【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 <= y <= 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 ≤ n ≤ 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

```
3
1 1 1
```

输出

```
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

```
3
3 7 10
```

输出

```
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++

```cpp
#include <iostream>
#include <vector>
#include <algorithm> // 用于 sort

using namespace std;
int main() {
    int n;
    // 读取银饰的数量
    cin >> n;

    // 创建一个整型向量存储银饰的重量
    vector<int> silverPieces(n);
    // 循环读取每个银饰的重量
    for (int i = 0; i < n; ++i) {
        cin >> silverPieces[i];
    }

    // 使用 sort 对 vector 进行降序排序
    sort(silverPieces.begin(), silverPieces.end(), greater<int>());

```

```cpp

    // 当银饰数量大于等于3时，进行处理
    while (silverPieces.size() >= 3) {
        // 取出最重的三块银饰
        int z = silverPieces[0]; // 最重的银饰
        int y = silverPieces[1]; // 第二重的银饰
        int x = silverPieces[2]; // 第三重的银饰
        // 从向量中移除这三块银饰
        silverPieces.erase(silverPieces.begin(), silverPieces.begin() + 3);

        // 如果三块银饰重量相同，则继续下一轮循环
        if (x == y && y == z) {
            continue;
        } else {
            // 否则计算剩余银饰的重量
            int remaining;
            if (x == y && y < z) {
                // 如果有两块重量相同，且第三块更重，则剩余 z - y
                remaining = z - y;
            } else if (x < y && y == z) {
                // 如果有两块重量相同，且第一块更轻，则剩余 y - x
                remaining = y - x;
            } else {
                // 如果三块银饰重量都不同，则计算剩余重量
                remaining = abs((z - y) - (y - x));
            }
            if(remaining !=0){
                        // 将剩余银饰的重量加入向量
                silverPieces.push_back(remaining);
            }

            // 再次对向量进行降序排序
            sort(silverPieces.begin(), silverPieces.end(), greater<int>());
        }
    }

    // 如果向量为空，表示没有银饰剩余，输出 0
    if (silverPieces.empty()) {
        cout << 0 << endl;
    } else {

```

```
        // 否则输出剩余银饰的重量（向量中的最大值）
        cout << silverPieces[0] << endl;
    }

    return 0;
}
```

Java

```java
import java.util.PriorityQueue;
import java.util.Scanner;
import java.util.Collections;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // 读取银饰的数量
        int n = scanner.nextInt();
        // 创建优先队列，使用反向顺序以便队列头是最大元素
        PriorityQueue<Integer> silverPieces = new PriorityQueue<>(Collections.reverseOrder());

        // 循环读取每块银饰的重量并添加到优先队列中
        for (int i = 0; i < n; i++) {
            silverPieces.add(scanner.nextInt());
        }

        // 当队列中至少有三块银饰时执行循环
        while (silverPieces.size() >= 3) {
            // 取出三块最重的银饰
            int z = silverPieces.poll(); // 最重的银饰
            int y = silverPieces.poll(); // 第二重的银饰
            int x = silverPieces.poll(); // 第三重的银饰
                        System.out.println( z   );
                        System.out.println(  y   );
                        System.out.println(  x );

            // 根据题目描述的规则处理银饰
            if (x == y && y == z) {
                // 如果三块银饰重量相同，则全部熔化，不剩余
                continue;
```

```
            } else {
                int remaining; // 剩余银饰的重量
                if (x == y && y < z) {
                    // 如果有两块重量相同，且第三块更重，则剩余 z - y
                    remaining = z - y;
                } else if (x < y && y == z) {
                    // 如果有两块重量相同，且第一块更轻，则剩余 y - x
                    remaining = y - x;
                } else {
                    // 如果三块银饰重量都不同，则计算剩余重量
                    remaining = Math.abs((z - y) - (y - x));
                }
                if(remaining != 0){
                    // 将剩余银饰的重量加入队列
                    silverPieces.add(remaining);
                }

            }
        }

        // 如果队列为空，表示没有银饰剩余，输出0
        if (silverPieces.isEmpty()) {
            System.out.println(0);
        } else {
            // 否则输出剩余银饰的重量（队列中的最大值）
            System.out.println(silverPieces.peek());
        }
    }
}
```

javaScript

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

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

```

```
// 创建一个数组 lines，用于存储用户输入的每一行
let lines = [];

// 当用户输入一行并按下回车键时，触发 'line' 事件
rl.on('line', (line) => {
    // 将用户输入的行添加到 lines 数组中
    lines.push(line);

    // 当 lines 数组中的行数等于 2 时，开始处理银饰的重量
    if (lines.length === 2) {
        // 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
        let silverPieces = lines[1].split(' ').map(Number);
        // 对 silverPieces 数组进行降序排序
        silverPieces.sort((a, b) => b - a);

        // 当 silverPieces 数组中至少有三块银饰时，执行循环
        while (silverPieces.length >= 3) {
            // 取出三块最重的银饰
            let z = silverPieces.shift();
            let y = silverPieces.shift();
            let x = silverPieces.shift();

            // 如果三块银饰的重量相同，则全部熔化，不剩余
            if (x === y && y === z) {
                continue;
            } else {
                let remaining; // 剩余银饰的重量
                // 如果有两块重量相同，且第三块更重，则剩余 z - y
                if (x === y && y < z) {
                    remaining = z - y;
                }
                // 如果有两块重量相同，且第一块更轻，则剩余 y - x
                else if (x < y && y === z) {
                    remaining = y - x;
                }
                // 如果三块银饰重量都不同，则计算剩余重量
                else {
                    remaining = Math.abs((z - y) - (y - x));
                }
                if (remaining !== 0) {

```

```
                    // 将剩余银饰的重量加入数组
                    silverPieces.push(remaining);
                }

                // 重新排序
                silverPieces.sort((a, b) => b - a);
            }
        }

        // 如果数组为空，表示没有银饰剩余，输出 0
        if (silverPieces.length === 0) {
            console.log(0);
        } else {
            // 否则输出剩余银饰的重量（数组中的最大值）
            console.log(silverPieces[0]);
        }
        // 关闭 readline 接口
        rl.close();
    }
});
```

Python

```
# 引入 heapq 模块用于维护优先队列
import heapq

# 读取银饰的数量
n = int(input(''))
# 存储银饰的重量
silverPieces = list(map(int, input().split()))



# 对 silverPieces 列表进行降序排序
silverPieces.sort(reverse=True)

# 当 silverPieces 列表中至少有三块银饰时，执行循环
while len(silverPieces) >= 3:
    # 取出三块最重的银饰
    z = silverPieces.pop(0)
```

```
    y = silverPieces.pop(0)
    x = silverPieces.pop(0)

    # 如果三块银饰的重量相同，则全部熔化，不剩余
    if x == y and y == z:
        continue
    else:
        # 剩余银饰的重量
        remaining = 0
        # 如果有两块重量相同，且第三块更重，则剩余 z - y
        if x == y and y < z:
            remaining = z - y
        # 如果有两块重量相同，且第一块更轻，则剩余 y - x
        elif x < y and y == z:
            remaining = y - x
        # 如果三块银饰重量都不同，则计算剩余重量
        else:
            remaining = abs((z - y) - (y - x))
        if remaining != 0:
            # 将剩余银饰的重量加入列表
            silverPieces.append(remaining)
        # 重新排序
        silverPieces.sort(reverse=True)

# 如果列表为空，表示没有银饰剩余，输出 0
if len(silverPieces) == 0:
    print(0)
else:
    # 否则输出剩余银饰的重量（列表中的最大值）
    print(silverPieces[0])
```

C语言

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

#define MAX_NUM 40

// 定义一个比较函数，用于 qsort 函数
int cmp(const void *a, const void *b) {
```

```
    return *(int *)b - *(int *)a;
}

int main() {
    int n;
    // 读取银饰的数量
    scanf("%d", &n);

    // 创建一个数组存储银饰的重量
    int silverPieces[MAX_NUM];
    // 循环读取每个银饰的重量
    for (int i = 0; i < n; ++i) {
        scanf("%d", &silverPieces[i]);
    }

    // 使用 qsort 对数组进行降序排序
    qsort(silverPieces, n, sizeof(int), cmp);

    // 当银饰数量大于等于3时，进行处理
    while (n >= 3) {
        // 取出最重的三块银饰
        int z = silverPieces[0]; // 最重的银饰
        int y = silverPieces[1]; // 第二重的银饰
        int x = silverPieces[2]; // 第三重的银饰
        // 从数组中移除这三块银饰
        for (int i = 0; i < n - 3; ++i) {
            silverPieces[i] = silverPieces[i + 3];
        }
        n -= 3;

        // 如果三块银饰重量相同，则继续下一轮循环
        if (x == y && y == z) {
            continue;
        } else {
            // 否则计算剩余银饰的重量
            int remaining;
            if (x == y && y < z) {
                // 如果有两块重量相同，且第三块更重，则剩余 z - y
                remaining = z - y;
            } else if (x < y && y == z) {

```

```
                // 如果有两块重量相同，且第一块更轻，则剩余 y - x
                remaining = y - x;
            } else {
                // 如果三块银饰重量都不同，则计算剩余重量
                remaining = abs((z - y) - (y - x));
            }
            if(remaining !=0){
                // 将剩余银饰的重量加入数组
                silverPieces[n++] = remaining;
            }

            // 再次对数组进行降序排序
            qsort(silverPieces, n, sizeof(int), cmp);
        }
    }

    // 如果数组为空，表示没有银饰剩余，输出 0
    if (n == 0) {
        printf("0\n");
    } else {
        // 否则输出剩余银饰的重量（数组中的最大值）
        printf("%d\n", silverPieces[0]);
    }

    return 0;
}
```

完整用例

用例1

```
5
1 1 1 1 1
```

用例2

```
5
1 2 3 4 5
```

用例3

```
5
5 4 3 2 1
```

用例4

```
6
10 15 20 25 30 35
```

用例5

```
6
2000 2000 2000 10 5 1
```

用例6

```
6
2000 2000 1999 10 5 1
```

用例7

```
6
2000 1999 1999 10 5 1
```

用例8

```
1
1234
```

用例9

```
3
3 7 10
```

用例10

```
5
1 1 2000 2000 200
```

文章目录


机考真题
华为OD
No No No
CSDN @算法大师