# Two Pointers

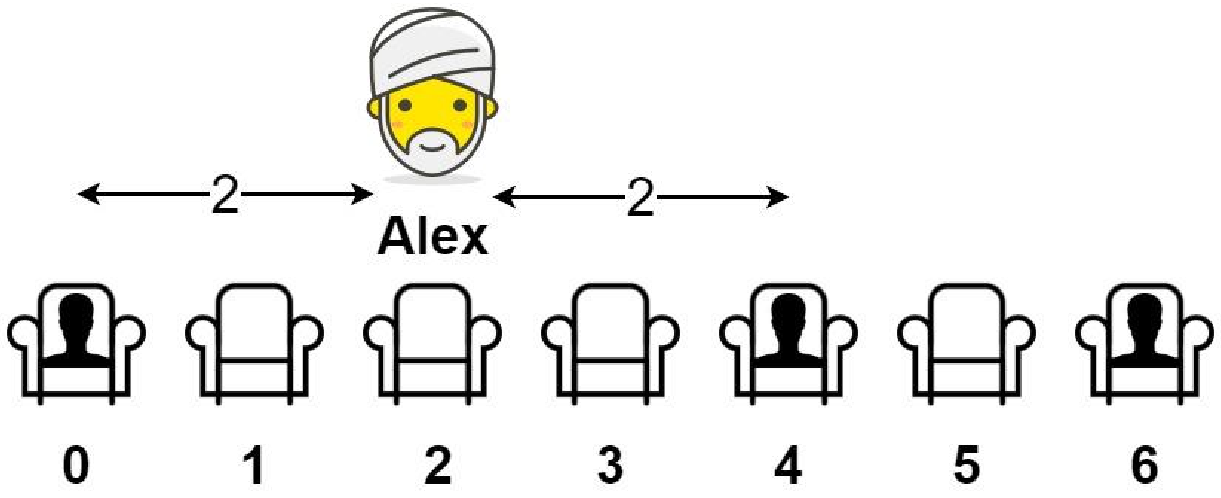
Question: https://leetcode.com/problems/maximize-distance-to-closest-person/
This was the daily leetcode question of 16/01/2022.

*You are given an array representing a row of seats where seats[i] = 1 represents a person sitting in the ith seat, and seats[i] = 0 represents that the ith seat is empty (0-indexed).*

*There is at least one empty seat, and at least one person sitting.*

*Alex wants to sit in the seat such that the distance between him and the closest person to him is maximized.*

*Return that maximum distance to the closest person.*



```
Input: seats = [1,0,0,0,1,0,1]
Output: 2
Explanation:
If Alex sits in the second open seat (i.e. seats[2]), then the closest person
has distance 2.
If Alex sits in any other open seat, the closest person has distance 1.
Thus, the maximum distance to the closest person is 2.
```

## My Approach:

https://gist.github.com/vermaayush680/a424ee32af48a7004635544cd749f6f7

My approach was to use Kadane's algorithm to find the maximum number of Zeros and see if there was a 1 next to it. If so print len(maxSoFar)//2.
If there were zeros till the end of the list then print maxSoFar.
It worked well with normal cases but failed at cases where the array starts from zero and ends with 1, ex : [0,0,0,1]

```
maxCur=0
maxSoFar=0
l=0
for i in seats:
    maxCur +=(i==0)
    if i!=0:
        print(i)
        maxSoFar=max(maxSoFar,maxCur)
        l=(maxSoFar//2)+((maxSoFar%2)!=0)
        maxCur=0
if maxCur>maxSoFar:
    return maxCur
else:
    return l
```

## Solution:

https://gist.github.com/vermaayush680/325c493066b97cb4a046288be8bd3443

Then looking at the discussions, I realized that my problem can be solved using Two Pointers.

Two Pointer Explanation: https://www.geeksforgeeks.org/two-pointers-technique/

We find seat==1 and check the distance from the ***last*** person.
Initially last=-1 to handle the special case of starting with zero.

```
last=-1
res=0
n=len(seats)
for i in range(n):
    if seats[i]:
        if last<0:
            res=max(res,i)
        else:
            res = max(res,(i-last)//2)
        last=i
return max(res,n-1-last)
```

An easy question with not so easy to come up with solutions.
The question without special cases is easy but the special cases make it a medium difficulty question.