【华为OD机考 统一考试机试C卷】 找座位（C++ Java JavaScript Python C 语言）

华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

目前在考C卷，经过两个月的收集整理，C卷真题已基本整理完毕
抽到原题的概率为2/3到3/3，也就是最少抽到两道原题。请注意：大家刷完C卷真题，最好要把B卷的真题刷一下，因为C卷的部分真题来自B卷。
另外订阅专栏还可以联系笔者开通在线 OJ 进行刷题，提高刷题效率。
真题目录：华为OD机考机试 真题目录（C卷 + D卷 + B卷 + A卷） + 考点说明
专栏：2023华为OD机试(B卷+C卷+D卷)（C++JavaJSPy）
华为OD面试真题精选：华为OD面试真题精选
在线OJ：点击立即刷题，模拟真实机考环境

题目描述

在一个大型体育场内举办了一场大型活动，由于疫情防控的需要，要求每位观众的必须间隔至少一个空位才允许落座。

现在给出一排观众座位分布图，座位中存在已落座的观众，请计算出，在不移动现有观众座位的情况下，最多还能坐下多少名观众。

输入描述

一个数组，用来标识某一排座位中，每个座位是否已经坐人。0表示该座位没有坐人，1表示该座位已经坐人。

- 1 ≤ 数组长度 ≤ 10000

输出描述

整数，在不移动现有观众座位的情况下，最多还能坐下多少名观众。

用例1

输入	10001
输出	1
说明	无

用例2

输入	0101
输出	0
说明	无

解题思路

1. **初始化变量：** 定义变量 `cnt` 来计数可以坐下的人数，以及变量 `i` 用作字符数组的索引。
2. **遍历字符数组：** 使用 `while` 循环遍历字符数组。循环的条件是索引 `i` 小于数组的长度。
3. **检查和标记座位：**
   检查当前位置是否为空座（seats[i] == ‘0’），
   并且会检查当前位置是否是第一个座位（i == 0）
   或者它的左侧座位是否为空座（seats[i - 1] == ‘0’），
   同时也会检查当前位置是否是最后一个座位（i == seats.length - 1）
   或者它的右侧座位是否为空座（seats[i + 1] == ‘0’）。

   如果这些条件都成立，那么就会执行if语句块中的代码，即增加最大额外观众数（maxAdditional++）、将当前位置标记为已坐人（seats[i] = ‘1’）并且跳过下一个位置（i++）。

   检查是否是第一个座位或者最后一个座位的目的是确保座位的两侧都是空座。如果当前座位是第一个座位，那么只需要检查右侧是否为空座；如果是最后一个座位，只需要检查左侧是否为空座。这样做是为了确保在座位排列的两端也可以增加额外的观众，因为两端的座位只有一侧有相邻座位，所以需要单独检查。
4. **跳过相邻座位：** 如果一个座位被标记为已坐，需要跳过下一个座位，因此索引 `i` 增加2。否则，只将索引 `i` 增加1，移动到下一个座位。

这个算法的关键在于确保没有两个相邻的座位同时被占据，同时尽可能多地占据空座位。通过在发现一个可用的空座位后立即跳过下一个座位，算法保证了所有被占据的座位都不会相邻。

C++

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    string input;
    cin >> input; // 读取输入的座位信息
    int maxAdditional = 0; // 最大额外观众数初始化为0

    for (int i = 0; i < input.length(); i++) { // 遍历座位数组
        if (input[i] == '0' && (i == 0 || input[i - 1] == '0') && (i == input.length() - 1 || input[i + 1] == '0')) {
            // 如果当前位置是空座且左侧或右侧也是空座，执行以下操作
            maxAdditional++; // 最大额外观众数加1
            input[i] = '1';  // 将当前位置标记为已坐人
            i++;  // 跳过下一个位置，因为已经坐人
        }
    }

    cout << maxAdditional << endl; // 打印最大额外观众数
    return 0;
}
```

Java

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        char[] seats = scanner.nextLine().toCharArray(); // 读取输入的座位信息并转换为字符数组
        int maxAdditional = 0; // 最大额外观众数初始化为0

        for (int i = 0; i < seats.length; i++) { // 遍历座位数组
            if (seats[i] == '0' && (i == 0 || seats[i - 1] == '0') && (i == seats.length - 1 || seats[i + 1] == '0')) {

```

```
                // 如果当前位置是空座且左侧或右侧也是空座，执行以下操作
                maxAdditional++; // 最大额外观众数加1
                seats[i] = '1';  // 将当前位置标记为已坐人
                i++;  // 跳过下一个位置，因为已经坐人
            }
        }

        System.out.println(maxAdditional); // 打印最大额外观众数
    }
}
```

javaScript

```
const readline = require('readline');

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

rl.on('line', (seats) => {
  let maxAdditional = 0; // 最大额外观众数初始化为0

  for (let i = 0; i < seats.length; i++) { // 遍历座位数组
      if (seats[i] === '0' && (i === 0 || seats[i - 1] === '0') && (i === seats.length - 1 || seats[i + 1] === '0')) {
          // 如果当前位置是空座且左侧或右侧也是空座，执行以下操作
          maxAdditional++; // 最大额外观众数加1
          seats = seats.substr(0, i) + '1' + seats.substr(i + 1);  // 将当前位置标记为已坐人
          i++;  // 跳过下一个位置，因为已经坐人
      }
  }

  console.log(maxAdditional); // 打印最大额外观众数
  rl.close();
});
```

Python

```python

seats = input("")  # 读取输入的座位信息
maxAdditional = 0  # 最大额外观众数初始化为0

for i in range(len(seats)):  # 遍历座位数组
    if seats[i] == '0' and (i == 0 or seats[i - 1] == '0') and (i == len(seats) - 1 or seats[i + 1] == '0'):
        # 如果当前位置是空座且左侧或右侧也是空座，执行以下操作
        maxAdditional += 1  # 最大额外观众数加1
        seats = seats[:i] + '1' + seats[i + 1:]  # 将当前位置标记为已坐人
        i += 1  # 跳过下一个位置，因为已经坐人

print(maxAdditional)  # 打印最大额外观众数
```

C语言

```c
#include <stdio.h>
#include <string.h>

int main() {
    char input[10001]; // 创建一个字符数组，用于存储用户输入的座位信息，最大长度10000
    scanf("%s", input); // 读取用户输入的座位信息

    int length = strlen(input); // 获取座位信息字符串的长度
    int maxAdditional = 0; // 初始化最大额外观众数为0

    // 遍历座位数组
    for (int i = 0; i < length; i++) {
        // 如果当前位置是空座且左侧或右侧也是空座
        if (input[i] == '0' && (i == 0 || input[i - 1] == '0') && (i == length - 1 || input[i + 1] == '0')) {
            maxAdditional++; // 最大额外观众数加1
            input[i] = '1';  // 将当前位置标记为已坐人
            i++;  // 跳过下一个位置，因为已经坐人
        }
    }

    printf("%d\n", maxAdditional); // 打印最大额外观众数
    return 0;
}
```

完整用例

用例1

00000

用例2

11111

用例3

10001

用例4

111000

用例5

1

用例6

0

用例7

11

用例8

00

用例9

10

用例10

0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000

文章目录


机考真题
华为OD
No No No
CSDN @算法大师