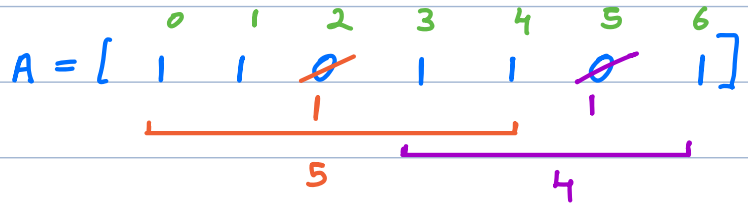
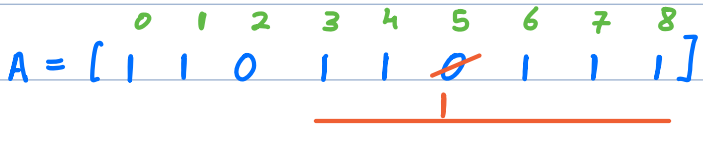
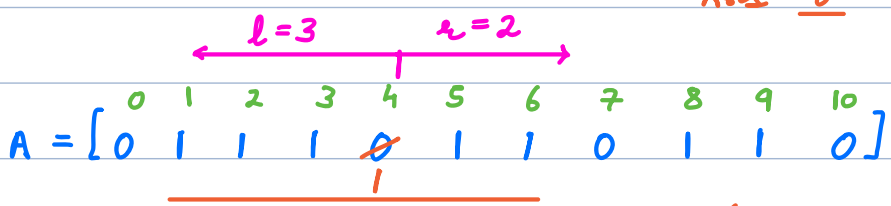
Q → Given an integer array with 1 & 0.
You are allowed to replace exactly one 0 with 1.
Find the max # of consecutive 1's that can be formed after replacement.



Ans = 5



Ans = 6



Ans = 6

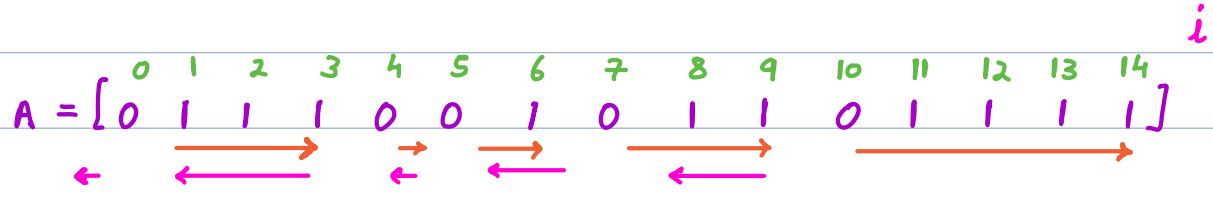
Sol → ∀ 0's convert it to 1 & find max # of consecutive 1's.

l = # consecutive 1's on left of 0
r = # consecutive 1's on right of 0
ans = l + 1 + r

if (∀i, A[i] = 1) → ans = n

```
cnt = 0
for i → 0 to (N-1) {
    cnt += A[i]
}
if (cnt == N) return N
```

```
ans = 0          // or 1
for i → 0 to (N-1) {
    if (A[i] == 0) {
        l = 0
        for j → (i-1) to 0 {
            if (A[j] == 1) l++
            else break
        }
        r = 0
        for j → (i+1) to (N-1) {
            if (A[j] == 1) r++
            else break
        }
        ans = max(ans, l + r + 1)
    }
}
return ans
```



total # iterations ~ O(2N) ⇒ TC = O(N)

Q → Given an integer array with 1 & 0.
You are allowed to ~~replace~~ swap exactly one 0 with 1.
Find the max # of consecutive 1's that can be formed after replacement.

```
      0  1  2  3  4  5
A = [ 1  1  0  1  1  1 ]
         1
```

ans = 6 } Ans = 5
cnt = 5

```
cnt = 0   // #1's
for i → 0 to (N-1) {
    cnt += A[i]
}
if (cnt == N)  return N

ans = 0        // or 1
for i → 0 to (N-1) {
    if (A[i] == 0) {
        l = 0
        for j → (i-1) to 0 {
            if (A[j] == 1) l++
            else break
        }
        r = 0
        for j → (i+1) to (N-1) {
            if (A[j] == 1) r++
            else break
        }
        ans = max(ans, l + r + 1)
    }
}
return ans   min(ans, cnt)
```

Q → Given an integer array, find the majority element i.e. an element that occurs > N/2 times. If no element is in majority, return -1.

A = [2 1 4] (0 1 2) Ans = -1

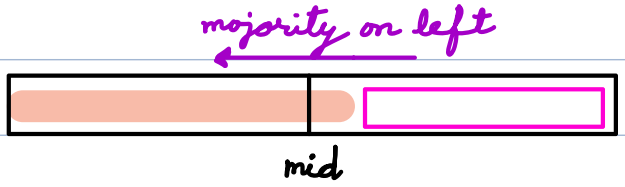
A = [3 4 3 4 4] (0 1 2 3 4) Ans = 4

A = [3 4 3 6 1 3 2 5 3 3 3] (0 1 2 3 4 5 6 7 8 9 10)

freq(3) > N/2 Ans = 3

6 > 11/2

A = [4 6 5 3 4 5 6 4 4 4] (0 1 2 3 4 5 6 7 8 9) Ans = -1



freq(majority element) > freq(everything else)

Bruteforce → ∀i, calculate freq of A[i] & check if the freq > N/2 to be the one.

1) Iterate & find freq. TC = O(N²) SC = O(1)

2) Sort & calculate freq. TC = O(N log(N)) SC = O(1)

3) Use hashmap to precompute freq. TC = O(N) SC = O(N)

Red → 8 8 8 8 8 8 8 8

Yellow → 8 8

Green → 8 8 8

Blue → 8 8

↓ more out 2 people

distinct

freq (majority element) > freq (everything else)

$x > y$

$x-1 > y-1$: 1 majority & 1 other

$x > y-2$: 2 others

Moore's Voting Algo

	0	1	2	3	4	5	6	7	8	9	10 (i)
A = [	3	4	3	6	1	3	2	5	3	3	3]

majority element = 3 TC = O(N)

cnt = 1 0 1 0 1 0 1 3

check if freq(3) > N/2 → TC = O(N)

Overall TC = O(N)

SC = O(1)

```
m = A[0]
cnt = 1
for i → 1 to (N-1) {
    if (A[i] == m) cnt++
    else {
        if (cnt > 0) cnt--
        else { m = A[i] cnt = 1 }
    }
```

```
}
f = 0
for i → 0 to (N-1) {
    if (A[i] == m) f++
}
if (f > N/2) return m
else return -1
```

A = [2 2 2 5 5 8 1 2] (indices 0 1 2 3 4 5 6 7)

i

m = ~~2~~ 1

cnt = ~~1 2 3 2 1 0~~ 1 0

Ans = -1

TC = O(N)

SC = O(1)

Q → Given a 2D matrix, make all the elements in a row 'i' & column 'j' zero if A[i][j] = 0.

$\forall i,j$ A[i][j] >= 0

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 0 \\ 0 & 6 & 7 \end{bmatrix} \qquad A = \begin{bmatrix} 0 & 2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & ~~2~~\to 0 & \times\ 0 \\ 0 & 0 & (0) \\ (0) & (0) & 0 \end{bmatrix}$$

SC = O(1)

Sol →

1) First update all row & col to -1 ✓

2) Update all -1 to 0 ✓

$$A = \begin{bmatrix} ~~1~~\to 0 & 2 & ~~1~~\to 0 \\ ~~1~~\to 0 & ~~1~~\to 0 & (0) \\ (0) & ~~1~~\to 0 & ~~1~~\to 0 \end{bmatrix}$$

TC = O(N*M)
