【排序】-智能成绩表

题目描述与示例

题目描述

小明来到某学校当老师，需要将学生按考试总分或单科分数进行排名，你能帮帮他吗？

输入描述

第 `1` 行输入两个整数，学生人数 `n` 和科目数量 `m` 。 `0 < n < 100` ， `0 < m < 10`

第 `2` 行输入 `m` 个科目名称，彼此之间用空格隔开，科目名称只由英文字母构成，单个长度不超过 `10` 个字符。科目的出现顺字和后续输入的学生成绩一一对应。不会出现重复的科目名称。

第 `3` 行开始的行，每行包含一个学生的姓名和该生 `m` 个科目的成绩空格隔开，学生不会重名。

学生姓名只由英文字母构成，长度不超过 `10` 个字符。成绩是 `0~100` 的整数，依次对应第 `2` 行中输入的科目。

第 `n+2` 行，输入用作排名的科目名称。

若科目不存在，则按总分进行排序。

输出描述

输出一行，按成绩排序后的学生名字，空格隔开。成绩相同的按照学生姓名字典顺序排序。

示例一

输入

```
3 2
yuwen shuxue
fangfang 95 90
xiaohua 88 95
minmin 100 82
```

```
6 shuxue
```

输出

```
1 xiaohua fangfang minmin
```

示例二

输入

```
1 3 3
2 yuwen shuxue yingyu
3 fangfang 95 90 100
4 xiaohua 88 95 20
5 minmin 100 82 60
```

输出

```
1 fangfang minmin xiaohua
```

解题思路

很简单、直接的排序模拟题。

要注意，题目描述的最后一句**“若科目不存在，则按总分进行排序”**存在一定的歧义。有以下两种理解：

1. 第 `n+2` 输入一定存在，但输入的科目可能在第 `2` 行输入的长度为 `m` 的科目数组中不存在
2. 第 `n+2` 输入不一定存在，最后一行输入不是单个科目。

这两种情况都是合理的，具体属于哪种情况，在考试的时候应该根据用例来反推。

本题中的示例二是我造出来的，沿用了上述讨论中的**第二种情况**。故后续的代码、OJ系统上的示例都将基于这种情况进行讨论。

本题的核心考点是哈希表和排序API的使用，核心的代码都是类似的。所以要学会变通，无论遇到哪种用例情况，都能够顺利解决问题。

代码

Python

```python
# 题目：【排序】2024E-智能成绩表
# 分值：100
# 作者：闭着眼睛学数理化
# 算法：直接调用排序API
# 代码看不懂的地方，请直接在群上提问


# 输入学生人数n，科目个数m
n, m = map(int, input().split())
# 输入m个科目名字
subjects = input().split()

# 构建哈希表，key为学生姓名，value为学生的各科分数
dic = dict()

# 循环n次，输入每一个学生的信息
for _ in range(n):
    lst = input().split()
    # 数组首位为学生姓名
    name = lst[0]
    # 数组剩下的m个元素为学生各科成绩
    grades = list(int(x) for x in lst[1:])
    # 将姓名和成绩储存在dic中
    dic[name] = grades

# 获得dic中的所有键，为所有学生姓名
ans = list(dic.keys())

# 使用异常处理语句，来解决最后一行的输入问题
```

```python
try:
    # 输入排序依据的键（科目名）
    k = input()
    # 获得该科目名在数组subjects中的下标
    idx = subjects.index(k)
    # 对ans进行排序
    # 优先按照第idx科成绩降序排序，相同时再按照姓名字典序排序
    ans.sort(key = lambda name: (-dic[name][idx], name))
except:
    # 优先按照总分成绩降序排序，相同时再按照姓名字典序排序
    ans.sort(key = lambda name: (-sum(dic[name]), name))

# 输出结果
print(" ".join(name for name in ans))
```

Java

```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int m = scanner.nextInt();
        scanner.nextLine(); // Consume newline

        String[] subjects = scanner.nextLine().split(" ");

        Map<String, List<Integer>> dic = new HashMap<>();

        for (int i = 0; i < n; i++) {
            String[] lst = scanner.nextLine().split(" ");
            String name = lst[0];
            List<Integer> grades = new ArrayList<>();
            for (int j = 1; j <= m; j++) {
                grades.add(Integer.parseInt(lst[j]));
            }
            dic.put(name, grades);
        }

        try {
            String k = scanner.nextLine();
            int idx = Arrays.asList(subjects).indexOf(k);
            List<String> ans = new ArrayList<>(dic.keySet());
            Collections.sort(ans, (name1, name2) -> {
```

```
                int val = dic.get(name2).get(idx) - dic.get(name1).get(idx);
                return val == 0 ? name1.compareTo(name2) : val;
            });
            System.out.println(String.join(" ", ans));
        } catch (Exception e) {
            List<String> ans = new ArrayList<>(dic.keySet());
            Collections.sort(ans, (name1, name2) -> {
                int val1 =
dic.get(name1).stream().mapToInt(Integer::intValue).sum();
                int val2 =
dic.get(name2).stream().mapToInt(Integer::intValue).sum();
                return val2 == val1 ? name1.compareTo(name2) :
Integer.compare(val2, val1);
            });
            System.out.println(String.join(" ", ans));
        }
    }
}

```

C++

```cpp
#include <iostream>
#include <vector>
#include <unordered_map>
#include <algorithm>
#include <numeric>

int main() {
    int n, m;
    std::cin >> n >> m;
    std::cin.ignore(); // Ignore newline

    std::string subjects[m];
    for (int i = 0; i < m; ++i) {
        std::cin >> subjects[i];
    }

    std::unordered_map<std::string, std::vector<int>> dic;
    for (int i = 0; i < n; ++i) {
        std::string name;
        std::cin >> name;
        std::vector<int> grades(m);
        for (int j = 0; j < m; ++j) {
            std::cin >> grades[j];
```

```
        }
        dic[name] = grades;
    }

    bool exceptionCaught = false;
    std::string k;
    std::cin >> k;

    int idx = -1;
    for (int i = 0; i < m; ++i) {
        if (subjects[i] == k) {
            idx = i;
            break;
        }
    }

    if (idx != -1) {
        std::vector<std::string> ans;
        for (const auto& entry : dic) {
            ans.push_back(entry.first);
        }

        std::sort(ans.begin(), ans.end(), [&](const std::string& name1, const std::string& name2) {
            int val = dic[name2][idx] - dic[name1][idx];
            return (val == 0) ? (name1 < name2) : (val < 0);
        });

        for (const std::string& name : ans) {
            std::cout << name << " ";
        }
    } else {
        exceptionCaught = true;
    }

    if (exceptionCaught) {
        std::vector<std::string> ans;
        for (const auto& entry : dic) {
            ans.push_back(entry.first);
        }

        std::sort(ans.begin(), ans.end(), [&](const std::string& name1, const std::string& name2) {
            int val1 = std::accumulate(dic[name1].begin(), dic[name1].end(), 0);
            int val2 = std::accumulate(dic[name2].begin(), dic[name2].end(), 0);
```

```
            return (val1 == val2) ? (name1 < name2) : (val2 < val1);
        });

        for (const std::string& name : ans) {
            std::cout << name << " ";
        }
    }

    return 0;
}

```

时空复杂度

时间复杂度：`O(NlogN+M)`。排序所需的时间复杂度为`O(NlogN)`，获取排序依据科目在数组`subjects`中的下标`idx`所需的时间复杂度为`O(M)`

空间复杂度：`O(NM)`。哈希表所占空间