【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 ≤ N/2 ≤ M ≤ N ≤ 10000
- 1 ≤ requirements[i] ≤ 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

```
3
3 5 3 4
```

输出

```
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：

1. **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。
2. **时间限制**：所有的需求必须在 M 个月内完成。
3. **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。
4. **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++

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

int main() {
    // 创建输入流对象用于读取输入
    std::ios_base::sync_with_stdio(false);
    std::cin.tie(NULL);

    // 读取第一行输入，表示需求开发时间要求
    int m;
    std::cin >> m;
    std::cin.ignore(); // 忽略换行符

    // 读取第二行输入，表示每个需求的工作量大小
    std::vector<int> requirements;
    int work;
    while (std::cin >> work) {
        requirements.push_back(work);
    }

    // 对需求工作量进行排序
    std::sort(requirements.begin(), requirements.end());
    // 初始化二分查找的左边界为最大的需求工作量
    int left = requirements.back();
    // 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
    int sum = std::accumulate(requirements.begin(), requirements.end(), 0);
    int right = sum / (m / 2) + 1;

    // 进行二分查找以确定最小人力
    while (left < right) {
        // 计算中间值
        int mid = left + (right - left) / 2;
        // 初始化所需月数
        int monthsNeeded = 0;
        // 遍历每个需求，判断是否可以在限定人力下完成
        for (int i = requirements.size() - 1, j = 0; i >= j; --i) {
            // 如果当前需求大于中间人力值，则增加左边界
            if (requirements[i] > mid) {
```

```
                left = mid + 1;
                break;
            }
            // 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
            if (i == j || requirements[i] + requirements[j] > mid) {
                monthsNeeded++;
            } else {
                // 否则，将下一个需求也计算在当前月份内，并增加所需月数
                j++;
                monthsNeeded++;
            }
            // 如果所需月数大于限定月数，则增加左边界
            if (monthsNeeded > m) {
                left = mid + 1;
                break;
            }
        }
        // 如果所需月数小于等于限定月数，则减小右边界
        if (monthsNeeded <= m) {
            right = mid;
        }
    }
    // 输出最小人力需求
    std::cout << left << std::endl;

    return 0;
}
```

Java

```
import java.util.Arrays;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        // 创建Scanner对象用于读取输入
        Scanner sc = new Scanner(System.in);
        // 读取第一行输入，表示需求开发时间要求
        int m = Integer.parseInt(sc.nextLine());
        // 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
```

```
        int[] requirements = Arrays.stream(sc.nextLine().split(" ")).mapToInt(Integer::parseInt).toArray();

        // 对需求工作量进行排序
        Arrays.sort(requirements);
        // 初始化二分查找的左边界为最大的需求工作量
        int left = requirements[requirements.length - 1];
        // 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
        int right = Arrays.stream(requirements).sum() / (m / 2) + 1;
        // 进行二分查找以确定最小人力
        while (left < right) {
            // 计算中间值
            int mid = left + (right - left) / 2;
            // 初始化所需月数
            int monthsNeeded = 0;
            // 遍历每个需求，判断是否可以在限定人力下完成
            for (int i = requirements.length - 1, j = 0; i >= j; --i) {
                // 如果当前需求大于中间人力值，则增加左边界
                if (requirements[i] > mid) {
                    left = mid + 1;
                    break;
                }
                // 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
                if (i == j || requirements[i] + requirements[j] > mid) {
                    monthsNeeded++;
                } else {
                    // 否则，将下一个需求也计算在当前月份内，并增加所需月数
                    j++;
                    monthsNeeded++;
                }
                // 如果所需月数大于限定月数，则增加左边界
                if (monthsNeeded > m) {
                    left = mid + 1;
                    break;
                }
            }
            // 如果所需月数小于等于限定月数，则减小右边界
            if (monthsNeeded <= m) {
                right = mid;
            }
        }

```

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

javaScript

```javascript
// 引入readline模块用于读取输入
const readline = require('readline');
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

// 读取输入
let lines = [];
rl.on('line', (line) => {
  lines.push(line);
}).on('close', () => {
  // 第一行输入，表示需求开发时间要求
  const m = parseInt(lines[0]);
  // 第二行输入，表示每个需求的工作量大小
  const requirements = lines[1].split(' ').map(Number);

  // 对需求工作量进行排序
  requirements.sort((a, b) => a - b);
  // 初始化二分查找的左边界为最大的需求工作量
  let left = requirements[requirements.length - 1];
  // 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
  let right = Math.floor(requirements.reduce((a, b) => a + b, 0) / (m / 2)) + 1;

  // 进行二分查找以确定最小人力
  while (left < right) {
    // 计算中间值
    const mid = Math.floor((left + right) / 2);
    // 初始化所需月数
    let monthsNeeded = 0;
    // 遍历每个需求，判断是否可以在限定人力下完成
    for (let i = requirements.length - 1, j = 0; i >= j; --i) {
      // 如果当前需求大于中间人力值，则增加左边界
```

```
      if (requirements[i] > mid) {
        left = mid + 1;
        break;
      }
      // 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
      if (i == j || requirements[i] + requirements[j] > mid) {
        monthsNeeded++;
      } else {
        // 否则，将下一个需求也计算在当前月份内，并增加所需月数
        j++;
        monthsNeeded++;
      }
      // 如果所需月数大于限定月数，则增加左边界
      if (monthsNeeded > m) {
        left = mid + 1;
        break;
      }
    }
    // 如果所需月数小于等于限定月数，则减小右边界
    if (monthsNeeded <= m) {
      right = mid;
    }
  }
  // 输出最小人力需求
  console.log(left);
});
```

Python

```
# 引入sys模块用于读取输入
import sys

# 读取第一行输入，表示需求开发时间要求
m = int(sys.stdin.readline().strip())
# 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
requirements = list(map(int, sys.stdin.readline().strip().split()))

# 对需求工作量进行排序
requirements.sort()
# 初始化二分查找的左边界为最大的需求工作量
```

```
left = requirements[-1]
# 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
right = sum(requirements) // (m // 2) + 1

# 进行二分查找以确定最小人力
while left < right:
    # 计算中间值
    mid = (left + right) // 2
    # 初始化所需月数
    months_needed = 0
    # 遍历每个需求，判断是否可以在限定人力下完成
    i, j = len(requirements) - 1, 0
    while i >= j:
        # 如果当前需求大于中间人力值，则增加左边界
        if requirements[i] > mid:
            left = mid + 1
            break
        # 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
        if i == j or requirements[i] + requirements[j] > mid:
            months_needed += 1
        else:
            # 否则，将下一个需求也计算在当前月份内，并增加所需月数
            j += 1
            months_needed += 1
        i -= 1
        # 如果所需月数大于限定月数，则增加左边界
        if months_needed > m:
            left = mid + 1
            break
    # 如果所需月数小于等于限定月数，则减小右边界
    if months_needed <= m:
        right = mid

# 输出最小人力需求
print(left)
```

文章目录