【华为OD机考 统一考试机试C卷】 游戏分组/王者荣耀（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

2020年题：

英雄联盟是一款十分火热的对战类游戏。每一场对战有10位玩家参与，分为两组，每组5人。每位玩家都有一个战斗力，代表着这位玩家的厉害程度。为了对战尽可能精彩，我们需要把玩家们分为实力尽量相等的两组。一组的实力可以表示为这一组5位玩家的战斗力和。现在，给你10位玩家的战斗力，请你把他们分为实力尽量相等的两组。请你输出这两组的实力差。

2023年题：

部门准备举办一场王者荣耀表演赛，有10名游戏爱好者参与，分5为两队，每队5人。每位参与者都有一个评分，代表着他的游戏水平。为了表演赛尽可能精彩，我们需要把10名参赛者分为实力尽量相近的两队。一队的实力可以表示为这一队5名队员的评分总和。
现在给你10名参与者的游戏水平评分，请你根据上述要求分队最后输出这两组的实力差绝对值。
例: 10名参赛者的评分分别为5 1 8 3 4 6 710 9 2，分组为 (135 8 10) (24 679)，两组实力差最小，差值为1。有多种分法，但实力差的绝对值最小为1。

输入描述

10个整数，表示10名参与者的游戏水平评分。范围在[1,10000]之间

输出描述

1个整数，表示分组后两组实力差绝对值的最小值.

用例1

输入：

```
1 2 3 4 5 6 7 8 9 10
```

输出：

```
1
```

说明：

> 10名队员分成两组，两组实力差绝对值最小为1.

解题思路

C++

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>
#include <climits>

using namespace std;

int res = INT_MAX;
int totalSum = 0;
int targetSum = 0;

// 深度优先搜索函数
void dfs(vector<int>& nums, int idx, int count, int currentSum) {
    // 剪枝条件：如果当前总和超过目标，则停止 ,考友反馈，去掉可得100%

```

```
    // if (currentSum > targetSum) return;

    // 当我们为一个队伍选择了5名玩家时
    if (count == 5) {
        // 计算另一个队伍的总和
        int otherTeamSum = totalSum - currentSum;
        // 用较小的差值更新结果
        res = min(res, abs(currentSum - otherTeamSum));
        return;
    }

    // 如果我们已经考虑了所有玩家，停止递归
    if (idx == 10) return;

    // 为第一个队伍选择当前玩家
    dfs(nums, idx + 1, count + 1, currentSum + nums[idx]);

    // 不为第一个队伍选择当前玩家
    dfs(nums, idx + 1, count, currentSum);
}

int main() {
    vector<int> nums(10);
    for (int i = 0; i < 10; ++i) {
        cin >> nums[i];
        totalSum += nums[i];
    }
    targetSum = totalSum / 2;
    dfs(nums, 0, 0, 0);
    cout << res << endl;
    return 0;
}
```

Java

```
import java.util.*;

public class Main {
    static int res = Integer.MAX_VALUE;
    static int totalSum = 0;
```

```
    static int targetSum = 0;

    public static void main(String[] args) {
        Scanner cin = new Scanner(System.in);
        int[] nums = Arrays.stream(cin.nextLine().split(" "))
                .mapToInt(Integer::parseInt).toArray();
        for (int num : nums) {
            totalSum += num;
        }
        targetSum = totalSum / 2;
        dfs(nums, 0, 0, 0);
        System.out.println(res);
        cin.close();
    }

    static void dfs(int[] nums, int idx, int count, int currentSum) {
        // 剪枝条件：如果当前总和超过目标，则停止.考友反馈，去掉可得100%
        // if (currentSum > targetSum) return;

        // 当我们为一个队伍选择了5名玩家时
        if (count == 5) {
            // 计算另一个队伍的总和
            int otherTeamSum = totalSum - currentSum;
            // 用较小的差值更新结果
            res = Math.min(res, Math.abs(currentSum - otherTeamSum));
            return;
        }

        // 如果我们已经考虑了所有玩家，停止递归
        if (idx == 10) return;

        // 为第一个队伍选择当前玩家
        dfs(nums, idx + 1, count + 1, currentSum + nums[idx]);

        // 不为第一个队伍选择当前玩家
        dfs(nums, idx + 1, count, currentSum);
    }
}
```

javaScript

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

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

let res = Number.MAX_SAFE_INTEGER;
let totalSum = 0;
let targetSum = 0;

// 深度优先搜索函数
function dfs(nums, idx, count, currentSum) {
    // 剪枝条件：如果当前总和超过目标，则停止 考友反馈，去掉可得100%
    // if (currentSum > targetSum) return;

    // 当我们为一个队伍选择了5名玩家时
    if (count === 5) {
        // 计算另一个队伍的总和
        let otherTeamSum = totalSum - currentSum;
        // 用较小的差值更新结果
        res = Math.min(res, Math.abs(currentSum - otherTeamSum));
        return;
    }

    // 如果我们已经考虑了所有玩家，停止递归
    if (idx === 10) return;

    // 为第一个队伍选择当前玩家
    dfs(nums, idx + 1, count + 1, currentSum + nums[idx]);

    // 不为第一个队伍选择当前玩家
    dfs(nums, idx + 1, count, currentSum);
}

rl.on('line', (input) => {
    let nums = input.split(' ').map(Number);
    for (let num of nums) {
        totalSum += num;
    }
```

```
    targetSum = totalSum / 2;
    dfs(nums, 0, 0, 0);
    console.log(res);
    rl.close();
});
```

Python

```python
import sys

res = sys.maxsize
totalSum = 0
targetSum = 0

# 深度优先搜索函数
def dfs(nums, idx, count, currentSum):
    global res, totalSum, targetSum
    # 剪枝条件：如果当前总和超过目标，则停止.考友反馈，去掉可得100%
    # if currentSum > targetSum:
    #    return

    # 当我们为一个队伍选择了5名玩家时
    if count == 5:
        # 计算另一个队伍的总和
        otherTeamSum = totalSum - currentSum
        # 用较小的差值更新结果
        res = min(res, abs(currentSum - otherTeamSum))
        return

    # 如果我们已经考虑了所有玩家，停止递归
    if idx == 10:
        return

    # 为第一个队伍选择当前玩家
    dfs(nums, idx + 1, count + 1, currentSum + nums[idx])

    # 不为第一个队伍选择当前玩家
    dfs(nums, idx + 1, count, currentSum)

nums = list(map(int, input().split()))
```

```
for num in nums:
    totalSum += num
targetSum = totalSum // 2
dfs(nums, 0, 0, 0)
print(res)
```

C语言

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

int res = INT_MAX;
int totalSum = 0;
int targetSum = 0;

// 深度优先搜索函数
void dfs(int nums[10], int idx, int count, int currentSum) {
    // 剪枝条件：如果当前总和超过目标，则停止.考友反馈，去掉可得100%
    // if (currentSum > targetSum) return;

    // 当我们为一个队伍选择了5名玩家时
    if (count == 5) {
        // 计算另一个队伍的总和
        int otherTeamSum = totalSum - currentSum;
        // 用较小的差值更新结果
        res = abs(currentSum - otherTeamSum) < res ? abs(currentSum - otherTeamSum) : res;
        return;
    }

    // 如果我们已经考虑了所有玩家，停止递归
    if (idx == 10) return;

    // 为第一个队伍选择当前玩家
    dfs(nums, idx + 1, count + 1, currentSum + nums[idx]);

    // 不为第一个队伍选择当前玩家
    dfs(nums, idx + 1, count, currentSum);
}

```

```
int main() {
    int nums[10];
    for (int i = 0; i < 10; ++i) {
        scanf("%d", &nums[i]);
        totalSum += nums[i];
    }
    targetSum = totalSum / 2;
    dfs(nums, 0, 0, 0);
    printf("%d\n", res);
    return 0;
}
```

完整用例

用例1

```
1 1 1 1 1 1 1 1 1 1
```

用例2

```
1 2 3 4 5 6 7 8 9 10
```

用例3

```
10 9 8 7 6 5 4 3 2 1
```

用例4

```
1 1 1 1 1 10000 10000 10000 10000 10000
```

用例5

```
4500 4600 4700 4800 4900 5100 5200 5300 5400 5500
```

用例6

```
1000 1000 1000 1000 1000 9000 9000 9000 9000 9000
```

用例7

```
1234 5678 910 1112 1314 1516 789 2345 6789 34
```

用例8

```
1000 900 800 700 600 500 400 300 200 100
```

用例9

```
1000 3000 5000 7000 9000 2000 4000 6000 8000 10000
```

用例10

```
5000 5000 4000 4000 3000 3000 2000 2000 1000 1000
```

文章目录

用例6
用例7
用例8
用例9
用例10