Problem # 849: Maximize Distance to Closest Person (Medium)

<https://leetcode.com/problems/maximize-distance-to-closest-person/>

Solution:

<https://leetcode.com/problems/maximize-distance-to-closest-person/discuss/917178/Simple-Python-3-Solution-Runtime-beats-61.15>

1. Let n be the length of the seats array (i.e. total number of seats).

2. Construct an array of occupied seats i.e. array occupied has the index of the seat where value is 1.  
3. Get the index of first occupied seat, i.e. first\_occupied.

4. Get theindex of the last occupied seat, i.e. last\_occupied.

5. Initialize j to 0 where j is the pointer for occupied list.

6. Initialize a dictionary called my\_dict with key = index of the seat and value = minimum distance from an occupied seat. Obviously if a seat is occupied, the minimum distance is 0.

7.Iterate through the array of seats using its index (i.e. position).

(1) If the value of the seat is 1, it is already occupied. So minimum distance to an occupied seat is 0. Increment the pointer j to the next occupied seat in the occupied array.

(2) However, if the value of the seat is 0, it is not occupied. So, it is a candidate seat for occupying.

(3) If the index is less than first occupied seat, then minimum distance is the difference between the first occupied seat and the index.

(4) If the index is more than the first occupied seat, but less than the last occupied seat, it is between 2 occupied seats. So calculate the distance from the left occupied seat and the distance from the right occupied seat. Find the minimum of these two distances.

(5) If the index is more than the last occupied seat, the minimum distance is the difference between the index and the last occupied seat.

Insert into the dictionary, the index as the key and the min\_distance as the value

8. Return the maxiumum value in the dictionary of the values (i.e. min\_distance)

---------------------------------------------------

class Solution:

def maxDistToClosest(self, seats: List[int]) -> int:

n = len(seats)

occupied = [i for i in range(n) if seats[i] == 1]

first\_occupied = occupied[0]

last\_occupied = occupied[-1]

j = 0 # pointer for occupied list

my\_dict = {}

for i in range(n):

if seats[i] == 1: # it is already occupied.

min\_distance = 0

j += 1

else: # if seats[i] == 0: # not occupied. Is this a candidate seat for occupying ?

if i < first\_occupied:

min\_distance = first\_occupied - i

elif i < last\_occupied:

left = i - occupied[j - 1]

right = occupied[j] - i

min\_distance = min(left, right)

else:

min\_distance = i - last\_occupied

my\_dict[i] = min\_distance

return(max(my\_dict.values()))