【华为OD机考 统一考试机试C卷】数组去重和排序（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

给定一个乱序的数组，删除所有的重复元素，使得每个元素只出现一次，并且按照出现的次数从高到低进行排序，相同出现次数按照第一次出现顺序进行先后排序。

输入描述

一个数组

输出描述

去重排序后的数组

用例

输入

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

输出

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

备注 数组大小不超过100 数组元素值大小不超过100。

题目解析

先统计数组每个元素重复的次数，然后去重，最后根据重复次数来排序去重元素。

C++

```
#include <iostream>
#include <vector>
#include <unordered_map>
#include <algorithm>
using namespace std;
int main() {
    string input;
    getline(cin, input);
    vector<string> inputArray;
    size_t pos = 0;
    while ((pos = input.find(",")) != string::npos) {
        string element = input.substr(0, pos);
        inputArray.push_back(element);
        input.erase(0, pos + 1);
    }
    inputArray.push_back(input);

    unordered_map<string, int> countMap;
    unordered_map<string, int> firstMap;
    for (int i = 0; i < inputArray.size(); i++) {
        string element = inputArray[i];
        countMap[element] = countMap[element] + 1;
        if (firstMap.find(element) == firstMap.end()) {
            firstMap[element] = i;
        }
    }


```

```
    vector<string> sortedKeys;
    for (const auto& pair : firstMap) {
        sortedKeys.push_back(pair.first);
    }
    sort(sortedKeys.begin(), sortedKeys.end(), [&](const string& a, const string& b) {
        int countA = countMap[a];
        int countB = countMap[b];
        if (countA != countB) {
            return countB < countA;
        } else {
            int firstA = firstMap[a];
            int firstB = firstMap[b];
            return firstA < firstB;
        }
    });

    string result;
    for (const auto& key : sortedKeys) {
        result += key + ",";
    }
    result.pop_back();

    cout << result << endl;

    return 0;
}
```

JavaScript

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

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

rl.on('line', (input) => {
    const inputArray = input.split(",");

    // 使用两个Map来记录元素出现的次数和第一次出现的位置
```

```
    const countMap = new Map(); // 记录元素出现的次数
    const firstMap = new Map(); // 记录元素第一次出现的位置
    for (let i = 0; i < inputArray.length; i++) {
        const element = inputArray[i];
        countMap.set(element, countMap.get(element) + 1 || 1); // 将元素的出现次数加1，如果元素不存在则默认为1
        if (!firstMap.has(element)) {
            firstMap.set(element, i); // 如果元素不存在，则将其位置加入到firstMap中
        }
    }

    // 根据元素出现的次数和第一次出现的位置进行排序
    const sortedKeys = Array.from(firstMap.keys()); // 获取所有元素的列表
    sortedKeys.sort((a, b) => {
        const countA = countMap.get(a); // 获取元素a的出现次数
        const countB = countMap.get(b); // 获取元素b的出现次数
        if (countA !== countB) {
            return countB - countA; // 如果两个元素的出现次数不相等，按照出现次数降序排序
        } else {
            const firstA = firstMap.get(a); // 获取元素a的第一次出现的位置
            const firstB = firstMap.get(b); // 获取元素b的第一次出现的位置
            return firstA - firstB; // 如果两个元素的出现次数相等，按照第一次出现的位置升序排序
        }
    });

    // 构建结果字符串
    let result = '';
    for (let i = 0; i < sortedKeys.length; i++) {
        result += sortedKeys[i];
        if (i !== sortedKeys.length - 1) {
            result += ',';
        }
    }

    console.log(result); // 输出结果字符串

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

Java

```java
import java.util.*;

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

        // 使用两个Map来记录元素出现的次数和第一次出现的位置
        Map<String, Integer> countMap = new HashMap<>(); // 记录元素出现的次数
        Map<String, Integer> firstMap = new HashMap<>(); // 记录元素第一次出现的位置
        for (int i = 0; i < inputArray.length; i++) {
            String element = inputArray[i];
            countMap.put(element, countMap.getOrDefault(element, 0) + 1); // 将元素的出现次数加1，如果元素不存在则默认为0
            firstMap.putIfAbsent(element, i); // 如果元素不存在，则将其位置加入到firstMap中
        }

        // 根据元素出现的次数和第一次出现的位置进行排序
        List<String> sortedKeys = new ArrayList<>(firstMap.keySet()); // 获取所有元素的列表
        Collections.sort(sortedKeys, (a, b) -> {
            int countA = countMap.get(a); // 获取元素a的出现次数
            int countB = countMap.get(b); // 获取元素b的出现次数
            if (countA != countB) {
                return countB - countA; // 如果两个元素的出现次数不相等，按照出现次数降序排序
            } else {
                int firstA = firstMap.get(a); // 获取元素a的第一次出现的位置
                int firstB = firstMap.get(b); // 获取元素b的第一次出现的位置
                return firstA - firstB; // 如果两个元素的出现次数相等，按照第一次出现的位置升序排序
            }
        });

        // 构建结果字符串
        StringBuilder sb = new StringBuilder();
        for (String key : sortedKeys) {
            sb.append(key).append(","); // 将排序后的元素逐个添加到结果字符串中
        }
        sb.deleteCharAt(sb.length() - 1); // 删除最后一个逗号

        System.out.println(sb.toString()); // 输出结果字符串
    }
}
```

Python

```python
import collections

inputArray = input().split(",")

# 使用两个字典来记录元素出现的次数和第一次出现的位置
countDict = collections.defaultdict(int) # 记录元素出现的次数
firstDict = {} # 记录元素第一次出现的位置
for i in range(len(inputArray)):
    element = inputArray[i]
    countDict[element] += 1 # 将元素的出现次数加1
    if element not in firstDict: # 如果元素不存在，则将其位置加入到firstDict中
        firstDict[element] = i

# 根据元素出现的次数和第一次出现的位置进行排序
sortedKeys = sorted(firstDict.keys(), key=lambda x: (-countDict[x], firstDict[x]))

# 构建结果字符串
result = ",".join(sortedKeys)

print(result) # 输出结果字符串
```

C语言

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

#define MAX_LEN 100

// 定义一个结构体，用于存储元素和对应的次数以及第一次出现的位置
typedef struct {
    int element;
    int count;
    int first;
} ElementCount;

// 定义一个比较函数，用于 qsort 函数
int cmp(const void *a, const void *b) {
```

```
    ElementCount *ec1 = (ElementCount *)a;
    ElementCount *ec2 = (ElementCount *)b;
    if (ec1->count != ec2->count)
        return ec2->count - ec1->count;
    return ec1->first - ec2->first;
}

int main() {
    // 输入的字符串
    char input[MAX_LEN * 4];
    fgets(input, MAX_LEN * 4, stdin);
    input[strcspn(input, "\n")] = '\0';  // 去掉末尾的换行符

    // 分割字符串并转换为整数数组
    int nums[MAX_LEN];
    int nums_len = 0;
    char *token = strtok(input, ",");
    while (token != NULL) {
        nums[nums_len++] = atoi(token);
        token = strtok(NULL, ",");
    }

    // 统计每个元素出现的次数以及第一次出现的位置
    ElementCount countMap[MAX_LEN];
    int countMap_len = 0;
    int firstMap[MAX_LEN] = {0};
    for (int i = 0; i < nums_len; ++i) {
        int element = nums[i];
        int j;
        for (j = 0; j < countMap_len; ++j) {
            if (countMap[j].element == element) {
                countMap[j].count++;
                break;
            }
        }
        if (j == countMap_len) {
            countMap[countMap_len].element = element;
            countMap[countMap_len].count = 1;
            countMap[countMap_len].first = i;
            countMap_len++;

```

```
        }
    }

    // 排序
    qsort(countMap, countMap_len, sizeof(ElementCount), cmp);

    // 输出结果
    for (int i = 0; i < countMap_len; ++i) { // 遍历 countMap
        printf("%d", countMap[i].element); // 输出每个元素
        if (i != countMap_len - 1) {
            printf(",");
        }
    }
    printf("\n"); // 输出换行符

    return 0;
}
```

完整用例

用例1

1,2,3,4,5

用例2

1,1,1,1,1

用例3

2,2,2,3,3,3

用例4

5,4,3,2,1

用例5

4,4,2,2,2,1,1

用例6

1,3,3,3,2,4,4,4,5

用例7

5,4,3,2,1,1,2,3,4,5

用例8

2,2,3,3,1,1,4,4,5,5

用例9

3,1,2,1,2,3,3,2,1

用例10

1,1,2,2,3,3,4,4,5,5

文章目录

用例5
用例6
用例7
用例8
用例9
用例10