

# 【华为OD机考 统一考试机试C卷】剩余银饰的重量（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

有  $N$  块二手市场收集的银饰，每块银饰的重量都是正整数，收集到的银饰会被熔化用于打造新的饰品。每一回合，从中选出三块 最重的 银饰，然后一起熔掉。假设银饰的重量分别为  $x$ 、 $y$  和  $z$ ，且  $x \leq y \leq z$ 。那么熔掉的可能结果如下：

- 如果  $x == y == z$ ，那么三块银饰都会被完全熔掉；
- 如果  $x == y$  且  $y != z$ ，会剩余重量为  $z - y$  的银块无法被熔掉；
- 如果  $x != y$  且  $y == z$ ，会剩余重量为  $y - x$  的银块无法被熔掉；
- 如果  $x != y$  且  $y != z$ ，会剩余重量为  $z - y$  与  $y - x$  差值的银块无法被熔掉。

如果剩余两块，返回较大的重量（若两块重量相同，返回任意一块皆可）；如果只剩下一块，返回该块的重量；如果没有剩下，就返回 0。

## 输入描述

输入数据为两行

第一行为银饰数组长度  $n$  ,  $1 \leq n \leq 40$  ,

第二行为  $n$  块银饰的重量, 重量的取值范围为  $[1, 2000]$  , 重量之间使用空格隔开

输出描述

如果剩余两块, 返回较大的重量 (若两块重量相同, 返回任意一块皆可) ; 如果只剩下一块, 返回该块的重量; 如果没有剩下, 就返回  $0$  。

示例一

输入

1		3
2		1 1 1

输出

1		0
---	--	---

说明

选出  $1\ 1\ 1$  , 得到  $0$  , 最终数组转换为  $[\ ]$  , 最后没有剩下银块, 返回  $0$

示例二

输入

1		3
2		3 7 10

输出

1		1
---	--	---

说明

选出  $3\ 7\ 10$  , 需要计算  $(7-3)$  和  $(10-7)$  的差值, 即  $(7-3)-(10-7)=1$  , 所以数组转换为  $[1]$  , 剩余一块, 返回该块重量, 返回  $1$

## 解题思路

模拟一个银饰熔化的过程。在这个过程中，每次都会选择三块最重的银饰进行熔化，熔化后的结果根据这三块银饰的重量关系有不同的计算方式。这个过程会一直进行，直到剩下的银饰不足三块为止。

程序使用了一个优先队列（PriorityQueue）来存储所有的银饰，优先队列的特性是每次取出的都是队列中最大的元素（在这个程序中，最大的元素就是最重的银饰）。这样就可以保证每次都是选择最重的三块银饰进行熔化。

下面是一个模拟的例子，初始银饰的重量为 2, 2, 3, 5:

1. 首先，将所有银饰的重量添加到优先队列中，队列中的元素为：5, 3, 2, 2。
2. 队列中有四块银饰，所以可以进行熔化。取出最重的三块银饰，分别是 5, 3, 2。这三块银饰的重量都不同，所以剩余的重量为  $|5 - 3| - (3 - 2)| = 1$ 。将剩余的重量 1 添加到队列中，队列中的元素为：2, 1。
3. 队列中只剩下两块银饰，所以不能再进行熔化。程序结束，输出剩余银饰的最大重量，即 2。

所以，对于输入 4, 2, 2, 3, 5，这个程序的输出应该是 2。

## C++

```
1  #include <iostream>
2  #include <vector>
3  #include <algorithm> // 用于 sort
4
5  using namespace std;
6  int main() {
7      int n;
8      // 读取银饰的数量
9      cin >> n;
10
11     // 创建一个整型向量存储银饰的重量
12     vector<int> silverPieces(n);
13     // 循环读取每个银饰的重量
14     for (int i = 0; i < n; ++i) {
15         cin >> silverPieces[i];
16     }
17
18     // 使用 sort 对 vector 进行降序排序
19     sort(silverPieces.begin(), silverPieces.end(), greater<int>());
20
```

```

21 // 当银饰数量大于等于3时, 进行处理
22 while (silverPieces.size() >= 3) {
23     // 取出最重的三块银饰
24     int z = silverPieces[0]; // 最重的银饰
25     int y = silverPieces[1]; // 第二重的银饰
26     int x = silverPieces[2]; // 第三重的银饰
27     // 从向量中移除这三块银饰
28     silverPieces.erase(silverPieces.begin(), silverPieces.begin() + 3);
29
30     // 如果三块银饰重量相同, 则继续下一轮循环
31     if (x == y && y == z) {
32         continue;
33     } else {
34         // 否则计算剩余银饰的重量
35         int remaining;
36         if (x == y && y < z) {
37             // 如果有两块重量相同, 且第三块更重, 则剩余 z - y
38             remaining = z - y;
39         } else if (x < y && y == z) {
40             // 如果有两块重量相同, 且第一块更轻, 则剩余 y - x
41             remaining = y - x;
42         } else {
43             // 如果三块银饰重量都不同, 则计算剩余重量
44             remaining = abs((z - y) - (y - x));
45         }
46         if(remaining !=0){
47             // 将剩余银饰的重量加入向量
48             silverPieces.push_back(remaining);
49         }
50
51         // 再次对向量进行降序排序
52         sort(silverPieces.begin(), silverPieces.end(), greater<int>());
53     }
54 }
55
56 // 如果向量为空, 表示没有银饰剩余, 输出 0
57 if (silverPieces.empty()) {
58     cout << 0 << endl;
59 } else {
60
61

```

```

62 // 否则输出剩余银饰的重量 (向量中的最大值)
63 cout << silverPieces[0] << endl;
64 }
65
66 return 0;
67 }

```

## Java

```

1  import java.util.PriorityQueue;
2  import java.util.Scanner;
3  import java.util.Collections;
4
5  public class Main {
6      public static void main(String[] args) {
7          Scanner scanner = new Scanner(System.in);
8          // 读取银饰的数量
9          int n = scanner.nextInt();
10         // 创建优先队列, 使用反向顺序以便队列头是最大元素
11         PriorityQueue<Integer> silverPieces = new PriorityQueue<>(Collections.reverseOrder());
12
13         // 循环读取每块银饰的重量并添加到优先队列中
14         for (int i = 0; i < n; i++) {
15             silverPieces.add(scanner.nextInt());
16         }
17
18         // 当队列中至少有三块银饰时执行循环
19         while (silverPieces.size() >= 3) {
20             // 取出三块最重的银饰
21             int z = silverPieces.poll(); // 最重的银饰
22             int y = silverPieces.poll(); // 第二重的银饰
23             int x = silverPieces.poll(); // 第三重的银饰
24             System.out.println( z );
25             System.out.println( y );
26             System.out.println( x );
27
28             // 根据题目描述的规则处理银饰
29             if (x == y && y == z) {
30                 // 如果三块银饰重量相同, 则全部熔化, 不剩余
31                 continue;
32             }
33         }
34     }
35 }

```

```

52     } else {
53         int remaining; // 剩余银饰的重量
54         if (x == y && y < z) {
55             // 如果有两块重量相同, 且第三块更重, 则剩余 z - y
56             remaining = z - y;
57         } else if (x < y && y == z) {
58             // 如果有两块重量相同, 且第一块更轻, 则剩余 y - x
59             remaining = y - x;
60         } else {
61             // 如果三块银饰重量都不同, 则计算剩余重量
62             remaining = Math.abs((z - y) - (y - x));
63         }
64         if(remaining != 0){
65             // 将剩余银饰的重量加入队列
66             silverPieces.add(remaining);
67         }
68     }
69 }
70
71 // 如果队列为空, 表示没有银饰剩余, 输出0
72 if (silverPieces.isEmpty()) {
73     System.out.println(0);
74 } else {
75     // 否则输出剩余银饰的重量 (队列中的最大值)
76     System.out.println(silverPieces.peek());
77 }
78 }
79 }
80 }

```

## JavaScript

```

1  const readline = require('readline');
2
3  const rl = readline.createInterface({
4      input: process.stdin,
5      output: process.stdout
6  });
7
8

```

```
8 // 创建一个数组 lines, 用于存储用户输入的每一行
9 let lines = [];
10
11 // 当用户输入一行并按下回车键时, 触发 'line' 事件
12 rl.on('line', (line) => {
13   // 将用户输入的行添加到 lines 数组中
14   lines.push(line);
15
16   // 当 lines 数组中的行数等于 2 时, 开始处理银饰的重量
17   if (lines.length === 2) {
18     // 从 lines 数组的第二行开始, 将输入的银饰重量转换为数字, 并存储在 silverPieces 数组中
19     let silverPieces = lines[1].split(' ').map(Number);
20     // 对 silverPieces 数组进行降序排序
21     silverPieces.sort((a, b) => b - a);
22
23     // 当 silverPieces 数组中至少有三块银饰时, 执行循环
24     while (silverPieces.length >= 3) {
25       // 取出三块最重的银饰
26       let z = silverPieces.shift();
27       let y = silverPieces.shift();
28       let x = silverPieces.shift();
29
30       // 如果三块银饰的重量相同, 则全部熔化, 不剩余
31       if (x === y && y === z) {
32         continue;
33       } else {
34         let remaining; // 剩余银饰的重量
35         // 如果有两块重量相同, 且第三块更重, 则剩余 z - y
36         if (x === y && y < z) {
37           remaining = z - y;
38         }
39         // 如果有两块重量相同, 且第一块更轻, 则剩余 y - x
40         else if (x < y && y === z) {
41           remaining = y - x;
42         }
43         // 如果三块银饰重量都不同, 则计算剩余重量
44         else {
45           remaining = Math.abs((z - y) - (y - x));
46         }
47         if (remaining !== 0) {
48
```

```

49         // 将剩余银饰的重量加入数组
50         silverPieces.push(remaining);
51     }
52
53     // 重新排序
54     silverPieces.sort((a, b) => b - a);
55 }
56 }
57
58 // 如果数组为空, 表示没有银饰剩余, 输出 0
59 if (silverPieces.length === 0) {
60     console.log(0);
61 } else {
62     // 否则输出剩余银饰的重量 (数组中的最大值)
63     console.log(silverPieces[0]);
64 }
65 // 关闭 readline 接口
66 rl.close();
67 }
});

```

## Python

```

1  # 引入 heapq 模块用于维护优先队列
2  import heapq
3
4  # 读取银饰的数量
5  n = int(input())
6  # 存储银饰的重量
7  silverPieces = list(map(int, input().split()))
8
9
10
11 # 对 silverPieces 列表进行降序排序
12 silverPieces.sort(reverse=True)
13
14 # 当 silverPieces 列表中至少有三块银饰时, 执行循环
15 while len(silverPieces) >= 3:
16     # 取出三块最重的银饰
17     z = silverPieces.pop(0)
18
19

```



```

18     y = silverPieces.pop(0)
19     x = silverPieces.pop(0)
20
21     # 如果三块银饰的重量相同, 则全部熔化, 不剩余
22     if x == y and y == z:
23         continue
24     else:
25         # 剩余银饰的重量
26         remaining = 0
27         # 如果有两块重量相同, 且第三块更重, 则剩余 z - y
28         if x == y and y < z:
29             remaining = z - y
30         # 如果有两块重量相同, 且第一块更轻, 则剩余 y - x
31         elif x < y and y == z:
32             remaining = y - x
33         # 如果三块银饰重量都不同, 则计算剩余重量
34         else:
35             remaining = abs((z - y) - (y - x))
36         if remaining != 0:
37             # 将剩余银饰的重量加入列表
38             silverPieces.append(remaining)
39             # 重新排序
40             silverPieces.sort(reverse=True)
41
42     # 如果列表为空, 表示没有银饰剩余, 输出 0
43     if len(silverPieces) == 0:
44         print(0)
45     else:
46         # 否则输出剩余银饰的重量 (列表中的最大值)
47         print(silverPieces[0])

```

## C语言

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define MAX_NUM 40
5
6  // 定义一个比较函数, 用于 qsort 函数
7  int cmp(const void *a, const void *b) {
8

```

```
8     return *(int *)b - *(int *)a;
9 }
10
11 int main() {
12     int n;
13     // 读取银饰的数量
14     scanf("%d", &n);
15
16     // 创建一个数组存储银饰的重量
17     int silverPieces[MAX_NUM];
18     // 循环读取每个银饰的重量
19     for (int i = 0; i < n; ++i) {
20         scanf("%d", &silverPieces[i]);
21     }
22
23     // 使用 qsort 对数组进行降序排序
24     qsort(silverPieces, n, sizeof(int), cmp);
25
26     // 当银饰数量大于等于3时, 进行处理
27     while (n >= 3) {
28         // 取出最重的三块银饰
29         int z = silverPieces[0]; // 最重的银饰
30         int y = silverPieces[1]; // 第二重的银饰
31         int x = silverPieces[2]; // 第三重的银饰
32         // 从数组中移除这三块银饰
33         for (int i = 0; i < n - 3; ++i) {
34             silverPieces[i] = silverPieces[i + 3];
35         }
36         n -= 3;
37
38         // 如果三块银饰重量相同, 则继续下一轮循环
39         if (x == y && y == z) {
40             continue;
41         } else {
42             // 否则计算剩余银饰的重量
43             int remaining;
44             if (x == y && y < z) {
45                 // 如果有两块重量相同, 且第三块更重, 则剩余 z - y
46                 remaining = z - y;
47             } else if (x < y && y == z) {
```

```

49         // 如果有两块重量相同，且第一块更轻，则剩余 y - x
50         remaining = y - x;
51     } else {
52         // 如果三块银饰重量都不同，则计算剩余重量
53         remaining = abs((z - y) - (y - x));
54     }
55     if(remaining !=0){
56         // 将剩余银饰的重量加入数组
57         silverPieces[n++] = remaining;
58     }
59
60     // 再次对数组进行降序排序
61     qsort(silverPieces, n, sizeof(int), cmp);
62 }
63 }
64
65 // 如果数组为空，表示没有银饰剩余，输出 0
66 if (n == 0) {
67     printf("0\n");
68 } else {
69     // 否则输出剩余银饰的重量（数组中的最大值）
70     printf("%d\n", silverPieces[0]);
71 }
72
73 return 0;
}

```

## 完整用例

### 用例1

```

1 | 5
2 | 1 1 1 1 1

```

### 用例2

```

1 | 5
2 | 1 2 3 4 5

```

用例3

1	5
2	5 4 3 2 1

用例4

1	6
2	10 15 20 25 30 35

用例5

1	6
2	2000 2000 2000 10 5 1

用例6

1	6
2	2000 2000 1999 10 5 1

用例7

1	6
2	2000 1999 1999 10 5 1

用例8

1	1
2	1234

用例9

1	3
2	3 7 10

用例10

1	5
2	1 1 2000 2000 200

## 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

示例一

**\*\*示例二\*\***

解题思路

C++

Java

javaScript

Python

C语言

完整用例

用例1

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