Avg PSP 74 → 77 → 75 → 77 → 80

By Thursday

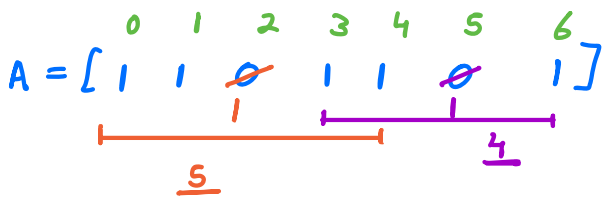
learners with 100% PSP → 55 → 100

Constraints

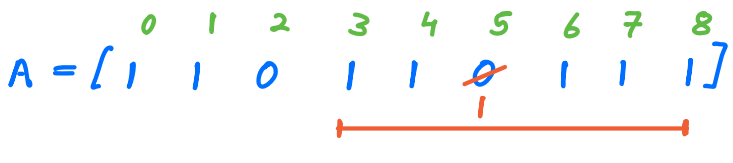
- can help in avoiding TLE.
- Decide datatype to use.

Use it but do not depend on it, ∵ in interviews we don't get constraints.

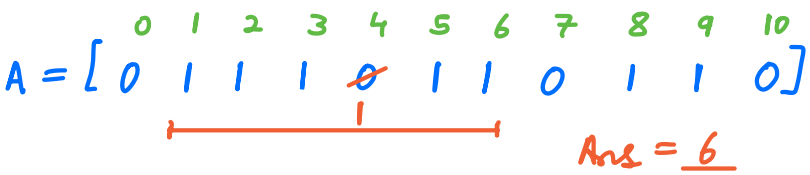
Q → Given a binary array of 0's & 1's.
Find the max # consecutive 1's that can be obtained by updating atmost one 0 to 1.



Ans = 5



Ans = 6



Ans = 6

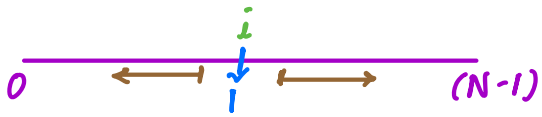
Ans = 5

If All 1's ⇒ Ans = N

else ⇒ check updating which 0 can give best ans.

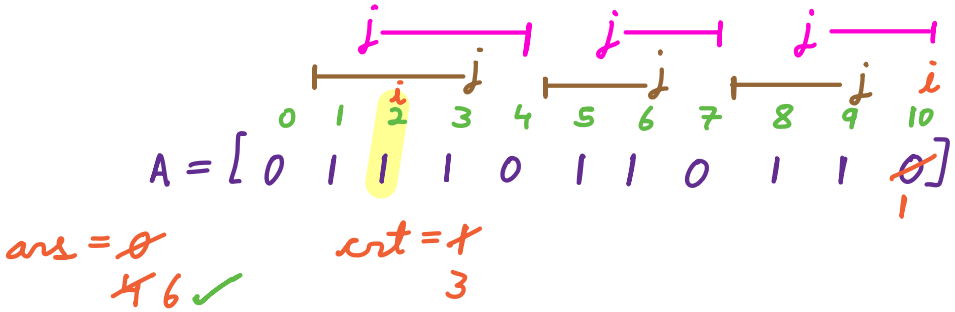
```
cnt = 0          // count # 1's
for i → 0 to (N-1) {
    if (A[i] == 1)  cnt++
}
if (cnt == N)  return N
```

(optional)




```
ans = 0
for i → 0 to (N-1) {
    if [A[i] == 0] {
        cnt = 1          // current 0 → 1
        for j → (i-1) to 0 {          // L ← R
            if (A[j] == 1)  cnt++     // # 1's on left
            else  break
        }
        for j → (i+1) to (N-1) {
            if (A[j] == 1)  cnt++     // # 1's on right
            else break
        }
        ans = max(ans, cnt)
    }
}
if (ans == 0)  return N
return ans
```

for (j = i-1; j >= 0; j--)

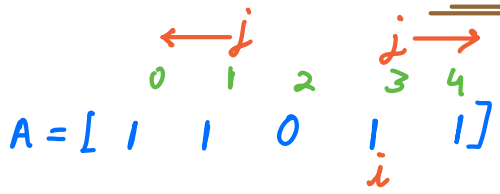


Total # element while travelling left = O(N)

right = O(N)

times any element is touched = 3

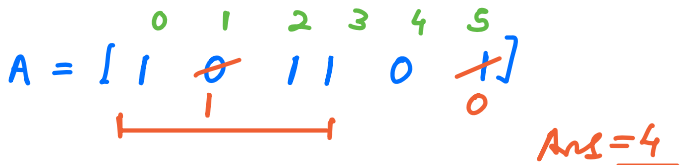
⇒ TC = O(N) SC = O(1)



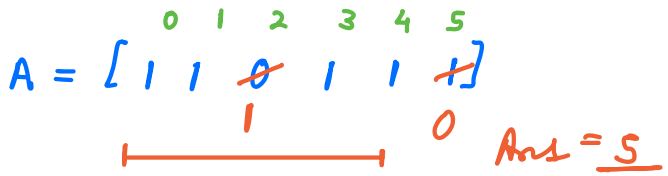
i	j	# iterations
0	—	1
1	—	1
2	(i-1)→0 (i+1)→(N-1)	N
3	—	1
4	—	1

~ 2N = O(N)

Q→ Given a binary array of 0's & 1's. Find the max # consecutive 1's that can be obtained by swapping atmost one 0 to 1.



Ans = 4



Ans = 5

✓ // same as previous solution

⇒ if (ans > total_One) return ans-1

	0	1	2	3	4	5	6	7	8	9
A =	1	1	0	1	1	0	1	1	1	1

l

ans = ~~0~~ 8 7

cnt = ~~1~~ 7

TC = O(N) SC = O(1)

Q → Given an integer array, find the majority element. Majority element → frequency > N/2
If not present, return -1.

	0	1	2
A =	2	1	4

Ans = -1

	0	1	2	3	4	5	6	7
A =	3	4	4	3	2	4	4	4

freq (4) = 5 > N/2 Ans = 4

	0	1	2	3	4	5	6	7	8	9	10
A =	3	4	3	6	1	3	2	5	3	3	3

freq (3) = 6 > N/2 Ans = 3

	0	1	2	3	4	5	6	7	8	9
A =	4	6	5	3	4	5	6	4	4	4

freq (4) = 5 ≯ N/2 Ans = -1

Max # majority elements = 1

if (freq(x) > N/2) ⇒ freq(!x) < N/2

Bruteforce → ∀i, count frequency & return if freq > N/2 for any element.

TC = O(N²) SC = O(1)

Store frequency → ∀i store freq in a single traversal. (HashMap<>)

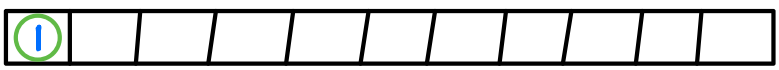
Iterate & check if freq > N/2 for any element.

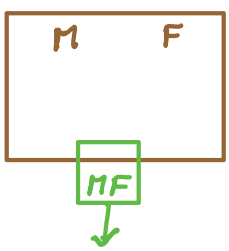
TC = O(N+N) = O(N) SC = O(N)

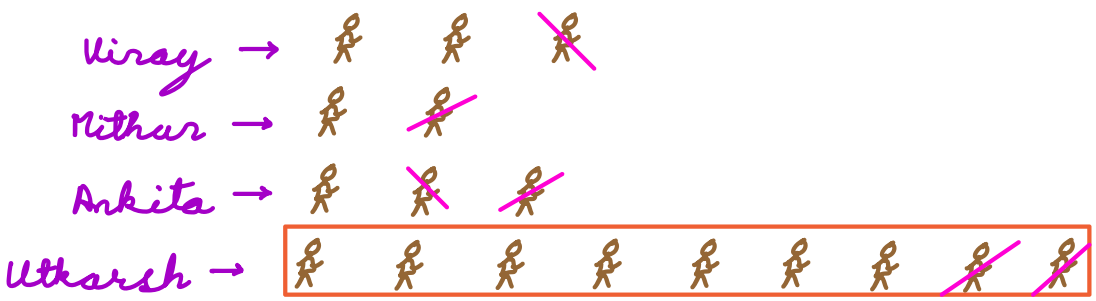
for i → 0 to (N-1) { F[A[i]]++ }

F[i] = freq of i.

Solution (Moore's Voting Algo)







freq(u) > N/2 freq(!u) < N/2

17 9 8

15 8 7

13 7 6

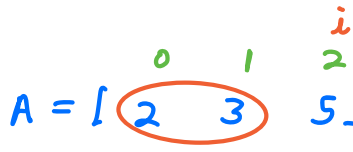
11 7 4

if 2 distinct elements are removed

⇒ majority element remains same.

majority = ~~3~~ ~~3~~ 3 ✓ check if freq(3) > N/2 ⇒ Ans = 3

freq = 1 0 1 0 1 ~~2~~ ~~3~~ ~~4~~ 1 ~~2~~ 1



m = ~~2~~ 5 Ans = -1

f = 1 0 1

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

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

TC = O(N)

SC = O(1)

if (f > N/2) return maj
else return -1

Q → Given a 2D matrix, make all the elements in a row or column 0 if any cell in that row or column is equal to 0.

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

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

while travelling first, update → INT_MAX
row & col values to -1 (if its != 0).
Once travelling completes → update all -1 to 0.
