

【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++

```
1  #include <iostream>
2  #include <sstream>
3  #include <vector>
4  #include <string>
5
6  int main() {
7      std::string input;
8      std::getline(std::cin, input);
9
10     // 将输入的字符串转换为停车场数组
11     std::istringstream iss(input);
12     std::string token;
13     std::string inputString;
14     while (std::getline(iss, token, ',')) {
15         inputString += token;
16     }
17     std::vector<std::string> parking_slots;
18     std::istringstream split(inputString);
19     while (std::getline(split, token, '0')) {
20         parking_slots.push_back(token);
21     }
22
23     // 初始化停车场最少停车数目为0
24     int min_cars = 0;
25
26     // 遍历停车场数组, 统计每个连续的1的长度
27     for (const auto& slot : parking_slots) {
28         // 计算当前连续1的长度
29         int occupied_length = slot.length();
30
31         // 如果当前连续1的长度为0, 不做任何操作
32         if (occupied_length == 0) {
33             min_cars = min_cars;
34         }
35         // 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
36         else if (occupied_length % 3 == 0 && occupied_length != 0) {
37             // 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
38             min_cars += occupied_length / 3;
39         }
40         // 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
41         else if (occupied_length % 3 != 0) {
42             // 计算可以放置的卡车数量, 并累加到最少停车数目
43             min_cars += (occupied_length - occupied_length % 3) / 3;
44             // 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
45             min_cars += 1;
46         }
```

```

47     }
48 }
49
50 // 输出停车场最少停车数目
51 std::cout << min_cars << std::endl;
52 return 0;
53 }

```

java

```

1  import java.util.Scanner;
2  import java.util.Arrays;
3
4  public class Main{
5      public static void main(String[] args) {
6          Scanner scanner = new Scanner(System.in);
7
8          // 将输入的字符串转换为停车场数组
9          String[] inputArray = scanner.nextLine().split(",");
10         String inputString = String.join("", inputArray);
11         String[] parking_slots = inputString.split("0");
12
13         // 初始化停车场最少停车数目为0
14         int min_cars = 0;
15
16         // 遍历停车场数组，统计每个连续的1的长度
17         for (String slot : parking_slots) {
18             // 计算当前连续1的长度
19             int occupied_length = slot.length();
20
21             // 如果当前连续1的长度为0，不做任何操作
22             if (occupied_length == 0) {
23                 min_cars = min_cars;
24             }
25             // 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
26             else if (occupied_length % 3 == 0 && occupied_length != 0) {
27                 // 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
28                 min_cars += occupied_length / 3;
29             }
30             // 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
31             else if (occupied_length % 3 != 0) {
32                 // 计算可以放置的卡车数量，并累加到最少停车数目
33                 min_cars += (occupied_length - occupied_length % 3) / 3;
34                 // 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
35                 min_cars += 1;
36             }
37         }
38
39         // 输出停车场最少停车数目
40         System.out.println(min_cars);
41     }
42 }
43

```

javaScript

```

1  const readline = require('readline').createInterface({
2      input: process.stdin,
3      output: process.stdout
4  });

```

```

4  });
5
6  readline.on('line', (input) => {
7    // 将输入的字符串转换为停车场数组
8    const inputArray = input.split(',');
9    const inputString = inputArray.join('');
10   const parking_slots = inputString.split('0');
11
12   // 初始化停车场最少停车数目为0
13   let min_cars = 0;
14
15   // 遍历停车场数组，统计每个连续的1的长度
16   for (const slot of parking_slots) {
17     // 计算当前连续1的长度
18     const occupied_length = slot.length;
19
20     // 如果当前连续1的长度为0，不做任何操作
21     if (occupied_length === 0) {
22       min_cars = min_cars;
23     }
24     // 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
25     else if (occupied_length % 3 === 0 && occupied_length !== 0) {
26       // 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
27       min_cars += Math.floor(occupied_length / 3);
28     }
29     // 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
30     else if (occupied_length % 3 !== 0) {
31       // 计算可以放置的卡车数量，并累加到最少停车数目
32       min_cars += Math.floor((occupied_length - occupied_length % 3) / 3);
33       // 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
34       min_cars += 1;
35     }
36   }
37
38   // 输出停车场最少停车数目
39   console.log(min_cars);
40   readline.close();
41 });
42

```

python

```

1  import sys
2
3
4  parking_slots = ("".join(i for i in (input().split(","))))).split("0")
5
6  # 初始化停车场最少停车数目为0
7  min_cars = 0
8
9  # 遍历停车场数组，统计每个连续的1的长度
10 for slot in parking_slots:
11     # 计算当前连续1的长度
12     occupied_length = len(slot)
13
14     # 如果当前连续1的长度为0，不做任何操作
15     if occupied_length == 0:
16         min_cars = min_cars
17     # 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
18     elif not occupied_length % 3 and occupied_length != 0:
19

```

```

20     # 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
21     min_cars += occupied_length // 3
22     # 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
23     elif occupied_length % 3:
24         # 计算可以放置的卡车数量，并累加到最少停车数目
25         min_cars += (occupied_length - occupied_length % 3) // 3
26         # 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
27         min_cars += 1
28
29 # 输出停车场最少停车数目
30 print(int(min_cars))

```

C语言

```

1  #include <stdio.h>
2  #include <string.h>
3
4  #define MAX_INPUT_LEN 1000
5
6  int main() {
7      char input[MAX_INPUT_LEN];
8      fgets(input, MAX_INPUT_LEN, stdin); // 从标准输入读取字符串
9
10     // 去除输入字符串末尾的换行符
11     input[strcspn(input, "\n")] = 0;
12
13     // 初始化停车场最少停车数目为0
14     int min_cars = 0;
15     char *token = strtok(input, ","); // 使用0作为分隔符分割字符串
16
17     // 遍历分割后的字符串数组，统计每个连续的1的长度
18     while (token != NULL) {
19         int occupied_length = strlen(token); // 计算当前连续1的长度
20
21         // 如果当前连续1的长度为0，不做任何操作
22         if (occupied_length == 0) {
23             // 不需要做任何操作
24         }
25         // 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
26         else if (occupied_length % 3 == 0) {
27             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
28         }
29         // 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
30         else {
31             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
32             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
33         }
34
35         token = strtok(NULL, "0"); // 继续分割字符串
36     }
37
38     // 输出停车场最少停车数目
39     printf("%d\n", min_cars);
40     return 0;
41 }

```

完整用例

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

机考真题 华为OD



CSDN @算法大师