

【华为OD机考 统一考试机试C卷】部门人力分配 (C++ Java JavaScript Python)

华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

2023年11月份，华为官方已经将 华为OD机考：OD统一考试（A卷 / B卷）切换到 OD统一考试（C卷）和 OD统一考试（D卷）。根据考友反馈：目前抽到的试卷为B卷或C卷/D卷，其中C卷居多，按照之前的经验C卷D卷部分考题会复用A卷/B卷题，博主正积极从考过的同学收集C卷和D卷真题，可以查看下面的真题目录。

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 华为OD机考B卷C卷华为OD机考华为OD机考B卷华为OD机试B卷华为OD机试C卷华为OD机考C卷华为OD机考D卷题目华为OD机考C卷/D卷答案华为OD机考C卷/D卷解析华为OD机考C卷和D卷真题华为OD机考C卷和D卷题解

题目描述

部门在进行需求开发时需要进行人力安排。

当前部门需要完成 N 个需求，需求用 requirements 表述，requirements[i] 表示第 i 个需求的工作量大小，单位：人月。

这部分需求需要在 M 个月内完成开发，进行人力安排后每个月人力时固定的。

目前要求每个月最多有2个需求开发，并且每个月需要完成的需求不能超过部门人力。

请帮助部门评估在满足需求开发进度的情况下，每个月需要的最小人力是多少？

输入描述

输入为 M 和 requirements， M 表示需求开发时间要求，requirements 表示每个需求工作量大小， N 为 requirements 长度，

- $1 \leq N/2 \leq M \leq N \leq 10000$
- $1 \leq requirements[i] \leq 10^9$

输出描述

对于每一组测试数据，输出部门需要人力需求，行末无多余的空格

用例

输入

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

输出

```
1 | 6
```

说明

输入数据两行，
第一行输入数据3表示开发时间要求，
第二行输入数据表示需求工作量大小，
输出数据一行，表示部门人力需求。

当选择人力为6时，2个需求量为3的工作可以在1个月里完成，其他2个工作各需要1个月完成。可以在3个月内完成所有需求。

当选择人力为5时，4个工作各需要1个月完成，一共需要4个月才能完成所有需求。

因此6是部门最小的人力需求。

解题思路

题目描述了一个关于人力资源规划的问题。具体来说，有以下要点：

- 需求开发任务**：存在 N 个需求开发任务，每个任务的工作量用一个数组 `requirements` 来表示，其中 `requirements[i]` 是第 i 个需求的工作量。
- 时间限制**：所有的需求必须在 M 个月内完成。
- 人力安排**：每个月的人力是固定的，并且每个月最多只能开发两个需求。
- 人力限制**：每个月完成的需求工作量总和不能超过可用的人力。

题目要求计算在满足上述条件的情况下，每个月所需的最小人力是多少。

C++

```
1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  #include <numeric>
5
6  int main() {
7      // 创建输入流对象用于读取输入
8      std::ios_base::sync_with_stdio(false);
9      std::cin.tie(NULL);
10
11     // 读取第一行输入, 表示需求开发时间要求
12     int m;
13     std::cin >> m;
14     std::cin.ignore(); // 忽略换行符
15
16     // 读取第二行输入, 表示每个需求的工作量大小
17     std::vector<int> requirements;
18     int work;
19     while (std::cin >> work) {
20         requirements.push_back(work);
21     }
22
23     // 对需求工作量进行排序
24     std::sort(requirements.begin(), requirements.end());
25     // 初始化二分查找的左边界为最大的需求工作量
26     int left = requirements.back();
27     // 初始化二分查找的右边界为所有需求工作量之和除以最小月数加一
28     int sum = std::accumulate(requirements.begin(), requirements.end(), 0);
29     int right = sum / (m / 2) + 1;
30
31     // 进行二分查找以确定最小人力
32     while (left < right) {
33         // 计算中间值
34         int mid = left + (right - left) / 2;
35         // 初始化所需月数
36         int monthsNeeded = 0;
37         // 遍历每个需求, 判断是否可以在限定人力下完成
38         for (int i = requirements.size() - 1, j = 0; i >= j; --i) {
39             // 如果当前需求大于中间人力值, 则增加左边界
40             if (requirements[i] > mid) {
```

```
41         left = mid + 1;
42         break;
43     }
44     // 如果当前和下一个需求之和大于中间人力值, 或者只剩一个需求, 则增加所需月数
45     if (i == j || requirements[i] + requirements[j] > mid) {
46         monthsNeeded++;
47     } else {
48         // 否则, 将下一个需求也计算在当前月份内, 并增加所需月数
49         j++;
50         monthsNeeded++;
51     }
52     // 如果所需月数大于限定月数, 则增加左边界
53     if (monthsNeeded > m) {
54         left = mid + 1;
55         break;
56     }
57 }
58 // 如果所需月数小于等于限定月数, 则减小右边界
59 if (monthsNeeded <= m) {
60     right = mid;
61 }
62 }
63 // 输出最小人力需求
64 std::cout << left << std::endl;
65
66 return 0;
67 }
```

Java

```
1 import java.util.Arrays;
2 import java.util.Scanner;
3
4 public class Main {
5     public static void main(String[] args) {
6         // 创建Scanner对象用于读取输入
7         Scanner sc = new Scanner(System.in);
8         // 读取第一行输入, 表示需求开发时间要求
9         int m = Integer.parseInt(sc.nextLine());
10        // 读取第二行输入, 表示每个需求的工作量大小, 并转换为整数数组
11    }
```

```
11 int[] requirements = Arrays.stream(sc.nextLine().split(" ")).mapToInt(Integer::parseInt).toArray();
12
13 // 对需求工作量进行排序
14 Arrays.sort(requirements);
15 // 初始化二分查找的左边界为最大的需求工作量
16 int left = requirements[requirements.length - 1];
17 // 初始化二分查找的右边界为所有需求工作量之和除以最小月数加一
18 int right = Arrays.stream(requirements).sum() / (m / 2) + 1;
19 // 进行二分查找以确定最小人力
20 while (left < right) {
21     // 计算中间值
22     int mid = left + (right - left) / 2;
23     // 初始化所需月数
24     int monthsNeeded = 0;
25     // 遍历每个需求, 判断是否可以在限定人力下完成
26     for (int i = requirements.length - 1, j = 0; i >= j; --i) {
27         // 如果当前需求大于中间人力值, 则增加左边界
28         if (requirements[i] > mid) {
29             left = mid + 1;
30             break;
31         }
32         // 如果当前和下一个需求之和大于中间人力值, 或者只剩一个需求, 则增加所需月数
33         if (i == j || requirements[i] + requirements[j] > mid) {
34             monthsNeeded++;
35         } else {
36             // 否则, 将下一个需求也计算在当前月份内, 并增加所需月数
37             j++;
38             monthsNeeded++;
39         }
40         // 如果所需月数大于限定月数, 则增加左边界
41         if (monthsNeeded > m) {
42             left = mid + 1;
43             break;
44         }
45     }
46     // 如果所需月数小于等于限定月数, 则减小右边界
47     if (monthsNeeded <= m) {
48         right = mid;
49     }
50 }
51 }
```

```
52 |         // 输出最小人力需求
53 |         System.out.println(left);
54 |     }
    | }
```

JavaScript

```
1 | // 引入readline模块用于读取输入
2 | const readline = require('readline');
3 | const rl = readline.createInterface({
4 |     input: process.stdin,
5 |     output: process.stdout
6 | });
7 |
8 | // 读取输入
9 | let lines = [];
10 | rl.on('line', (line) => {
11 |     lines.push(line);
12 | }).on('close', () => {
13 |     // 第一行输入, 表示需求开发时间要求
14 |     const m = parseInt(lines[0]);
15 |     // 第二行输入, 表示每个需求的工作量大小
16 |     const requirements = lines[1].split(' ').map(Number);
17 |
18 |     // 对需求工作量进行排序
19 |     requirements.sort((a, b) => a - b);
20 |     // 初始化二分查找的左边界为最大的需求工作量
21 |     let left = requirements[requirements.length - 1];
22 |     // 初始化二分查找的右边界为所有需求工作量之和除以最小月数加一
23 |     let right = Math.floor(requirements.reduce((a, b) => a + b, 0) / (m / 2)) + 1;
24 |
25 |     // 进行二分查找以确定最小人力
26 |     while (left < right) {
27 |         // 计算中间值
28 |         const mid = Math.floor((left + right) / 2);
29 |         // 初始化所需月数
30 |         let monthsNeeded = 0;
31 |         // 遍历每个需求, 判断是否可以在限定人力下完成
32 |         for (let i = requirements.length - 1, j = 0; i >= j; --i) {
33 |             // 如果当前需求大于中间人力值, 则增加左边界
34 |             if (requirements[i] > mid) {
35 |                 left = mid + 1;
36 |                 continue;
37 |             }
38 |             monthsNeeded += Math.ceil(requirements[i] / mid);
39 |             if (monthsNeeded > (m / 2)) {
40 |                 right = mid;
41 |                 continue;
42 |             }
43 |             j = i;
44 |         }
45 |         if (monthsNeeded > (m / 2)) {
46 |             right = mid;
47 |         } else {
48 |             left = mid;
49 |         }
50 |     }
51 |     console.log(left);
52 | }
```

```
34     if (requirements[i] > mid) {
35         left = mid + 1;
36         break;
37     }
38     // 如果当前和下一个需求之和大于中间人力值, 或者只剩一个需求, 则增加所需月数
39     if (i == j || requirements[i] + requirements[j] > mid) {
40         monthsNeeded++;
41     } else {
42         // 否则, 将下一个需求也计算在当前月份内, 并增加所需月数
43         j++;
44         monthsNeeded++;
45     }
46     // 如果所需月数大于限定月数, 则增加左边界
47     if (monthsNeeded > m) {
48         left = mid + 1;
49         break;
50     }
51 }
52 // 如果所需月数小于等于限定月数, 则减小右边界
53 if (monthsNeeded <= m) {
54     right = mid;
55 }
56 }
57 // 输出最小人力需求
58 console.log(left);
59 };
```

Python

```
1 # 引入sys模块用于读取输入
2 import sys
3
4 # 读取第一行输入, 表示需求开发时间要求
5 m = int(sys.stdin.readline().strip())
6 # 读取第二行输入, 表示每个需求的工作量大小, 并转换为整数列表
7 requirements = list(map(int, sys.stdin.readline().strip().split()))
8
9 # 对需求工作量进行排序
10 requirements.sort()
11 # 初始化二分查找的左边界为最大的需求工作量
12 left = requirements[-1]
```



```
12 left = requirements[-1]
13 # 初始化二分查找的右边界为所有需求工作量之和除以最小月数加一
14 right = sum(requirements) // (m // 2) + 1
15
16 # 进行二分查找以确定最小人力
17 while left < right:
18     # 计算中间值
19     mid = (left + right) // 2
20     # 初始化所需月数
21     months_needed = 0
22     # 遍历每个需求, 判断是否可以在限定人力下完成
23     i, j = len(requirements) - 1, 0
24     while i >= j:
25         # 如果当前需求大于中间人力值, 则增加左边界
26         if requirements[i] > mid:
27             left = mid + 1
28             break
29         # 如果当前和下一个需求之和大于中间人力值, 或者只剩一个需求, 则增加所需月数
30         if i == j or requirements[i] + requirements[j] > mid:
31             months_needed += 1
32         else:
33             # 否则, 将下一个需求也计算在当前月份内, 并增加所需月数
34             j += 1
35             months_needed += 1
36         i -= 1
37         # 如果所需月数大于限定月数, 则增加左边界
38         if months_needed > m:
39             left = mid + 1
40             break
41         # 如果所需月数小于等于限定月数, 则减小右边界
42         if months_needed <= m:
43             right = mid
44
45 # 输出最小人力需求
46 print(left)
```

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

用例

解题思路

C++

Java

JavaScript

Python

机考真题 华为OD



CSDN @算法大师