在一排座位（`seats`）中，`1` 代表有人坐在座位上，`0` 代表座位上是空的。

至少有一个空座位，且至少有一人坐在座位上。

亚历克斯希望坐在一个能够使他与离他最近的人之间的距离达到最大化的座位上。

返回他到离他最近的人的最大距离。

示例 1：

输入： `[1,0,0,0,1,0,1]`
输出： `2`
解释：
如果亚历克斯坐在第二个空位（`seats[2]`）上，他到离他最近的人的距离为 `2` 。
如果亚历克斯坐在其它任何一个空位上，他到离他最近的人的距离为 `1` 。
因此，他到离他最近的人的最大距离是 `2` 。

示例 2：

输入： `[1,0,0,0]`
输出： `3`
解释：
如果亚历克斯坐在最后一个座位上，他离最近的人有 `3` 个座位远。
这是可能的最大距离，所以答案是 `3` 。

提示：

1. `1 <= seats.length <= 20000`
2. `seats` 中只含有 0 和 1，至少有一个 `0`，且至少有一个 `1`。

思路，动态统计从最初到最后的最大距离，按0+1，1重置的规则来统计当前连续的最大距离，需要注意两个特殊情况，从最初位为0和末位为0，他们左右没有相邻的。

```
class Solution {
public:
    int maxDistToClosest(vector<int>& seats) {
        int n=seats.size();
        int cs=1;
        int num=0;
        int max=0;
        for(int i=0;i<n;++i){
            if(seats[i]==0){
                ++num;
                if(i==n-1){
                    if(max<num-1){
```

```
                    max=num-1;
                }
            }
        }else if(cs==1){
            cs=0;
            if(max<(num-1)){
                max=num-1;
            }
            num=0;
        }else{
            if((num-1)/2>max){
                max=(num-1)/2;
            }
            num=0;
        }
    }
    return max+1;
  }
};
```