【华为OD机考 统一考试机试C卷】最少停车数/停车场车辆统计（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

特定大小的停车场，数组cars[]表示，其中1表示有车，0表示没车。车辆大小不一，小车占一个车位（长度1），货车占两个车位（长度2），卡车占三个车位（长度3）。

统计停车场最少可以停多少辆车，返回具体的数目。

输入描述

整型字符串数组cars[]，其中1表示有车，0表示没车，数组长度小于1000。

输出描述

整型数字字符串，表示最少停车数目。

用例

输入	
输出	
说明	

输入	
输出	

说明	

C++

```cpp
#include <iostream>
#include <sstream>
#include <vector>
#include <string>

int main() {
    std::string input;
    std::getline(std::cin, input);

    // 将输入的字符串转换为停车场数组
    std::istringstream iss(input);
    std::string token;
    std::string inputString;
    while (std::getline(iss, token, ',')) {
        inputString += token;
    }
    std::vector<std::string> parking_slots;
    std::istringstream split(inputString);
    while (std::getline(split, token, '0')) {
        parking_slots.push_back(token);
    }

    // 初始化停车场最少停车数目为0
    int min_cars = 0;

    // 遍历停车场数组，统计每个连续的1的长度
    for (const auto& slot : parking_slots) {
        // 计算当前连续1的长度
        int occupied_length = slot.length();

        // 如果当前连续1的长度为0，不做任何操作
        if (occupied_length == 0) {
            min_cars = min_cars;
        }
        // 如果当前连续1的长度能被3整除，说明可以完全放置卡车
        else if (occupied_length % 3 == 0 && occupied_length != 0) {
            // 将当前连续1的长度除以3，得到卡车数量，并累加到最少停车数目
            min_cars += occupied_length / 3;
        }
        // 如果当前连续1的长度不能被3整除，说明需要放置小车或货车
        else if (occupied_length % 3 != 0) {
            // 计算可以放置的卡车数量，并累加到最少停车数目
            min_cars += (occupied_length - occupied_length % 3) / 3;
            // 由于还有剩余的车位，需要放置一个小车或货车，所以最少停车数目加1
            min_cars += 1;

```

```
        }
    }

    // 输出停车场最少停车数目
    std::cout << min_cars << std::endl;
    return 0;
}
```

java

```
import java.util.Scanner;
import java.util.Arrays;

public class Main{
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // 将输入的字符串转换为停车场数组
        String[] inputArray = scanner.nextLine().split(",");
        String inputString = String.join("", inputArray);
        String[] parking_slots = inputString.split("0");

        // 初始化停车场最少停车数目为0
        int min_cars = 0;

        // 遍历停车场数组，统计每个连续的1的长度
        for (String slot : parking_slots) {
            // 计算当前连续1的长度
            int occupied_length = slot.length();

            // 如果当前连续1的长度为0，不做任何操作
            if (occupied_length == 0) {
                min_cars = min_cars;
            }
            // 如果当前连续1的长度能被3整除，说明可以完全放置卡车
            else if (occupied_length % 3 == 0 && occupied_length != 0) {
                // 将当前连续1的长度除以3，得到卡车数量，并累加到最少停车数目
                min_cars += occupied_length / 3;
            }
            // 如果当前连续1的长度不能被3整除，说明需要放置小车或货车
            else if (occupied_length % 3 != 0) {
                // 计算可以放置的卡车数量，并累加到最少停车数目
                min_cars += (occupied_length - occupied_length % 3) / 3;
                // 由于还有剩余的车位，需要放置一个小车或货车，所以最少停车数目加1
                min_cars += 1;
            }
        }

        // 输出停车场最少停车数目
        System.out.println(min_cars);
    }
}

```

javaScript

```
const readline = require('readline').createInterface({
  input: process.stdin,
  output: process.stdout
```

```
});

readline.on('line', (input) => {
  // 将输入的字符串转换为停车场数组
  const inputArray = input.split(',');
  const inputString = inputArray.join('');
  const parking_slots = inputString.split('0');

  // 初始化停车场最少停车数目为0
  let min_cars = 0;

  // 遍历停车场数组，统计每个连续的1的长度
  for (const slot of parking_slots) {
    // 计算当前连续1的长度
    const occupied_length = slot.length;

    // 如果当前连续1的长度为0，不做任何操作
    if (occupied_length === 0) {
      min_cars = min_cars;
    }
    // 如果当前连续1的长度能被3整除，说明可以完全放置卡车
    else if (occupied_length % 3 === 0 && occupied_length !== 0) {
      // 将当前连续1的长度除以3，得到卡车数量，并累加到最少停车数目
      min_cars += Math.floor(occupied_length / 3);
    }
    // 如果当前连续1的长度不能被3整除，说明需要放置小车或货车
    else if (occupied_length % 3 !== 0) {
      // 计算可以放置的卡车数量，并累加到最少停车数目
      min_cars += Math.floor((occupied_length - occupied_length % 3) / 3);
      // 由于还有剩余的车位，需要放置一个小车或货车，所以最少停车数目加1
      min_cars += 1;
    }
  }

  // 输出停车场最少停车数目
  console.log(min_cars);
  readline.close();
});
```

python

```
import sys


parking_slots = ("".join(i for i in (input().split(","))).split("0")

# 初始化停车场最少停车数目为0
min_cars = 0

# 遍历停车场数组，统计每个连续的1的长度
for slot in parking_slots:
    # 计算当前连续1的长度
    occupied_length = len(slot)

    # 如果当前连续1的长度为0，不做任何操作
    if occupied_length == 0:
        min_cars = min_cars
    # 如果当前连续1的长度能被3整除，说明可以完全放置卡车
    elif not occupied_length % 3 and occupied_length != 0:

```

```
        # 将当前连续1的长度除以3，得到卡车数量，并累加到最少停车数目
        min_cars += occupied_length // 3
    # 如果当前连续1的长度不能被3整除，说明需要放置小车或货车
    elif occupied_length % 3:
        # 计算可以放置的卡车数量，并累加到最少停车数目
        min_cars += (occupied_length - occupied_length % 3) // 3
        # 由于还有剩余的车位，需要放置一个小车或货车，所以最少停车数目加1
        min_cars += 1

# 输出停车场最少停车数目
print(int(min_cars))
```

C语言

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

#define MAX_INPUT_LEN 1000

int main() {
    char input[MAX_INPUT_LEN];
    fgets(input, MAX_INPUT_LEN, stdin); // 从标准输入读取字符串

    // 去除输入字符串末尾的换行符
    input[strcspn(input, "\n")] = 0;

    // 初始化停车场最少停车数目为0
    int min_cars = 0;
    char *token = strtok(input, ","); // 使用0作为分隔符分割字符串

    // 遍历分割后的字符串数组，统计每个连续的1的长度
    while (token != NULL) {
        int occupied_length = strlen(token); // 计算当前连续1的长度

        // 如果当前连续1的长度为0，不做任何操作
        if (occupied_length == 0) {
            // 不需要做任何操作
        }
        // 如果当前连续1的长度能被3整除，说明可以完全放置卡车
        else if (occupied_length % 3 == 0) {
            min_cars += occupied_length / 3; // 将当前连续1的长度除以3，得到卡车数量，并累加到最少停车数目
        }
        // 如果当前连续1的长度不能被3整除，说明需要放置小车或货车
        else {
            min_cars += occupied_length / 3; // 计算可以放置的卡车数量，并累加到最少停车数目
            min_cars += 1; // 由于还有剩余的车位，需要放置一个小车或货车，所以最少停车数目加1
        }

        token = strtok(NULL, "0"); // 继续分割字符串
    }

    // 输出停车场最少停车数目
    printf("%d\n", min_cars);
    return 0;
}
```

完整用例

用例1

```
1,0,1
```

用例2

```
1,1,0,0,1,1,1,0,1
```

用例3

```
0,0,0,0,0
```

用例4

```
1,1,1,1,1
```

用例5

```
1,1,1,0,0,0,1,1,1
```

用例6

```
1,1,1,0,1,1,1,0,1,1,1
```

用例7

```
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
```

用例8

```
1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,
```

用例9

```
1,0,0,1,1,0,1,0,0,1,1,1
```

用例10

```
1,0,0,1,1,0,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
```

文章目录

用例2
用例3
用例4
用例5
用例6
用例7
用例8
用例9
用例10