

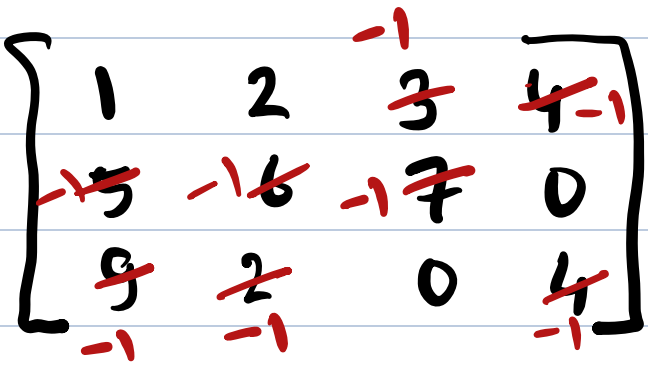
$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 0 \\ 9 & 2 & 13 & 4 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 2 & 3 & 0 \\ 0 & 0 & 0 & 0 \\ 9 & 2 & 13 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} \cancel{2}^{-1} & 0 & \cancel{3}^{-1} & \cancel{4}^{-1} \\ 5 & \cancel{6}^{-1} & 7 & 1 \\ 9 & \cancel{2}^{-1} & 5 & 4 \end{bmatrix}$$

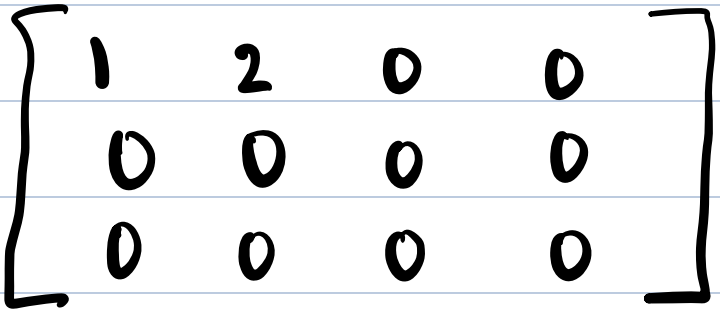
identify updated original 0

1) update all row/col with (≠0) -1.
where current ≠ 0

2) update all -1 with 0.

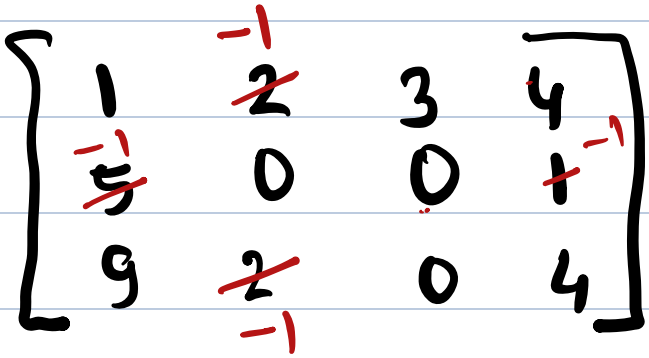


(2)



TC: O(n*m)

Doubt session



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

```
for (row=0; row<N; row++)
{
    for (col=0; col<M; col++)
    {
        if (A[row][col]==0)
        {
```