【华为OD机考 统一考试机试C卷】 查找众数及中位数（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：查找众数及中位数（本题分值100）

众数是指一组数据中出现次数量多的那个数，众数可以是多个。
中位数是指把一组数据从小到大排列，最中间的那个数，如果这组数据的个数是奇数，那最中间那个就是中位数，如果这组数据的个数为偶数，那就把中间的两个数之和除以2，所得的结果就是中位数。
查找整型数组中元素的众数并组成一个新的数组，求新数组的中位数。

输入描述

输入一个一维整型数组，数组大小取值范围 0<N<1000，数组中每个元素取值范围 0<E<1000

输出描述

输出众数组成的新数组的中位数

用例

输入	
输出	

输入	
输出	

输入	
输出	

题目解析

题目要求找到整型数组中的众数，可以使用哈希表来统计每个数出现的次数，然后找到出现次数最多的数，即为众数。如果有多个众数，都放入一个新数组中。接着，对新数组进行排序，找到其中位数即可。

具体步骤如下：

1. 使用哈希表统计每个数出现的次数，找到出现次数最多的数，即为众数。如果有多个众数，都放入一个新数组中。
2. 对新数组进行排序，找到其中位数。

C++

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

using namespace std;

int main() {
    // 处理输入
    string input;
    getline(cin, input);
    vector<int> numbers;
    size_t pos = 0;
    while ((pos = input.find(' ')) != string::npos) {
        int num = stoi(input.substr(0, pos));
        numbers.push_back(num);
        input.erase(0, pos + 1);
    }
    numbers.push_back(stoi(input));

    // 统计数字出现次数及出现最大次数
    map<int, int> countMap;
    for (int number : numbers) {
        if (countMap.count(number)) {
            countMap[number]++;
        } else {
            countMap[number] = 1;
        }
    }
    int maxCount = 0;
    for (auto count : countMap) {
        if (count.second >= maxCount) {
            maxCount = count.second;
        }
    }

    // 获取出现最大次数的数字并排序
    vector<int> maxCountNumbers;
    for (auto entry : countMap) {
        if (entry.second == maxCount) {
            maxCountNumbers.push_back(entry.first);
        }
    }
    sort(maxCountNumbers.begin(), maxCountNumbers.end());

    // 计算中位数
    if (maxCountNumbers.size() % 2 != 0) {
        int index = (maxCountNumbers.size() + 1) / 2 - 1;
        cout << maxCountNumbers[index] << endl;
    } else {
        int index1 = maxCountNumbers.size() / 2 - 1;
        int index2 = maxCountNumbers.size() / 2;
        cout << (maxCountNumbers[index1] + maxCountNumbers[index2]) / 2 << endl;
    }

    return 0;
}

```

JavaScript

```javascript
const readline = require('readline');

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

rl.on('line', (input) => {
  const numbers = input.split(" ").map(num => parseInt(num));

  const countMap = new Map();
  let maxCount = 0;
  for (let number of numbers) {
    let count = countMap.get(number) || 0;
    count++;
    countMap.set(number, count);
    maxCount = Math.max(maxCount, count);
  }

  const maxCountNumbers = Array.from(countMap.entries())
    .filter(entry => entry[1] === maxCount)
    .map(entry => entry[0])
    .sort((a, b) =>b - a);

  let median;
  if (maxCountNumbers.length % 2 !== 0) {
    let index = Math.floor((maxCountNumbers.length + 1) / 2) - 1;
    median = maxCountNumbers[index];
  } else {
    let index1 = maxCountNumbers.length / 2 - 1;
    let index2 = maxCountNumbers.length / 2;
    median = Math.floor((maxCountNumbers[index1] + maxCountNumbers[index2]) / 2);
  }

  console.log(median);

  rl.close();
});
```

Java

```java
import java.util.*;
import java.util.stream.Collectors;

public class Main {
    public static void main(String[] args) {
        // 处理输入
        Scanner scanner = new Scanner(System.in);
        String input = scanner.nextLine();
        List<Integer> numbers = new ArrayList<>();
        for (String num : input.split(" ")) {
            numbers.add(Integer.parseInt(num));
        }

        // 统计数字出现次数及出现最大次数
        Map<Integer, Integer> countMap = new HashMap<>();
        int maxCount = 0;
        for (int number : numbers) {
            int count = countMap.getOrDefault(number, 0) + 1;
            countMap.put(number, count);
```

```
            maxCount = Math.max(maxCount, count);
        }

        final int finalMaxCount = maxCount;
        List<Integer> maxCountNumbers = countMap.entrySet()
                .stream()
                .filter(entry -> entry.getValue() == finalMaxCount)
                .map(Map.Entry::getKey)
                .sorted()
                .collect(Collectors.toList());

        // 计算中位数
        int median;
        if (maxCountNumbers.size() % 2 != 0) {
            int index = (maxCountNumbers.size() + 1) / 2 - 1;
            median = maxCountNumbers.get(index);
        } else {
            int index1 = maxCountNumbers.size() / 2 - 1;
            int index2 = maxCountNumbers.size() / 2;
            median = (maxCountNumbers.get(index1) + maxCountNumbers.get(index2)) / 2;
        }

        System.out.println(median);
    }
}
```

python

```
# 处理输入
input_str = input()
numbers = list(map(int, input_str.split()))

# 统计数字出现次数及出现最大次数
count_map = {}
for number in numbers:
    if number in count_map:
        count_map[number] += 1
    else:
        count_map[number] = 1
max_count = max(count_map.values())

# 获取出现最大次数的数字并排序
max_count_numbers = [number for number, count in count_map.items() if count == max_count]
max_count_numbers.sort()

# 计算中位数
if len(max_count_numbers) % 2 != 0:
    index = (len(max_count_numbers) + 1) // 2 - 1
    print(max_count_numbers[index])
else:
    index1 = len(max_count_numbers) // 2 - 1
    index2 = len(max_count_numbers) // 2
    print((max_count_numbers[index1] + max_count_numbers[index2]) // 2)
```

C语言

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

```

```c
#define MAX_SIZE 1000 // 定义最大数组和字符串大小

int compare(const void *a, const void *b) {
    // qsort排序函数的比较函数
    return (*(int *)a - *(int *)b);
}

int main() {
    char input[MAX_SIZE]; // 存储输入的字符串
    int numbers[MAX_SIZE]; // 存储解析后的数字
    int count[MAX_SIZE] = {0}; // 计数数组，存储每个数字出现的次数
    int modeArray[MAX_SIZE]; // 存储众数的数组
    int modeSize = 0; // 众数数组的大小
    int n = 0; // 数字的数量

    // 读取一行输入
    fgets(input, MAX_SIZE, stdin);
    input[strcspn(input, "\n")] = 0; // 移除换行符

    // 解析字符串并填充数字数组
    char *token = strtok(input, " ");
    while (token != NULL) {
        numbers[n++] = atoi(token);
        token = strtok(NULL, " ");
    }

    // 统计数字出现次数及出现最大次数
    int maxCount = 0;
    for (int i = 0; i < n; i++) {
        count[numbers[i]]++;
        if (count[numbers[i]] > maxCount) {
            maxCount = count[numbers[i]];
        }
    }

    // 找出所有众数
    for (int i = 0; i < MAX_SIZE; i++) {
        if (count[i] == maxCount) {
            modeArray[modeSize++] = i;
        }
    }

    // 对众数数组排序
    qsort(modeArray, modeSize, sizeof(int), compare);

    // 计算中位数
    int median;
    if (modeSize % 2 != 0) {
        median = modeArray[modeSize / 2];
    } else {
        median = (modeArray[modeSize / 2 - 1] + modeArray[modeSize / 2]) / 2;
    }

    // 输出中位数
    printf("%d\n", median);

    return 0;
}
```

文章目录

完整用例

用例1

```
1 2 3 4 5
```

用例2

```
5 5 5 5 5 5
```

用例3

```
10 20 30 40 50 60 70 80 90 100
```

用例4

```
7 7 7 7 7 7 7 7 7 7
```

用例5

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

用例6

```
1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 10 10 11 11 12 12 13 13 14 14 15 15 16 16 17 17 18 18 19 19 20 20 21 21 22 22 23 23 24 24 25 25 26 26 27 27 28 28 29 29 30 30 31 31 32 32 33 33 34 34 35 35 36 36 37 37 38 38 39 39 40 40 41 41 42 42 43 43 44 4
```

用例7

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 8
```

用例8

```
10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250 260 270 280 290 300 310 320 330 340 350 360 370 380 390 400 410 420 430 440 450 460 470 480 490 500 510 520 530 540 550 560 570 580 590 600 610 620 630 6
```

用例9

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 8
```

用例10

```
5 1 5 3 5 2 5 5 7 6 7 3 7 11 7 55 7 9 98 9 17 9 15 9 9 1 3
```