Interview Problems

ANALYSING CONSTRAINTS

① This can help us to determine which time complexity & data structure or algorithm to use for a given problem.

② It is important to look at the constraint whenever we are solving a problem.

Note :- In Interviews, dont ask the constraint directly Rather, tell your approach & ask interviewer if we need to optimize further.

POSSIBLE CONSTRAINTS		TC
①	$N <= 10^6$	$O(N), O(N \log N)$
②	$N <= 20$	$O(N!), O(2^N)$
③	$N <= 10^{10}$	$O(\log N), O(\sqrt{N})$

$\because \log_2(2^{10}) = \log(1024) = 10$

$\therefore \log_2(10^{10}) = \log(10^3 \times 10^3 \times 10^3 \times 10)$

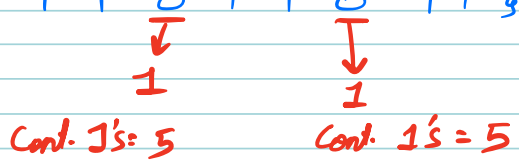
$= \log 10^3 + \log 10^3 + \log 10^3 + \log 10$

≤ 30

[Amazon, Microsoft, Adobe]

Question :- Given an array of 1's & 0's, you are allowed to replace one 0 with 1. Find the maximum no. of consecutive 1's that can be obtained after making the replacement.

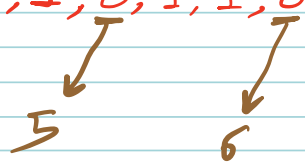
Ex :- arr[] = { 1 1 0 1 1 0 1 1 }



Ans = 5

Quiz 1 Find the maximum no. of consecutive 1's that can be obtained after replacing only one 0 with 1.

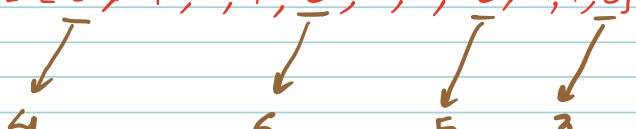
A[] = [1, 1, 0, 1, 1, 0, 1, 1, 1]



Ans = 6

Quiz 2 Find the maximum no. of consecutive 1's that can be obtained after replacing only one 0 with 1.

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



Ans = 6

OBSERVATION

↳ For all 0's Count the no. of 1's on Left & Right.

PSEUDO CODE

```
int n = nums.size()
int maxCount = 0;
for (i = 0; i < n; i++){
    if (nums[i] == 0){
        int left = 0
        int right = 0
        int j = i-1;
        while (j >= 0 && nums[j] == 1){
            j--;
            left++;
        }
        j = i+1;
        while (j < n && nums[j] == 1){
            j++;
            right++;
        }
        maxCount = max(maxCount, l+R+1);
    }
}
return maxCount;
```

Count 1's on Left

Count 1's on Right

Edge Case :- ① All of them are 1's.

↳ How to Handle ??

↳ before for loop, Count no of 1's in the array.

```
if (Count of 1's == length){
    Return length
}
```

② All of them are 0's.
↳ [Already handled].

Quiz 3 what will be the TC of this approach?

TC ⇒ O(N)

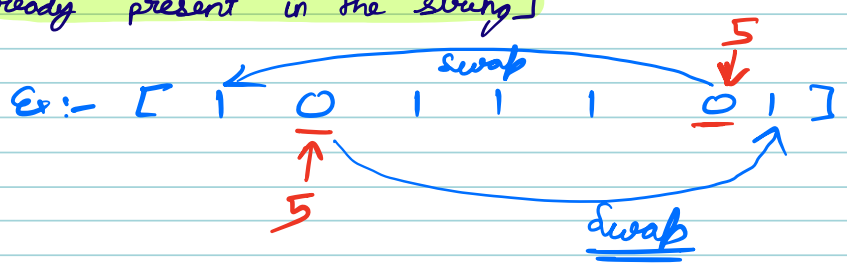
Ex:- 0 1 1 1 0 1 1 0 1 1 1 1 0

Ex:- 1 1 1 1 1 1 0 X

SC → O(1)

[Variation of previous problem]

Question ⇒ Given an array of 0's & 1's. Find the maximum no. of consecutive 1's that can be obtained by SWAPPING at most one 0 with 1.
[already present in the string]



Ans = 5

Observation
↳ I should not swap 0 with continuous 1's on the Left or Right.

Ques 4:- Find the maximum no. of consecutive 1's that can be obtained by SWAPPING at most one 0 with 1

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

5

Soln Ans = 5

IDEA :-

Ex:- 0 1 1 1 1 0 1 1 1 0 1 1 0

5 8 6 3

★ For this scenario Ans is L+R+1

Ex:- 1 1 1 0 1 1 1 0

L R

3 3 → 6

3 0 → 4

★ If No option to choose except Left & Right continuous 1's then Ans = L+R.

Observation :-

```
if (L + R == Count of 1's in array) {
    Ans = max(Ans, L+R);
}
else {
    Ans = max(Ans, L+R+1);
}
```

PSEUDO CODE

```
int n = nums.size();
int maxCount = 0;
countOfOnes // : TODO
for (i = 0; i < n; i++){
    if (nums[i] == 0){
        int left = 0
        int right = 0

        int j = i - 1;
        while (j >= 0 && nums[j] == 1){
            j--;
            left++;
        }

        j = i + 1;
        while (j < n && nums[j] == 1){
            j++;
            right++;
        }

        if (L + R == countOfOnes){
            maxCount = max(maxCount,
                           L + R);
        } else {
            maxCount = max(maxCount,
                           L + R + 1);
        }
    }
}
return maxCount;
```

Count 1's on Left

Count 1's on Right

TC → O(N)

SC → O(1)

MAJORITY ELEMENT

Question:- Given an array of N integers, find the Majority Element.

The Majority element is the element that occurs more than $n/2$ times where n is size of the array.

Ex:- { 2, 1, 4 }

$n = 3 \longrightarrow$ 2 element required.

[No Majority]

Ex:- { 3, 4, 3, 2, 4, 4, 4, 4 }

$n = 8 \longrightarrow$ 5 elements required.

[Majority $\Rightarrow 4$]

Ex:- { 3, 3, 4, 2, 4, 4, 2, 4 }

$n = 8 \longrightarrow$ 5 element required.

[No Majority].

Quiz 5 what is the majority element in this array?

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

$n = 11 \longrightarrow >= 6$ elements required

Ans = 3

Quiz 6 what is the Majority element in the array.

4, 6, 5, 3, 4, 5, 6, 4, 4, 4

$n = 10 \longrightarrow >6$ elements required.

[No majority element]

Quiz 7 At most how many majority elements can be there ?

Ans = 1

Proof:- Let say there are 2 elements existing m_1 & m_2

So, $m_1 > \frac{n}{2}$ & $m_2 > \frac{n}{2}$

Let add them

$m_1 + m_2 > \frac{n}{2} + \frac{n}{2}$

$\Rightarrow m_1 + m_2 > n$ ∠ Not possible

Hence proved.

Brute Force

↳ Count freq of all elements

Nested loop → TC → $O(N^2)$, SC → $O(1)$

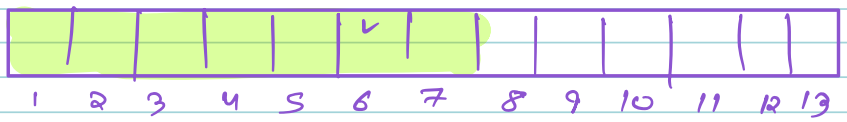
hashmap → TC → $O(N)$, SC → $O(N)$

↳ OPTIMIZATION

Observations

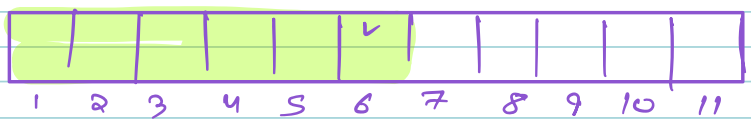
① There can be only 1 majority elements

(2)



N = 13 ⟶ 7 elements required.

⇒ Remove element from Left & Right



N = 11 ⟶ 6 elements required.

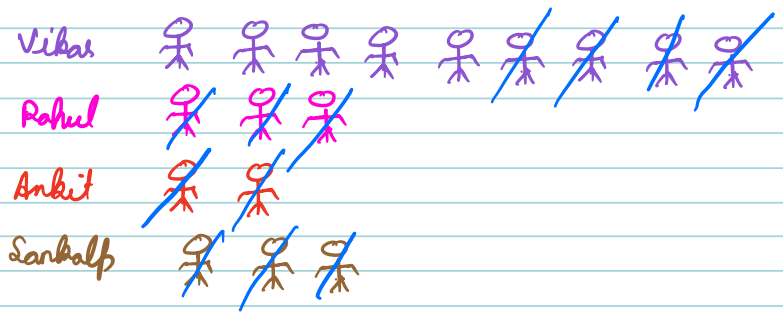
⇒ Again Remove

N = 9 ⟶ 5 elements required.

* If we continue this then only Majority element will be remaining.

So, Removing elements will not Majority

Ex2 :- Elections



Remove	Purple	Pink	Red	Brown	Winner
1 Vikas & 1 Rahul	8	2	2	3	Vikas
1 Vikal & 1 Sankalp	7	2	2	2	Vikas
1 Vikas & 1 Ankit	6	2	1	2	Vikas
1 Rahul & 1 Ankit	6	1	0	2	Vikas
1 Rahul & 1 Sankalp	6	0	0	1	Vikas
1 Vikas & 1 Sankalp	5	0	0	0	Vikas

Observation :- Deleting two different party member Won't change Majority.

Moore's Voting Algorithm

↳ Majority whatever is in Majority remains in every after deleting elements.

DRY RUN

Ex1 :- 3, 4, 3, 6, 1, 3, 2, 5, 3

Majority = 3 3 4 2 3

Count = 1 0 1 0 1 0 1 0 1

Ex2 :- 3, 3, 3, 3, 4, 3, 6, 1, 3, 2, 5

Majority :- 3

count :- 1 2 3 4 3 4 3 2 3 2 1

Ex3:- 3 4 3 6 3 1 3 2 3 5 3

Majority = 3

Count = 1 0 1 0 1 0 1 0 1 0 1

Ex4 :- 3 1 2 2 2

Majority : 2

Count : 1 0 1 0 1

EDGE CASE

arr[] = 4 3 4 3 4 3 6

Majority = 4

Count = 1 0 1 0 1 0 1

Conclusion:- If there is majority than it will be given by Max Variable otherwise No. Majority. So In last we need to check frequency of Majority in the given array to conclude anything!

N = 8 → 5

3 3 3 4 7 6 3

Maj = 6

Count = 1 0 1 2 1 0 1 0

PSEUDO CODE

```
Maj Ele = arr[0] , count = 1
for( i = 1 ; i < n; i++){
    if (count == 0) {
        Maj Ele = arr[i]
        count = 1;
    }
    else {
        if (arr[i] == Maj Ele){
            count++;
        }
        else {
            count--;
        }
    }
}
// check the freq of Maj Ele.
int freq = 0;
for( i = 0; i < N; i++){
    if ( arr[i] == Maj Ele){
        freq++
    }
}
if ( freq > N/2 ) {
    return Maj Ele;
}else {
    return -1;
}
```

TC → O(N)

SC → O(1)

Question :- you are Given 2D integer matrix A, make all the elements in a row or column zero if the A[i][j] = 0. Specifically, make entire i^{th} row & j^{th} column zero.

Note :- `arr[i][j] >= 0`

Ex :-

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

Ex2 :-

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & 5 \\ 0 & 7 & 8 \end{bmatrix}$$

Observations

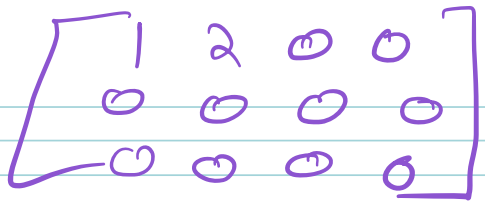
① We can't Do it on fly because it will make entire 2D 0 in some case.

② We need to store 0's position.

IDEA :- whenever I got 0, Convert entire Row & column element to -ve. And later Convert all -ves to 0.

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

⇓



PSEUDO CODE

```
=> For Rows  <- for (i=0; i<n; i++){
                  int flag = 0;
                  for (j=0; j<m; j++){
                      if (arr[i][j] == 0){
                          flag = 1;
                      }
                  }
                  if (flag == 1){
                      for (j=0; j<m; j++){
                          arr[i][j] *= -1;
                      }
                  }
              }
```

To check 0 in row

Make element -ve if there is 0 in row

=> Illy for columns // TODO

=> then Iterate again & make all -ves 0.

TC	$\rightarrow O(N \times M)$
SC	$\rightarrow O(1)$