【华为OD机考 统一考试机试C卷】分月饼（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

中秋节，公司分月饼，m 个员工，买了 n 个月饼，m ≤ n，每个员工至少分 1 个月饼，但可以分多个，
单人分到最多月饼的个数是 Max1 ， 单人分到第二多月饼个数是 Max2 ， Max1 - Max2 ≤ 3 ，
单人分到第 n - 1 多月饼个数是 Max(n-1)，单人分到第n多月饼个数是 Max(n) ， Max(n-1) – Max(n) ≤ 3,
问有多少种分月饼的方法?

输入描述

每一行输入m n，表示m个员工，n个月饼，m ≤ n

输出描述

输出有多少种月饼分法

用例1

输入

```
1 | 2 4
```

输出

```
2
```

说明

> 分法有2种:
> 4 = 1 + 3
> 4 = 2 + 2
> 注意：1+3和3+1算一种分法

用例2

输入

```
3 5
```

输出

```
2
```

说明

> 5 = 1 + 1 + 3
> 5 = 1 + 2 + 2

用例3

输入

```
3 12
```

输出

```
6
```

说明

> 满足要求的有6种分法：
> 12 = 1 + 1 + 10（Max1 = 10, Max2 = 1，不满足Max1 - Max2 ≤ 3要求）
> 12 = 1 + 2 + 9（Max1 = 9, Max2 = 2，不满足Max1 - Max2 ≤ 3要求）
> 12 = 1 + 3 + 8（Max1 = 8, Max2 = 3，不满足Max1 - Max2 ≤ 3要求）
> 12 = 1 + 4 + 7（Max1 = 7, Max2 = 4，Max3 = 1，满足要求）
> 12 = 1 + 5 + 6（Max1 = 6, Max2 = 5，Max3 = 1，不满足要求）
> 12 = 2 + 2 + 8（Max1 = 8, Max2 = 2，不满足要求）
> 12 = 2 + 3 + 7（Max1 = 7, Max2 = 3，不满足要求）
> 12 = 2 + 4 + 6（Max1 = 6, Max2 = 4，Max3 = 2，满足要求）
> 12 = 2 + 5 + 5（Max1 = 5, Max2 = 2，满足要求）
> 12 = 3 + 3 + 6（Max1 = 6, Max2 = 3，满足要求）
> 12 = 3 + 4 + 5（Max1 = 5, Max2 = 4，Max3 = 3，满足要求）
> 12 = 4 + 4 + 4（Max1 = 4，满足要求）

解题思路

1. `distribute` 函数接收五个参数：员工数量 `m`，剩余月饼数量 `remaining`，当前允许的最大月饼数量 `maxAllowed`，当前处理的员工索引 `index`，以及一个记录每个员工分配月饼数量的向量 `distribution`。
2. **递归终止条件**：如果当前处理的是最后一个员工（`index == m - 1`），则检查是否可以将所有剩余的月饼分配给他，同时满足月饼数量差的限制（不超过3）。如果可以，返回1表示找到一种方法；如果不可以，返回0表示这条路径不可行。
3. **初始化方法数**：变量 `ways` 用于记录总的分配方法数，初始值为0。
4. **确定分配范围**：计算当前员工可以分配的月饼数量的起始值 `start` 和结束值 `end`。起始值是上一个员工分配的月饼数减3（或者1，如果是第一个员工），结束值是剩余月饼数和 `maxAllowed` 中的较小值减去为剩下的员工至少保留1个月饼的数量。
5. **遍历分配**：从 `start` 到 `end` 遍历所有可能分配给当前员工的月饼数量。对于每个可能的分配数量 `i`：
   - 将 `i` 个月饼分配给当前员工，并记录在 `distribution` 中。
   - 如果这个分配满足月饼数量差的限制（对于第一个员工没有前一个月饼数，所以不需要比较），则递归调用 `distribute` 函数处理下一个员工，剩余月饼数减去 `i`，并更新 `maxAllowed` 为 `i`（因为下一个员工分配的月饼数不能超过当前员工）。
   - 将递归调用返回的方法数加到 `ways` 上。
6. **返回结果**：函数返回总的分配方法数 `ways`。

这个算法通过递归遍历所有可能的分配组合，并在每一步检查是否满足条件，直到找到所有合法的分配方法。

C++

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

// 递归分配月饼的函数
int distribute(int m, int remaining, int maxAllowed, int index, vector<int>& distribution) {
    // 如果当前是最后一个员工
    if (index == m - 1) {
        // 检查剩余的月饼是否可以分配给最后一个员工
        // 并且满足最大和最小月饼数差值不超过3的条件
        if (remaining <= maxAllowed && (index == 0 || abs(distribution[index - 1] - remaining) <= 3)) {
            return 1; // 如果满足条件，返回1种方法
        }
        return 0; // 如果不满足条件，返回0种方法
    }

    int ways = 0; // 初始化分配方法数为0
    // 确定当前员工可以分配的月饼数的起始值
    int start = (index == 0) ? 1 : distribution[index - 1] - 3;
    // 确定当前员工可以分配的月饼数的结束值
    int end = min(remaining - (m - index - 1), maxAllowed);

    // 遍历所有可能的分配数量
    for (int i = start; i <= end; i++) {
        distribution[index] = i; // 尝试分配i个月饼给当前员工
        // 如果当前分配满足最大和最小月饼数差值不超过3的条件
        if (index == 0 || abs(distribution[index - 1] - i) <= 3) {
            // 递归调用distribute函数，计算剩余员工和月饼的分配方法
            ways += distribute(m, remaining - i, i, index + 1, distribution);
        }
    }

    return ways; // 返回总的分配方法数
}

```

```cpp
int main() {
    int m, n;
    cin >> m >> n; // 读取员工数量m和月饼数量n
    vector<int> distribution(m, 0); // 创建一个大小为m的向量用于存储分配情况
    // 调用distribute函数并输出返回的分配方法总数
    cout << distribute(m, n, n, 0, distribution) << endl;
    return 0;
}
```

Java

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int m = sc.nextInt(); // 读取员工数量m
        int n = sc.nextInt(); // 读取月饼数量n
        // 调用distribute方法并打印返回的分配方法总数
        System.out.println(distribute(m, n, n, 0, new int[m]));
        sc.close();
    }

    // 递归分配月饼的方法
    private static int distribute(int m, int remaining, int maxAllowed, int index, int[] distribution) {
        // 如果当前是最后一个员工
        if (index == m - 1) {
            // 检查剩余的月饼是否可以分配给最后一个员工
            // 并且满足最大和最小月饼数差值不超过3的条件
            if (remaining <= maxAllowed && (index == 0 || Math.abs(distribution[index - 1] - remaining) <= 3)) {
                return 1; // 如果满足条件，返回1种方法
            }
            return 0; // 如果不满足条件，返回0种方法
        }

        int ways = 0; // 初始化分配方法数为0
        // 确定当前员工可以分配的月饼数的起始值
        int start = (index == 0) ? 1 : distribution[index - 1] - 3;
        // 确定当前员工可以分配的月饼数的结束值
        int end = Math.min(remaining - (m - index - 1), maxAllowed);
```

```
        // 遍历所有可能的分配数量
        for (int i = start; i <= end; i++) {
            distribution[index] = i; // 尝试分配i个月饼给当前员工
            // 如果当前分配满足最大和最小月饼数差值不超过3的条件
            if (index == 0 || Math.abs(distribution[index - 1] - i) <= 3) {
                // 递归调用distribute方法，计算剩余员工和月饼的分配方法
                ways += distribute(m, remaining - i, i, index + 1, distribution);
            }
        }

        return ways; // 返回总的分配方法数
    }
}
```

javaScript

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

// 递归分配月饼的函数
function distribute(m, remaining, maxAllowed, index, distribution) {
  // 如果当前是最后一个员工
  if (index === m - 1) {
    // 检查剩余的月饼是否可以分配给最后一个员工
    // 并且满足最大和最小月饼数差值不超过3的条件
    if (remaining <= maxAllowed && (index === 0 || Math.abs(distribution[index - 1] - remaining) <= 3)) {
      return 1; // 如果满足条件，返回1种方法
    }
    return 0; // 如果不满足条件，返回0种方法
  }

  let ways = 0; // 初始化分配方法数为0
  // 确定当前员工可以分配的月饼数的起始值
  let start = (index === 0) ? 1 : distribution[index - 1] - 3;
  // 确定当前员工可以分配的月饼数的结束值
  let end = Math.min(remaining - (m - index - 1), maxAllowed);

```

```
  // 遍历所有可能的分配数量
  for (let i = start; i <= end; i++) {
    distribution[index] = i; // 尝试分配i个月饼给当前员工
    // 如果当前分配满足最大和最小月饼数差值不超过3的条件
    if (index === 0 || Math.abs(distribution[index - 1] - i) <= 3) {
      // 递归调用distribute函数，计算剩余员工和月饼的分配方法
      ways += distribute(m, remaining - i, i, index + 1, distribution);
    }
  }

  return ways; // 返回总的分配方法数
}

readline.on('line', (line) => {
    const [m, n] = line.split(' ').map(Number);
    // 调用distribute函数并输出返回的分配方法总数
    console.log(distribute(m, n, n, 0, Array(m).fill(0)));
    readline.close();

});
```

Python

```
# 递归分配月饼的函数
def distribute(m, remaining, max_allowed, index, distribution):
    # 如果当前是最后一个员工
    if index == m - 1:
        # 检查剩余的月饼是否可以分配给最后一个员工
        # 并且满足最大和最小月饼数差值不超过3的条件
        if remaining <= max_allowed and (index == 0 or abs(distribution[index - 1] - remaining) <= 3):
            return 1  # 如果满足条件，返回1种方法
        return 0  # 如果不满足条件，返回0种方法

    ways = 0  # 初始化分配方法数为0
    # 确定当前员工可以分配的月饼数的起始值
    start = 1 if index == 0 else distribution[index - 1] - 3
    # 确定当前员工可以分配的月饼数的结束值
    end = min(remaining - (m - index - 1), max_allowed)

    # 遍历所有可能的分配数量
```

```
    for i in range(start, end + 1):
        distribution[index] = i  # 尝试分配i个月饼给当前员工
        # 如果当前分配满足最大和最小月饼数差值不超过3的条件
        if index == 0 or abs(distribution[index - 1] - i) <= 3:
            # 递归调用distribute函数，计算剩余员工和月饼的分配方法
            ways += distribute(m, remaining - i, i, index + 1, distribution)

    return ways  # 返回总的分配方法数

# 主函数，程序的入口点
if __name__ == '__main__':
    m, n = map(int, input().split())
    # 调用distribute函数并打印返回的分配方法总数
    print(distribute(m, n, n, 0, [0] * m))
```

文章目录


机考真题
华为OD
No No No
CSDN @算法大师