

【华为OD机考 统一考试机试C卷】Wonderland (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

Wonderland是小王居住地一家很受欢迎的游乐园。

Wonderland目前有4种售票方式，分别为

- 一日票 (1天) 、
- 三日票 (3天) 、
- 周票 (7天)
- 月票 (30天) 。

每种售票方式的价格由一个数组给出，每种票据在票面时限内可以无限制地进行游玩。例如：

小王在第10日买了一张三日票，小王可以在第10日、第11日和第12日进行无限制地游玩。

小王计划在接下来一年多次游玩该游乐园。小王计划地游玩日期将由一个数组给出。

现在，请您根据给出地售票价格数组和小王计划游玩日期数组，返回游玩计划所需要地最低消费。

输入描述

输入为2个数组：

- 售票价格数组为costs，costs.length = 4，默认顺序为一日票、三日票、周票和月票。
- 小王计划游玩日期数组为days， $1 \leq \text{days.length} \leq 365$ ， $1 \leq \text{days}[i] \leq 365$ ，默认顺序为升序。

输出描述

完成游玩计划的最低消费。

用例

输入	5 14 30 100 1 3 5 20 21 200 202 230
输出	40
说明	根据售票价格数组和游玩日期数组给出的信息，发现每次去玩的时候买一张一日票是最省钱的，所以小王会卖8张一日票，每张5元，最低花费是40元

解题思路

我们可以使用动态规划。核心思想是为每个游玩日期计算最低消费成本，并存储这些结果以供后续日期使用。具体步骤如下：

- 创建一个数组 `dp` 以存储直至每一天的最低消费。数组的长度为365天加1（因为数组是从0开始的，而天数是从1开始的）。
- 初始化 `dp[0] = 0`，因为在开始之前没有任何消费。
- 遍历每一天，对于每一天，我们有几种选择：
 - 如果这一天不在计划游玩日期中，则这一天的消费与前一天相同，即 `dp[i] = dp[i-1]`。
 - 如果这一天在计划游玩日期中，我们需要考虑三种票价中的最低消费：
 - 一日票的消费是：`dp[i-1] + costs[0]`。
 - 三日票的消费是：`dp[max(i-3, 0)] + costs[1]`。
 - 周票的消费是：`dp[max(i-7, 0)] + costs[2]`。
 - 月票的消费是：`dp[max(i-30, 0)] + costs[3]`。
 - 对于这一天，选择上述四种情况中的最小值作为 `dp[i]` 的值。

4. 最终 `dp[365]` 即为所求的最低消费。

C++

```
1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  #include <sstream>
5
6  using namespace std;
7
8  // 将输入的字符串转换为整数数组
9  vector<int> split(const string &s, char delim) {
10     stringstream ss(s);
11     string item;
12     vector<int> tokens;
13     while (getline(ss, item, delim)) {
14         tokens.push_back(stoi(item));
15     }
16     return tokens;
17 }
18
19 int mincostTickets(vector<int> &costs, vector<int> &days) {
20     // 找到days数组中的最大值, 确定旅游的最后一天
21     int maxDay = *max_element(days.begin(), days.end());
22     // 创建一个长度为maxDay+1的布尔数组, 用于标记每一天是否需要游玩
23     vector<bool> travelDays(maxDay + 1, false);
24     for (int day : days) {
25         travelDays[day] = true;
26     }
27
28     // 创建一个长度为maxDay+1的整数数组, 用于保存每一天的最低消费
29     vector<int> dp(maxDay + 1, 0);
30     for (int i = 1; i <= maxDay; i++) {
31         // 如果这一天不需要游玩, 那么这一天的最低消费就和前一天的最低消费相同
32         if (!travelDays[i]) {
33             dp[i] = dp[i - 1];
34             continue;
35         }
36     }
37 }
```

```
37 // 计算购买各种票后的消费
38 int cost1 = dp[max(0, i - 1)] + costs[0]; // 购买1天的票
39 int cost3 = dp[max(0, i - 3)] + costs[1]; // 购买3天的票
40 int cost7 = dp[max(0, i - 7)] + costs[2]; // 购买7天的票
41 int cost30 = dp[max(0, i - 30)] + costs[3]; // 购买30天的票
42
43 // 这一天的最低消费就是购买各种票后消费的最小值
44 dp[i] = min({cost1, cost3, cost7, cost30});
45 }
46
47 // 返回最后一天的最低消费，即为完成整个游玩计划的最低消费
48 return dp[maxDay];
49 }
50
51 int main() {
52     string costsStr, daysStr;
53     getline(cin, costsStr);
54     getline(cin, daysStr);
55
56     vector<int> costs = split(costsStr, ' ');
57     vector<int> days = split(daysStr, ' ');
58
59     cout << mincostTickets(costs, days) << endl;
60
61     return 0;
62 }
```

Java

```
1 import java.util.Arrays;
2 import java.util.Scanner;
3
4 public class Main {
5     public static void main(String[] args) {
6         // 创建Scanner对象用于获取用户输入
7         Scanner scanner = new Scanner(System.in);
8
9         // 从用户输入中获取票价，输入格式为以空格分隔的四个整数
10        int[] costs = Arrays.stream(scanner.nextLine().split(" "))
11            .mapToInt(Integer::parseInt)
12    }
```

```
12         .toArray();
13
14     // 从用户输入中获取游玩日期, 输入格式为以空格分隔的整数
15     int[] days = Arrays.stream(scanner.nextLine().split(" "))
16         .mapToInt(Integer::parseInt)
17         .toArray();
18
19     // 关闭Scanner对象
20     scanner.close();
21
22     // 调用mincostTickets方法计算最低消费, 并打印结果
23     System.out.println(mincostTickets(costs, days));
24 }
25
26 public static int mincostTickets(int[] costs, int[] days) {
27     // 找到days数组中的最大值, 确定旅游的最后一天
28     int maxDay = Arrays.stream(days).max().getAsInt();
29     // 创建一个长度为maxDay+1的布尔数组, 用于标记每一天是否需要游玩
30     boolean[] travelDays = new boolean[maxDay + 1];
31     for (int day : days) {
32         travelDays[day] = true;
33     }
34
35     // 创建一个长度为maxDay+1的整数数组, 用于保存每一天的最低消费
36     int[] dp = new int[maxDay + 1];
37     for (int i = 1; i <= maxDay; i++) {
38         // 如果这一天不需要游玩, 那么这一天的最低消费就和前一天的最低消费相同
39         if (!travelDays[i]) {
40             dp[i] = dp[i - 1];
41             continue;
42         }
43
44         // 计算购买各种票后的消费
45         int cost1 = dp[Math.max(0, i - 1)] + costs[0]; // 购买1天的票
46         int cost3 = dp[Math.max(0, i - 3)] + costs[1]; // 购买3天的票
47         int cost7 = dp[Math.max(0, i - 7)] + costs[2]; // 购买7天的票
48         int cost30 = dp[Math.max(0, i - 30)] + costs[3]; // 购买30天的票
49
50         // 这一天的最低消费就是购买各种票后消费的最小值
51         dp[i] = Math.min(Math.min(cost1, cost3), Math.min(cost7, cost30));
52     }
53 }
```

```
53     }
54
55     // 返回最后一天的最低消费, 即为完成整个游玩计划的最低消费
56     return dp[maxDay];
57 }
}
```

javaScript

```
1  const readline = require('readline').createInterface({
2    input: process.stdin,
3    output: process.stdout
4  });
5
6  let input = [];
7
8  readline.on('line', (line) => {
9    input.push(line);
10 });
11
12 readline.on('close', () => {
13   const costs = input[0].split(' ').map(Number);
14   const days = input[1].split(' ').map(Number);
15
16   console.log(mincostTickets(costs, days));
17 });
18
19 function mincostTickets(costs, days) {
20   // 找到days数组中的最大值, 确定旅游的最后一天
21   const maxDay = Math.max(...days);
22   // 创建一个长度为maxDay+1的布尔数组, 用于标记每一天是否需要游玩
23   const travelDays = Array(maxDay + 1).fill(false);
24   days.forEach(day => travelDays[day] = true);
25
26   // 创建一个长度为maxDay+1的整数数组, 用于保存每一天的最低消费
27   const dp = Array(maxDay + 1).fill(0);
28   for (let i = 1; i <= maxDay; i++) {
29     // 如果这一天不需要游玩, 那么这一天的最低消费就和前一天的最低消费相同
30     if (!travelDays[i]) {
31       dp[i] = dp[i - 1];
32     }
33   }
34 }
```

```

32         continue;
33     }
34
35     // 计算购买各种票后的消费
36     const cost1 = dp[Math.max(0, i - 1)] + costs[0]; // 购买1天的票
37     const cost3 = dp[Math.max(0, i - 3)] + costs[1]; // 购买3天的票
38     const cost7 = dp[Math.max(0, i - 7)] + costs[2]; // 购买7天的票
39     const cost30 = dp[Math.max(0, i - 30)] + costs[3]; // 购买30天的票
40
41     // 这一天的最低消费就是购买各种票后消费的最小值
42     dp[i] = Math.min(cost1, cost3, cost7, cost30);
43 }
44
45 // 返回最后一天的最低消费，即为完成整个游玩计划的最低消费
46 return dp[maxDay];
47 }

```

Python

```

1 def mincostTickets(costs, days):
2     # 找到days数组中的最大值，确定旅游的最后一天
3     maxDay = max(days)
4     # 创建一个长度为maxDay+1的布尔数组，用于标记每一天是否需要游玩
5     travelDays = [False] * (maxDay + 1)
6     for day in days:
7         travelDays[day] = True
8
9     # 创建一个长度为maxDay+1的整数数组，用于保存每一天的最低消费
10    dp = [0] * (maxDay + 1)
11    for i in range(1, maxDay + 1):
12        # 如果这一天不需要游玩，那么这一天的最低消费就和前一天的最低消费相同
13        if not travelDays[i]:
14            dp[i] = dp[i - 1]
15            continue
16
17        # 计算购买各种票后的消费
18        cost1 = dp[max(0, i - 1)] + costs[0] # 购买1天的票
19        cost3 = dp[max(0, i - 3)] + costs[1] # 购买3天的票
20        cost7 = dp[max(0, i - 7)] + costs[2] # 购买7天的票
21        cost30 = dp[max(0, i - 30)] + costs[3] # 购买30天的票
22
23

```



```

22
23     # 这一天的最低消费就是购买各种票后消费的最小值
24     dp[i] = min(cost1, cost3, cost7, cost30)
25
26     # 返回最后一天的最低消费，即为完成整个游玩计划的最低消费
27     return dp[maxDay]
28
29
30 costs = list(map(int, input().split()))
31 days = list(map(int, input().split()))
32
33 print(mincostTickets(costs, days))

```

C语言

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  #define MAX 366 // 定义最大天数为366天
6
7  // 实现max函数，用于比较两个整数的大小
8  int max(int a, int b) {
9      return a > b ? a : b; // 如果a大于b，返回a，否则返回b
10 }
11
12 // 将输入的字符串转换为整数数组
13 void split(char *s, int *arr, int *len) {
14     char *token = strtok(s, " "); // 使用空格作为分隔符，分割字符串
15     while (token != NULL) { // 当分割得到的字符串不为空时
16         arr[*len] = atoi(token); // 将分割得到的字符串转换为整数，并存储在数组中
17         (*len)++; // 数组长度加1
18         token = strtok(NULL, " "); // 继续分割字符串
19     }
20 }
21
22 // 实现min函数，用于比较两个整数的大小
23 int min(int a, int b) {
24     return a < b ? a : b; // 如果a小于b，返回a，否则返回b
25 }
26

```

```
26
27 // 计算最小花费
28 int mincostTickets(int *costs, int *days, int daysSize) {
29     int maxDay = days[daysSize - 1]; // 找到需要游玩的最后一天
30     int travelDays[MAX] = {0}; // 创建一个数组, 标记每一天是否需要游玩
31     int dp[MAX] = {0}; // 创建一个数组, 保存每一天的最低消费
32     for (int i = 0; i < daysSize; i++) { // 遍历每一天
33         travelDays[days[i]] = 1; // 标记需要游玩的天数
34     }
35
36     for (int i = 1; i <= maxDay; i++) { // 从第一天开始, 计算每一天的最低消费
37         if (!travelDays[i]) { // 如果这一天不需要游玩
38             dp[i] = dp[i - 1]; // 这一天的最低消费和前一天的最低消费相同
39             continue;
40         }
41
42         // 计算购买各种票后的消费
43         int cost1 = dp[max(0, i - 1)] + costs[0]; // 购买1天的票
44         int cost3 = dp[max(0, i - 3)] + costs[1]; // 购买3天的票
45         int cost7 = dp[max(0, i - 7)] + costs[2]; // 购买7天的票
46         int cost30 = dp[max(0, i - 30)] + costs[3]; // 购买30天的票
47
48         // 这一天的最低消费就是购买各种票后消费的最小值
49         dp[i] = min(min(min(cost1, cost3), cost7), cost30);
50     }
51
52     // 返回最后一天的最低消费, 即为完成整个游玩计划的最低消费
53     return dp[maxDay];
54 }
55
56 int main() {
57     char costsStr[100], daysStr[1000]; // 定义两个字符串, 用于接收输入的票价和游玩的天数
58     fgets(costsStr, 100, stdin); // 从标准输入读取票价
59     fgets(daysStr, 1000, stdin); // 从标准输入读取游玩的天数
60
61     int costs[4] = {0}, days[MAX] = {0}; // 定义两个数组, 用于存储票价和游玩的天数
62     int costsSize = 0, daysSize = 0; // 定义两个变量, 用于保存票价和游玩天数的数量
63     split(costsStr, costs, &costsSize); // 将票价字符串转换为整数数组
64     split(daysStr, days, &daysSize); // 将游玩天数字符串转换为整数数组
65
66     --
```

```
67 |     printf("%d\n", mincostTickets(costs, days, daysSize)); // 计算并输出最小花费
68 |
69 |     return 0;
    | }
```

文章目录

华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

题目描述

输入描述

输出描述

用例

解题思路

C++

Java

JavaScript

Python

C语言

机考真题 华为OD



CSDN @算法大师

