链滴

PAT 甲级刷题实录——1012

作者：aopstudio
原文链接：https://ld246.com/article/1579683121973
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

原题链接

https://pintia.cn/problem-sets/994805342720868352/problems/994805502658068480

思路

这题看上去比较复杂，但稍微分析一下就发现其实问题很明确，逻辑并不复杂。问题问的就是每个学排名最高的科目，如果有相同的最高名次，那么输出优先级更高的科目。科目优先级从高到低的顺序：A>C>M>E。如果找不到成绩信息，则输出 N/A。

首先我们需要确定存储输入的成绩的数据结构。二维数组是比较合适的，因为我们一方面需要算出一同学每一个科目的名次并从中选出最高名次，如果我们选择二维数组的话，这个体现为对列下标进行历，非常方便，另一方面在找具体的单科名次时需要和其他同学的该科目成绩做比较，同样的，如果择二维数组的话体现为对行下标进行遍历。而如果选择其他的数据结构，则并不能进行如此方便的操。另外，因为题中给了科目优先级顺序，所以二维数组的列顺序也是这样时最方便计算，即第一列存，第二列存 C，第三列存 M，第四列存 E。具体到算法中是当后面的科目名次小于目前已知的最高名时，才更新最高名次和最好科目，如果相等的话，则不更新，这样就保证了优先级的要求。二维数组实现方式依然是我们最喜欢的 vector，即 vector<vector<int> > grade 。

另外，我们注意到成绩列表中每一列都对应一个科目，而我们并没有明确存储科目简称，在计算时我只知道列下标，因此我们需要写一个列下标转科目简称的方法。

这题还有一个注意到就是如何根据学生 ID 去查找它在二维数组中对应的成绩信息。我一开始也想写个方法进行 ID 到行下标的转换的，但是后来发现有一个叫 map 的神器非常适合这个需求。map 这东西我们其实把它理解成一个数组，操作也和数组类似，但下标不仅仅只是整数，而可以是各种数据型。它可以建立不同数据类型之间的映射关系，同时可以直接用一种数据类型的元素去找到对应的另种数据类型的元素。在本题中，我们需要建立一个 string 映射到 int 的 map，即 map<string,int> index 。关于 map 的用法大家可以上网去搜索，这里就不展开讲了。

最后的结果存储我是用了 vector 存储结构体，结构体包含两个元素，bestRank 和 bestCourse，分代表最高名次和最高名次对应的科目。bestRank 为-1 时代表不存在该名学生的成绩信息。

综上，代码如下

代码

```
#include <iostream>
#include <vector>
#include <string>
#include <map>
using namespace std;
char numToCourse(int num);
struct result
{
    int bestRank;    //-1代表不存在
    char bestCourse;
};
int main()
{
    int stuNum, rankNum;    //学生总数，要排名的总数
    cin >> stuNum >> rankNum;
    vector<vector<int> > grade;    //成绩矩阵
```

```
map<string,int> index;    //学生ID和在成绩矩阵中的编号对应信息
vector<result> results;
for (int i = 0; i < stuNum; i++)
{
    string id;
    int sum = 0;
    vector<int> stuGrade(4);
    cin >> id;
    index[id] = i;
    for (int j = 1; j < 4; j++)
    {
        int g;
        cin >> g;
        sum += g;
        stuGrade[j] = g;
    }
    stuGrade[0] = sum / 3;
    grade.push_back(stuGrade);
}
for (int i = 0; i < rankNum; i++)
{
    string id;
    cin >> id;
    if (index.find(id) != index.end())    //需要排序的编号在所有学生编号中
    {
        int bestRank = stuNum+1;
        char bestCourse;
        for (int i = 0; i < 4; i++)
        {
            int in = index[id];    //该ID学生对应在grade中的下标
            int g = grade[in][i];    //每次循环时依次为A,C,M,E的成绩
            int gRank = 1;    //g对应的排名
            for (int j = 0; j < stuNum; j++)    //查看当前科目成绩在所有学生中的成绩排名
            {
                if (grade[j][i] > g)
                    gRank++;
            }
            if (gRank < bestRank)    //有更高的排名
            {
                bestRank = gRank;    //更新最高排名
                bestCourse = numToCourse(i);
            }
        }
        result r;
        r.bestRank = bestRank;
        r.bestCourse = bestCourse;
        results.push_back(r);
    }
    else //找不到这个编号
    {
        result r;
        r.bestRank = -1;
        results.push_back(r);
    }
```

```
    }
    for (int i = 0; i < rankNum; i++)
    {
        if (results[i].bestRank != -1)
            cout << results[i].bestRank << ' ' << results[i].bestCourse << endl;
        else
            cout << "N/A" << endl;
    }
    return 0;
}
char numToCourse(int num)
{
    switch (num)
    {
    case 0:
        return 'A';
    case 1:
        return 'C';
    case 2:
        return 'M';
    case 3:
        return 'E';
    default:
        break;
    }
}
```