【华为OD机考 统一考试机试C卷】CPU算力分配（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

现有两组服务器A和B，每组有多个算力不同的CPU，其中 A[i] 是 A 组第 i 个CPU的运算能力，B[i] 是 B组 第 i 个CPU的运算能力。

一组服务器的总算力是各CPU的算力之和。

为了让两组服务器的算力相等，允许从每组各选出一个CPU进行一次交换，

求两组服务器中，用于交换的CPU的算力，并且要求从A组服务器中选出的CPU，算力尽可能小。

输入描述

第一行输入为L1和L2，以空格分隔，L1表示A组服务器中的CPU数量，L2表示B组服务器中的CPU数量。

第二行输入为A组服务器中各个CPU的算力值，以空格分隔。

第三行输入为B组服务器中各个CPU的算力值，以空格分隔。

$1 \le L1 \le 10000$
$1 \le L2 \le 10000$

1 ≤ A[i] ≤ 100000
1 ≤ B[i] ≤ 100000

输出描述

对于每组测试数据，输出两个整数，以空格分隔，依次表示A组选出的CPU算力，B组选出的CPU算力。

要求从A组选出的CPU的算力尽可能小。

备注

- 保证两组服务器的初始总算力不同。
- 答案肯定存在

用例1

输入	**2 2** **1 1** **2 2**
输出	1 2
说明	从A组中选出算力为1的CPU，与B组中算力为2的进行交换，使两组服务器的算力都等于3。

用例2

输入	**2 2** **1 2** **2 3**
输出	1 2

用例3

输入	1 2 2 1 3
输出	2 3

用例4

输入	3 2 1 2 5 2 4
输出	5 4

解题思路

这个问题的目标是通过交换A组和B组的服务器，使得两组的总算力尽可能接近。为了达到这个目标，我们需要找到一对服务器，使得交换后两组的总算力差最小。

假设A组的总算力为a，B组的总算力为b，且a > b。我们希望找到A组的一个服务器，其算力为x，和B组的一个服务器，其算力为y，使得交换后的总算力差为(a - x + y) - (b - y + x) = a - b - 2(x - y)尽可能小。也就是说，我们希望x - y尽可能接近(a - b) / 2，也就是两组总算力差的一半。

因此，我们需要计算两组总算力差的一半，然后在A组的服务器中找到一个算力，使得它减去这个值后的结果在B组的服务器中存在。这样，我们就找到了一对可以交换的服务器，使得交换后两组的总算力尽可能接近。

C++

```cpp
#include <iostream>
#include <vector>
#include <map>
#include <algorithm>

using namespace std;

int main() {
    // 读取A组和B组的服务器数量
    int serverCountGroupA, serverCountGroupB;
    cin >> serverCountGroupA >> serverCountGroupB;
```

```cpp

    // 初始化A组的总算力为0
    int totalPowerGroupA = 0;
    // 创建数组存储A组的服务器算力
    vector<int> powerGroupA(serverCountGroupA);
    // 读取A组的服务器算力，并计算总算力
    for (int i = 0; i < serverCountGroupA; i++) {
        cin >> powerGroupA[i];
        totalPowerGroupA += powerGroupA[i];
    }

    // 初始化B组的总算力为0
    int totalPowerGroupB = 0;
    // 创建map存储B组的服务器算力和对应的数量
    map<int, int> powerCountGroupB;
    // 读取B组的服务器算力，并计算总算力
    for (int i = 0; i < serverCountGroupB; i++) {
        int power;
        cin >> power;
        totalPowerGroupB += power;
        powerCountGroupB[power]++;
    }

    // 计算两组总算力差的一半
    int halfDifference = (totalPowerGroupA - totalPowerGroupB) / 2;

    // 对A组的服务器算力进行排序
    sort(powerGroupA.begin(), powerGroupA.end());
    // 从小到大遍历A组的服务器
    for (int powerA : powerGroupA) {
        // 计算需要在B组中找到的服务器算力
        int powerB = powerA - halfDifference;

        // 如果B组中存在这样的服务器，并且数量大于0
        if (powerCountGroupB.count(powerB) && powerCountGroupB[powerB] > 0) {
            // 输出A组和B组选出的服务器的算力
            cout << powerA << " " << powerB << endl;
            break;
        }
    }

```

```
53
54      return 0;
}
```

Java

```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        // 创建扫描器读取输入
        Scanner scanner = new Scanner(System.in);

        // 读取A组和B组的服务器数量
        int serverCountGroupA = scanner.nextInt();
        int serverCountGroupB = scanner.nextInt();

        // 初始化A组的总算力为0
        int totalPowerGroupA = 0;
        // 创建数组存储A组的服务器算力
        int[] powerGroupA = new int[serverCountGroupA];
        // 读取A组的服务器算力，并计算总算力
        for (int i = 0; i < serverCountGroupA; i++) {
            powerGroupA[i] = scanner.nextInt();
            totalPowerGroupA += powerGroupA[i];
        }

        // 初始化B组的总算力为0
        int totalPowerGroupB = 0;
        // 创建HashMap存储B组的服务器算力和对应的数量
        HashMap<Integer, Integer> powerCountGroupB = new HashMap<>();
        // 读取B组的服务器算力，并计算总算力
        for (int i = 0; i < serverCountGroupB; i++) {
            int power = scanner.nextInt();
            totalPowerGroupB += power;
            powerCountGroupB.put(power, powerCountGroupB.getOrDefault(power, 0) + 1);
        }

        // 计算两组总算力差的一半，四舍五入取整
        int halfDifference = (int)Math.round((totalPowerGroupA - totalPowerGroupB) / 2.0);
```

```
        // 对A组的服务器算力进行排序
        Arrays.sort(powerGroupA);
        // 从小到大遍历A组的服务器
        for (int powerA : powerGroupA) {
            // 计算需要在B组中找到的服务器算力
            int powerB = powerA - halfDifference;

            // 如果B组中存在这样的服务器，并且数量大于0
            if (powerCountGroupB.containsKey(powerB) && powerCountGroupB.get(powerB) > 0) {
                // 输出A组和B组选出的服务器的算力
                System.out.println(powerA + " " + powerB);
                break;
            }
        }
    }
}
```

javaScript

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

let lines = [];

// 将输入行存储在数组中
readline.on('line', line => {
    lines.push(line);
});

// 在所有输入行都读取完毕后处理这些行
readline.on('close', () => {
    // 读取A组和B组的服务器数量
    const [serverCountGroupA, serverCountGroupB] = lines[0].split(' ').map(Number);

    // 初始化A组的总算力为0
    let totalPowerGroupA = 0;
    // 创建数组存储A组的服务器算力
```

```
    let powerGroupA = lines[1].split(' ').map(Number);
    // 计算A组的服务器算力总和
    totalPowerGroupA = powerGroupA.reduce((a, b) => a + b, 0);

    // 初始化B组的总算力为0
    let totalPowerGroupB = 0;
    // 创建map存储B组的服务器算力和对应的数量
    let powerCountGroupB = new Map();
    // 读取B组的服务器算力，并计算总算力
    const powers = lines[2].split(' ').map(Number);
    for (let power of powers) {
        totalPowerGroupB += power;
        powerCountGroupB.set(power, (powerCountGroupB.get(power) || 0) + 1);
    }

    // 计算两组总算力差的一半，四舍五入取整
    let halfDifference = Math.round((totalPowerGroupA - totalPowerGroupB) / 2);

    // 对A组的服务器算力进行排序
    powerGroupA.sort((a, b) => a - b);
    // 从小到大遍历A组的服务器
    for (let powerA of powerGroupA) {
        // 计算需要在B组中找到的服务器算力
        let powerB = powerA - halfDifference;

        // 如果B组中存在这样的服务器，并且数量大于0
        if (powerCountGroupB.has(powerB) && powerCountGroupB.get(powerB) > 0) {
            // 输出A组和B组选出的服务器的算力
            console.log(powerA + " " + powerB);
            break;
        }
    }
});
```

Python

```
# 导入需要的库
import sys

# 读取A组和B组的服务器数量
```

```python
serverCountGroupA, serverCountGroupB = map(int, sys.stdin.readline().split())

# 初始化A组的总算力为0
totalPowerGroupA = 0
# 读取A组的服务器算力，并计算总算力
powerGroupA = list(map(int, sys.stdin.readline().split()))
totalPowerGroupA = sum(powerGroupA)

# 初始化B组的总算力为0
totalPowerGroupB = 0
# 创建字典存储B组的服务器算力和对应的数量
powerCountGroupB = {}
# 读取B组的服务器算力，并计算总算力
powers = list(map(int, sys.stdin.readline().split()))
for power in powers:
    totalPowerGroupB += power
    powerCountGroupB[power] = powerCountGroupB.get(power, 0) + 1

# 计算两组总算力差的一半，四舍五入取整
halfDifference = round((totalPowerGroupA - totalPowerGroupB) / 2)

# 对A组的服务器算力进行排序
powerGroupA.sort()
# 从小到大遍历A组的服务器
for powerA in powerGroupA:
    # 计算需要在B组中找到的服务器算力
    powerB = powerA - halfDifference

    # 如果B组中存在这样的服务器，并且数量大于0
    if powerB in powerCountGroupB and powerCountGroupB[powerB] > 0:
        # 输出A组和B组选出的服务器的算力
        print(powerA, powerB)
        break
```

C语言

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

// 比较函数，用于整数数组排序
```

```c
int compare(const void *a, const void *b) {
    return (*(int *)a - *(int *)b);
}

int main() {
    int L1, L2;
    // 读取A组和B组的服务器数量
    scanf("%d %d", &L1, &L2);

    int A[L1], B[L2];
    int totalPowerA = 0, totalPowerB = 0;

    // 读取A组的服务器算力，并计算总算力
    for (int i = 0; i < L1; i++) {
        scanf("%d", &A[i]);
        totalPowerA += A[i];
    }

    // 读取B组的服务器算力，并计算总算力
    for (int i = 0; i < L2; i++) {
        scanf("%d", &B[i]);
        totalPowerB += B[i];
    }

    // 计算两组总算力差的一半
    int halfDifference = (totalPowerA - totalPowerB) / 2;

    // 对A组的服务器算力进行排序
    qsort(A, L1, sizeof(int), compare);

    // 使用哈希表记录B组的服务器算力
    int hash[100001] = {0};
    for (int i = 0; i < L2; i++) {
        hash[B[i]] = 1;
    }

    // 从小到大遍历A组的服务器
    for (int i = 0; i < L1; i++) {
        int powerA = A[i];
        // 计算需要在B组中找到的服务器算力
```

```
        int powerB = powerA - halfDifference;

        // 如果B组中存在这样的服务器
        if (powerB >= 1 && powerB <= 100000 && hash[powerB]) {
            // 输出A组和B组选出的服务器的算力
            printf("%d %d\n", powerA, powerB);
            break;
        }
    }

    return 0;
}
```

完整用例

用例1

3 3

1 2 3

4 5 6

用例2

2 2

10 20

10 20

用例3

3 3

1 2 3

10 20 30

用例4

4 4

5 10 15 20

30 25 20 15

用例5

2 2
99999 100000
99998 99997

用例6

2 2
1 2
3 4

用例7

5 5
17 23 42 50 60
22 35 37 41 55

用例8

3 3
20 20 20
10 15 25

用例9

3 3
10 15 25
20 20 20

用例10

3 2
1 2 5
2 4

文章目录


机考真题
华为OD
No No No
CSDN @算法大师