华为OD机考算法题：MVP争夺战

御剑乐逍遥　已于 2023-09-15 12:22:24 修改


华为OD机考　专栏收录该内容　1 订阅　11 篇文章

目录

题目部分

题目　MVP争夺战

难度　易

题目说明　在星球争霸篮球赛对抗赛中，强大的宇宙战队，希望每个人都能拿到MVP。
MVP的条件是，单场最高分得分获得者，可以并列，所以宇宙战队决定在比赛中，尽可能让更多的队员上场，且让所有有得分的队员得分都相同。
然而比赛过程中的每一分钟的得分都只能由某一个人包揽。

输入描述　输入第一行为一个数字t，表示有得分的分钟数（ 1 <= t <= 50 ）。
第二行为t个数字，代表每一分钟的得分 p（ 1 <= p <= 50 ）。

输出描述　输出有得分的队员都是MVP时最少的MVP得分。

补充说明　无

示例

内容来源：csdn.net
作者昵称：御剑乐逍遥
原文链接：https://athena.blog.csdn.net/article/details/132807904
作者主页：https://athena.blog.csdn.net

示例1

输入　9
5 2 1 5 2 1 5 2 1

输出　6

说明　样例解释：一共4人得分，分别都为6分
5 + 1
5 + 1
5 + 1
2 + 2 + 2

解读与分析

题目解读：

输入一系列数字代表球员的得分，要求把这些分数尽可能分给多的球员，且保证每个球员的得分之和相等。求出当平均分配的人数最多时，这个平均得分是多少。

换种说法，就是把指定的一组数字分成若干组（分成多少组不确定，每组的数字个数也不固定），使每组数字之和相等。求这个和的最小值。和最小，意味着分成的组数最多。

题目中给出了 n 个数字，要求把这 n 个数字划分成 m 组，保证 m 组中每组的数字之和相等，即每组的数字之和等于 n / m（注： n/m 取整）。

需要注意：每组的数字个数并不是固定的，可能各不相同。

分析与思路：

虽然此题难度标识为“易”，但思路和方法却不那么容易。

我能想到最好的办法是动态规划，尝试把 n 个数字放到不同的组中。假设 n 个数中最大的值为 max，那么组的个数的取值范围是 [1, n / max]。

我们采用递归的方式，尝试把某个数字放到一个组中，然后在此前提下，使用递归尝试把剩下的数字放进某一组，以此穷尽所有可能的情形。

从分组数为 (n / max 取整) 开始尝试，如果满足分组条件，此时的分组数就是最大分组数，返回此时每组的数字之和，退出程序。如果不满足，分组数减 1，继续尝试，直至分组数为 1。
满足分组的条件是：每组的数字之和相等，且所有数字都归属于某一组。

此题最后一定会有结果。在最坏的情况下，所有的数字都在同一个组。在题目中代表着着，只有一个MVP球员，这个球员包揽了所有的得分。

内容来源：csdn.net
作者昵称：御剑乐逍遥
作者主页：https://athena.blog.csdn.net

此算法用到了递归，会遍历可能的分组数情况。在每一次分组，穷尽所有数字放到各个组，所需的时间复杂度 O(n^2)，空间复杂度 O(n)。

代码实现

Java代码

```java
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;


/**
 * MVP争夺战
 *
 * @since 2023.09.11
 * @version 0.1
 * @author Frank
 *
 */
public class MVPCompetition {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (sc.hasNext()) {
            String input = sc.nextLine();
            int count = Integer.parseInt( input );
            input = sc.nextLine();
            String[] numbers = input.split( " " );
            processMVPCompetition( numbers );
        }
    }

    private static void processMVPCompetition( String numbers[] )
    {
        int sum = 0;
        int maxNum = 0;
        List<Integer> numList = new ArrayList<Integer>();
        for( int i = 0; i < numbers.length; i ++ )
        {
            int tmpNum = Integer.parseInt( numbers[i] );
            if( tmpNum > maxNum )
            {
```

内容来源：csdn.net
作者昵称：御剑乐逍遥
原文链接：https://athena.blog.csdn.net/article/details/132807904
作者主页：https://athena.blog.csdn.net

```
                maxNum = tmpNum;
            }
            sum += tmpNum;
            numList.add( tmpNum );
        }

        int maxMVPCnt = sum / maxNum;
        for( int i = maxMVPCnt; i >= 1; i --)
        {
            if( sum % i != 0 )
            {
                continue;
            }
            int aveScroe = sum / i;

            int[] tmpSum = new int[ i ];
            for( int j = 0; j < tmpSum.length; j ++ )
            {
                tmpSum[j] = 0;
            }
            int ret = processAverageScroe( aveScroe, tmpSum, numList );
            if( ret != -1 )
            {
                System.out.println( ret );
                return;
            }
        }

    }

    private static int processAverageScroe( int score, int[] tmpSum, List<Integer> numbers)
    {
        int ret = -1;

        int tmpNum = numbers.get( 0 );
        numbers.remove( 0 );

        for( int i = 0; i < tmpSum.length; i ++ )
        {
            if( tmpNum + tmpSum[i] > score )
            {
                continue;
            }
```

内容来源：csdn.net
作者昵称：御剑乐逍遥
原文链接：https://athena.blog.csdn.net/article/details/132807904
作者主页：https://athena.blog.csdn.net

```
            tmpSum[i] = tmpSum[i] + tmpNum;
            boolean meet = isArrayAllScore( score, tmpSum, numbers );
            if( meet )
            {
                return score;
            }
            ret = processAverageScroe( score, tmpSum, numbers);
            if( ret != -1 )
            {
                return ret;
            }
            tmpSum[i] = tmpSum[i] - tmpNum;
        }

        numbers.add( 0, tmpNum );
        return ret;
    }

    private static boolean isArrayAllScore( int score, int[] tmpSum, List<Integer> numbers )
    {
        boolean ret = true;
        if( numbers.size() > 0 )
        {
            return false;
        }
        for( int i = 0; i < tmpSum.length; i ++ )
        {
            if( tmpSum[i] != score )
            {
                return false;
            }
        }
        return ret;
    }

}
```

JavaScript代码

```
const rl = require("readline").createInterface({ input: process.stdin });
var iter = rl[Symbol.asyncIterator]();
```

内容来源：csdn.net
作者昵称：御剑乐逍遥
原文链接：https://athena.blog.csdn.net/article/details/132807904
作者主页：https://athena.blog.csdn.net

```
const readline = async () => (await iter.next()).value;
    while (line = await readline()) {
        // count 可以忽略
        var count = parseInt(line);
        line = await readline();
        var numberArr = line.split(" ");
        processMVPCompetition(numberArr);
    }
}();

function processMVPCompetition(numbers) {
    var sum = 0;
    var maxNum = 0;
    var numList = new Array();
    for (var i = 0; i < numbers.length; i++) {
        var tmpNum = parseInt(numbers[i]);
        if (tmpNum > maxNum) {
            maxNum = tmpNum;
        }
        sum += tmpNum;
        numList.push(tmpNum);
    }

    var maxMVPCnt = parseInt( sum / maxNum );
    for (var i = maxMVPCnt; i >= 1; i--) {

        if (sum % i != 0) {
            continue;
        }
        var aveScroe = sum / i;

        var tmpSum = new Array();
        for (var j = 0; j < i; j++) {
            tmpSum[j] = 0;
        }

        var ret = processAverageScroe(aveScroe, tmpSum, numList);
        if (ret != -1) {
            console.log(ret);
            return;
        }
    }
```

```
4 void async function() {
```

内容来源：csdn.net
作者昵称：御剑乐逍遥
原文链接：https://athena.blog.csdn.net/article/details/132807904
作者主页：https://athena.blog.csdn.net

```
}

function processAverageScroe( score, tmpSum, numbers) {
    var ret = -1;

    var tmpNum = numbers.shift(0);
    for (var i = 0; i < tmpSum.length; i++) {
        if (tmpNum + tmpSum[i] > score) {
            continue;
        }
        tmpSum[i] = tmpSum[i] + tmpNum;
        var meet = isArrayAllScore(score, tmpSum, numbers);
        if (meet) {
            return score;
        }
        ret = processAverageScroe(score, tmpSum, numbers);
        if (ret != -1) {
            return ret;
        }
        tmpSum[i] = tmpSum[i] - tmpNum;
    }

    numbers.unshift( tmpNum );
    return ret;
}

function isArrayAllScore(score, tmpSum, numbers) {
    var ret = true;
    if (numbers.length > 0) {
        return false;
    }
    for (var i = 0; i < tmpSum.length; i++) {
        if (tmpSum[i] != score) {
            return false;
        }
    }
    return ret;
}
```

(完)

内容来源：csdn.net
作者昵称：御剑乐逍遥
原文链接：https://athena.blog.csdn.net/article/details/132807904
作者主页：https://athena.blog.csdn.net

文章知识点与官方知识档案匹配，可进一步学习相关知识

算法技能树 > 首页 > 概览　51562 人正在系统学习中

内容来源：csdn.net
作者昵称：御剑乐逍遥
原文链接：https://athena.blog.csdn.net/article/details/132807904
作者主页：https://athena.blog.csdn.net