【华为OD机考 统一考试机试C卷】 爱吃蟠桃的孙悟空（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

孙悟空爱吃蟠桃，有一天趁着蟠桃园守卫不在来偷吃。已知蟠桃园有 N 棵桃树，每颗树上都有桃子，守卫将在 H 小时后回来。

孙悟空可以决定他吃蟠桃的速度K（个/小时），每个小时选一颗桃树，并从树上吃掉 K 个，如果树上的桃子少于 K 个，则全部吃掉，并且这一小时剩余的时间里不再吃桃。

孙悟空喜欢慢慢吃，但又想在守卫回来前吃完桃子。

请返回孙悟空可以在 H 小时内吃掉所有桃子的最小速度 K（K为整数）。如果以任何速度都吃不完所有桃子，则返回0。

输入描述

第一行输入为 N 个数字，N 表示桃树的数量，这 N 个数字表示每颗桃树上蟠桃的数量。

第二行输入为一个数字，表示守卫离开的时间 H。

其中数字通过空格分割，N、H为正整数，每颗树上都有蟠桃，且 0 < N < 10000，0 < H < 10000。

输出描述

吃掉所有蟠桃的最小速度 K，无解或输入异常时输出 0。

用例1

输入	2 3 4 5 4
输出	5
说明	无

用例2

输入	2 3 4 5 3
输出	0
说明	无

解题思路

本题22年考过！！！

本题原题：https://leetcode.cn/problems/koko-eating-bananas/

C++

```cpp
#include <iostream>
#include <vector>
#include <cmath>
#include <algorithm>
#include <sstream>
using namespace std;

// 判断以速度k是否能在h小时内吃完所有桃子
bool canFinish(vector<int>& p, int h, int k) {
    long long ans = 0; // 使用 long long 防止溢出
    for (int x : p) {
```

```cpp
        ans += ceil(x * 1.0 / k); // 向上取整
    }
    return ans <= h;
}

int main() {
    // 读取输入
    string line;
    getline(cin, line);
    istringstream iss(line);
    vector<int> peachCounts;
    int x;
    while (iss >> x) {
        peachCounts.push_back(x);
    }
    int h;
    cin >> h;

    // 输入验证
    int n = peachCounts.size();
    if (n == 0 || h <= 0 || n >= 10000 || h >= 10000 || n > h) {
        cout << 0 << endl;
        return 0;
    }

    // 二分查找最小吃桃速度
    int left = 1, right = 1e9; // 假设最大的吃桃速度不会超过1e9
    while (left < right) {
        int mid = left + (right - left) / 2;
        if (canFinish(peachCounts, h, mid)) {
            right = mid;
        } else {
            left = mid + 1;
        }
    }

    // 输出最小吃桃速度
    cout << left << endl;
    return 0;
}
```

Java

```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        // 创建一个Scanner对象用于读取输入
        Scanner cin = new Scanner(System.in);
        // 读取一行输入并转换为整数数组，代表每棵桃树上的桃子数量
        int[] peachCounts = Arrays.stream(cin.nextLine().split(" ")).mapToInt(Integer::parseInt).toArray();
        // 读取下一行输入，转换为整数，代表可用的小时数
        int h = Integer.parseInt(cin.nextLine());
        // 获取桃树的数量
        int n = peachCounts.length;

        // 输入验证：如果桃树数量为0，或小时数不合法，或桃树数量大于小时数，则输出0并返回
        if (n == 0 || h <= 0 || n >= 10000 || h >= 10000 || n > h) {
            System.out.println(0);
            return;
        }

        // 初始化二分查找的左右边界
        int left = 1, right = (int)1e9; // 假设最大的吃桃速度不会超过1e9
        // 当左边界小于右边界时，执行二分查找
        while (left < right) {
            // 计算中间值
            int mid = left + (right - left) / 2;
            // 如果以mid的速度可以在h小时内吃完桃子，则尝试更小的速度
            if (canFinish(peachCounts, h, mid)) {
                right = mid;
            } else {
                // 否则尝试更大的速度
                left = mid + 1;
            }
        }

        // 输出最小吃桃速度，此时left是满足条件的最小速度
        System.out.println(left);
    }


```

```
    // 定义一个方法，判断以速度k是否能在h小时内吃完所有桃子
    static boolean canFinish(int[] p, int h, int k) {
        // 初始化所需的总小时数
        int ans = 0;
        // 遍历每棵桃树
        for (int x : p) {
            // 计算吃完这棵桃树上桃子所需的小时数，向上取整
            ans += Math.ceil(x * 1.0 / k);
        }
        // 如果所需总小时数小于等于h，则返回true，表示可以完成
        return ans <= h;
    }
}
```

javaScript

```
// 读取标准输入
const readline = require('readline');
const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});

// 判断以速度k是否能在h小时内吃完所有桃子
function canFinish(p, h, k) {
    let ans = 0;
    for (let x of p) {
        ans += Math.ceil(x / k);
    }
    return ans <= h;
}

// 处理输入
rl.on('line', (input) => {
    if (!this.peachCounts) {
        // 第一行输入，转换为桃子数量数组
        this.peachCounts = input.split(' ').map(Number);
        return;
    }
    // 第二行输入，转换为小时数
```

```
    const h = Number(input);
    rl.close(); // 不再读取输入

    // 输入验证
    const n = this.peachCounts.length;
    if (n === 0 || h <= 0 || n >= 10000 || h >= 10000 || n > h) {
        console.log(0);
        return;
    }

    // 二分查找最小吃桃速度
    let left = 1, right = 1e9;
    while (left < right) {
        const mid = Math.floor((left + right) / 2);
        if (canFinish(this.peachCounts, h, mid)) {
            right = mid;
        } else {
            left = mid + 1;
        }
    }

    // 输出最小吃桃速度
    console.log(left);
});
```

Python

```
import math

# 判断以速度k是否能在h小时内吃完所有桃子
def can_finish(p, h, k):
    ans = 0
    for x in p:
        ans += math.ceil(x / k)
    return ans <= h

# 读取输入
peach_counts = list(map(int, input().split()))
h = int(input())

```

```python
# 输入验证
n = len(peach_counts)
if n == 0 or h <= 0 or n >= 10000 or h >= 10000 or n > h:
    print(0)
    exit(0)

# 二分查找最小吃桃速度
left, right = 1, int(1e9)
while left < right:
    mid = (left + right) // 2
    if can_finish(peach_counts, h, mid):
        right = mid
    else:
        left = mid + 1

# 输出最小吃桃速度
print(left)
```

C语言

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

// 判断以速度k是否能在h小时内吃完所有桃子
int can_finish(int* p, int n, int h, int k) {
    int ans = 0;
    for (int i = 0; i < n; i++) {
        ans += (int)ceil((double)p[i] / k);
    }
    return ans <= h;
}

int main() {

    char input[10000];
    fgets(input, sizeof(input), stdin);

    // 将输入分割并存入数组
```

```
    int peach_counts[10000];
    int n = 0;

    char *token = strtok(input, " ");
    while (token != NULL) {
        peach_counts[n++] = atoi(token);
        token = strtok(NULL, " ");
    }


    int h;
    scanf("%d", &h);

    // 输入验证
    if (n == 0 || h <= 0 || n >= 10000 || h >= 10000 || n > h) {
        printf("0\n");
        return 0;
    }

    // 二分查找最小吃桃速度
    int left = 1, right = (int)1e9;
    while (left < right) {
        int mid = (left + right) / 2;
        if (can_finish(peach_counts, n, h, mid)) {
            right = mid;
        } else {
            left = mid + 1;
        }
    }

    // 输出最小吃桃速度
    printf("%d\n", left);

    return 0;
}
```

完整用例

用例1

```
2 3 4 5
4
```

用例2

```
2 3 4 5
3
```

用例3

```
30 11 23 4 20
6
```

用例4

```
1
1
```

用例5

```
1 1 1 1 1
5
```

用例6

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

用例7

```
100 200 300 400 500
5
```

用例8

```
1000 2000 3000 4000 5000
10
```

用例9

```
10000 20000 30000 40000 50000
20
```

用例10

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

文章目录

机考真题
华为OD