【华为OD机考 统一考试机试C卷】结队编程（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

某部门计划通过结队编程来进行项目开发，已知该部门有 N 名员工，每个员工有独一无二的职级，每三个员工形成一个小组进行结队编程，结队分组规则如下：

从部门中选出序号分别为 i、j、k 的3名员工，他们的职级分贝为 level[i]，level[j]，level[k]，

结队小组满足 level[i] < level[j] < level[k] 或者 level[i] > level[j] > level[k]，

其中 0 ≤ i < j < k < n。

请你按上述条件计算可能组合的小组数量。同一员工可以参加多个小组。

输入描述

第一行输入：员工总数 n

第二行输入：按序号依次排列的员工的职级 level，中间用空格隔开

备注：

- 1 ≤ n ≤ 6000
- 1 ≤ level[i] ≤ 10^5

输出描述

可能结队的小组数量

用例

输入	4 1 2 3 4
输出	4
说明	可能结队成的组合(1,2,3)、(1,2,4)、(1,3,4)、(2,3,4)

输入	3 5 4 7
输出	0
说明	根据结队条件，我们无法为该部门组建小组

解题思路

主要思路：统计每个员工左侧和右侧的职级比较情况，然后利用这些信息来计算每个员工能够作为中间员工参与的队伍数量。

1. 初始化四个数组，分别用于存储每个员工左侧和右侧职级小于和大于他的员工数量。
2. 对于每个员工，遍历其左侧的所有员工，统计左侧职级小于和大于当前员工的数量。
3. 对于每个员工，遍历其右侧的所有员工，统计右侧职级小于和大于当前员工的数量。
4. 对于每个员工，计算其能够作为中间员工参与的队伍数量。这包括：
    - 左侧职级小于他的员工数量乘以右侧职级大于他的员工数量。
    - 左侧职级大于他的员工数量乘以右侧职级小于他的员工数量。
5. 将每个员工作为中间员工能参与的队伍数量累加起来，得到总的队伍数量。

具体来说：

- 假设中间员工的职级大于左侧员工且小于右侧员工，那么：

- 如果左侧有 `L` 个员工的职级小于中间员工，
- 并且右侧有 `R` 个员工的职级大于中间员工，
- 那么就存在 `L * R` 种不同的组合方式，每种组合都对应一个有效的队伍。

- 同理，如果中间员工的职级小于左侧员工且大于右侧员工，那么：
  - 如果左侧有 `L'` 个员工的职级大于中间员工，
  - 并且右侧有 `R'` 个员工的职级小于中间员工，
  - 那么也存在 `L' * R'` 种不同的组合方式，每种组合都对应一个有效的队伍。

这是因为组合的本质是从左侧选择一个员工和从右侧选择一个员工，而这两个选择是独立的。所以，左侧每增加一个可能的员工，就会与右侧的每个可能员工形成新的组合，因此是乘法关系。这就是为什么我们用乘积来计算每个员工作为中间员工时可能形成的队伍总数。

C++

```cpp
#include <iostream>
#include <vector>
#include <string>
#include <sstream>
using namespace std;
// 将字符串分割为字符串向量
vector<string> split(const string &s, char delimiter) {
    vector<string> tokens;
    string token;
    istringstream tokenStream(s);
    while (getline(tokenStream, token, delimiter)) {
        tokens.push_back(token);
    }
    return tokens;
}

// 计算可以结成的队伍数量
long countTeams(int n, const vector<int> &levels) {
    long ans = 0; // 初始化答案变量，用于存储可能的队伍数量
    vector<int> smallerToLeft(n, 0); // 创建向量，存储每个员工左侧比他职级小的员工数量
    vector<int> greaterToLeft(n, 0); // 创建向量，存储每个员工左侧比他职级大的员工数量
    vector<int> smallerToRight(n, 0); // 创建向量，存储每个员工右侧比他职级小的员工数量
    vector<int> greaterToRight(n, 0); // 创建向量，存储每个员工右侧比他职级大的员工数量
```

```
    // 预计算每个员工左侧的职级比较情况
    for (int i = 1; i < n; i++) {
        for (int j = 0; j < i; j++) {
            if (levels[j] < levels[i]) {
                smallerToLeft[i]++;
            } else if (levels[j] > levels[i]) {
                greaterToLeft[i]++;
            }
        }
    }

    // 预计算每个员工右侧的职级比较情况
    for (int i = n - 2; i >= 0; i--) {
        for (int j = n - 1; j > i; j--) {
            if (levels[j] < levels[i]) {
                smallerToRight[i]++;
            } else if (levels[j] > levels[i]) {
                greaterToRight[i]++;
            }
        }
    }

    // 计算可能的队伍数量
    for (int i = 0; i < n; i++) {
        ans += (long) smallerToLeft[i] * greaterToRight[i] + (long) greaterToLeft[i] * smallerToRight[i];
    }

    return ans;
}

int main() {
    int n; // 员工总数
    cin >> n;
    cin.ignore(); // 忽略换行符
    string line;
    getline(cin, line); // 读取一行员工职级
    vector<string> inputLevels = split(line, ' '); // 分割字符串获取每个员工职级
    vector<int> levels(n); // 每个员工的职级
    for (int i = 0; i < n; i++) {
```

```
        levels[i] = stoi(inputLevels[i]); // 将字符串转换为整数
    }

    cout << countTeams(n, levels) << endl; // 输出可以结成的队伍数量
    return 0;
}
```

Java

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int n = Integer.parseInt(sc.nextLine()); //  员工总数
        int[] levels = new int[n]; //  每个员工的职级
        String[] inputLevels = sc.nextLine().split(" "); //  每个员工职级的字符串数组
        for (int i = 0; i < n; i++) { // 遍历字符串数组
            levels[i] = Integer.parseInt(inputLevels[i]); // 将每个员工的职级字符串转换为整数，并存储到levels数组中
        }

        System.out.println(countTeams(n, levels)); // 调用countTeams方法计算并输出可以结成的队伍数量
    }

    public static long countTeams(int n, int[] levels) {
        long ans = 0; // 初始化答案变量，用于存储可能的队伍数量
        int[] smallerToLeft = new int[n]; // 创建数组，存储每个员工左侧比他职级小的员工数量
        int[] greaterToLeft = new int[n]; // 创建数组，存储每个员工左侧比他职级大的员工数量
        int[] smallerToRight = new int[n]; // 创建数组，存储每个员工右侧比他职级小的员工数量
        int[] greaterToRight = new int[n]; // 创建数组，存储每个员工右侧比他职级大的员工数量

        // 预计算每个员工左侧的职级比较情况
        for (int i = 1; i < n; i++) { // 从第二个员工开始遍历
            for (int j = 0; j < i; j++) { // 遍历当前员工左侧的所有员工
                if (levels[j] < levels[i]) { // 如果左侧员工职级小于当前员工
                    smallerToLeft[i]++; // 对应的数量加一
                } else if (levels[j] > levels[i]) { // 如果左侧员工职级大于当前员工
                    greaterToLeft[i]++; // 对应的数量加一
                }
```

```
            }
        }

        // 预计算每个员工右侧的职级比较情况
        for (int i = n - 2; i >= 0; i--) { // 从倒数第二个员工开始向前遍历
            for (int j = n - 1; j > i; j--) { // 遍历当前员工右侧的所有员工
                if (levels[j] < levels[i]) { // 如果右侧员工职级小于当前员工
                    smallerToRight[i]++; // 对应的数量加一
                } else if (levels[j] > levels[i]) { // 如果右侧员工职级大于当前员工
                    greaterToRight[i]++; // 对应的数量加一
                }
            }
        }

        // 计算可能的队伍数量
        for (int i = 0; i < n; i++) { // 遍历每个员工
            // 将当前员工左侧小于他的员工数量与右侧大于他的员工数量相乘，并加到答案中
            // 将当前员工左侧大于他的员工数量与右侧小于他的员工数量相乘，并加到答案中
            ans += (long) smallerToLeft[i] * greaterToRight[i] + (long) greaterToLeft[i] * smallerToRight[i];
        }

        return ans; // 返回计算出的队伍数量
    }
}
```

javaScript

```
// Node.js 版本

const readline = require('readline');

// 创建 readline 接口实例
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

// 读取员工总数和员工职级
rl.on('line', (n) => {
  rl.on('line', (levelsStr) => {
```

```
    const levels = levelsStr.split(' ').map(Number); // 将员工职级字符串转换为数字数组
    console.log(countTeams(parseInt(n, 10), levels)); // 计算并输出可以结成的队伍数量
    rl.close();
  });
});

// 计算可以结成的队伍数量的函数
function countTeams(n, levels) {
  let ans = 0; // 初始化答案变量，用于存储可能的队伍数量
  let smallerToLeft = new Array(n).fill(0); // 创建数组，存储每个员工左侧比他职级小的员工数量
  let greaterToLeft = new Array(n).fill(0); // 创建数组，存储每个员工左侧比他职级大的员工数量
  let smallerToRight = new Array(n).fill(0); // 创建数组，存储每个员工右侧比他职级小的员工数量
  let greaterToRight = new Array(n).fill(0); // 创建数组，存储每个员工右侧比他职级大的员工数量

  // 预计算每个员工左侧的职级比较情况
  for (let i = 1; i < n; i++) {
    for (let j = 0; j < i; j++) {
      if (levels[j] < levels[i]) {
        smallerToLeft[i]++;
      } else if (levels[j] > levels[i]) {
        greaterToLeft[i]++;
      }
    }
  }

  // 预计算每个员工右侧的职级比较情况
  for (let i = n - 2; i >= 0; i--) {
    for (let j = n - 1; j > i; j--) {
      if (levels[j] < levels[i]) {
        smallerToRight[i]++;
      } else if (levels[j] > levels[i]) {
        greaterToRight[i]++;
      }
    }
  }

  // 计算可能的队伍数量
  for (let i = 0; i < n; i++) {
    ans += smallerToLeft[i] * greaterToRight[i] + greaterToLeft[i] * smallerToRight[i];
  }
```

```
    return ans;
}
```

Python

```python
# Python 版本

# 计算可以结成的队伍数量的函数
def count_teams(n, levels):
    ans = 0  # 初始化答案变量，用于存储可能的队伍数量
    smaller_to_left = [0] * n  # 创建列表，存储每个员工左侧比他职级小的员工数量
    greater_to_left = [0] * n  # 创建列表，存储每个员工左侧比他职级大的员工数量
    smaller_to_right = [0] * n  # 创建列表，存储每个员工右侧比他职级小的员工数量
    greater_to_right = [0] * n  # 创建列表，存储每个员工右侧比他职级大的员工数量

    # 预计算每个员工左侧的职级比较情况
    for i in range(1, n):
        for j in range(i):
            if levels[j] < levels[i]:
                smaller_to_left[i] += 1
            elif levels[j] > levels[i]:
                greater_to_left[i] += 1

    # 预计算每个员工右侧的职级比较情况
    for i in range(n - 2, -1, -1):
        for j in range(n - 1, i, -1):
            if levels[j] < levels[i]:
                smaller_to_right[i] += 1
            elif levels[j] > levels[i]:
                greater_to_right[i] += 1

    # 计算可能的队伍数量
    for i in range(n):
        ans += smaller_to_left[i] * greater_to_right[i] + greater_to_left[i] * smaller_to_right[i]

    return ans

# 读取输入
n = int(input())  # 员工总数
```

```
levels = list(map(int, input().split()))  # 每个员工的职级

# 输出可以结成的队伍数量
print(count_teams(n, levels))
```

文章目录