2011 浙江大学计算机考研机试题解

我要解决什么问题？
我要如何去解决？
编码！

第一题：A+B for Matrices

This time, you are supposed to find A+B where A and B are two matrices, and then count the number of zero rows and columns.

Input Specification:

The input consists of several test cases, each starts with a pair of positive integers M and N (≤10) which are the number of rows and columns of the matrices, respectively. Then 2*M lines follow, each contains N integers in [-100, 100], separated by a space. The first M lines correspond to the elements of A and the second M lines to that of B.

The input is terminated by a zero M and that case must NOT be processed.

Output Specification:

For each test case you should output in one line the total number of zero rows and columns of A+B.

Sample Input:

```
2 2
1 1
1 1
-1 -1
10 9
2 3
```

1 2 3

4 5 6

-1 -2 -3

-4 -5 -6

0

Sample Output:

1

5

解题思路：

本题需要解决的问题是求出两个矩阵相加后的结果矩阵中一共有多少零行和零列。所谓零行在线性代数中的就是一行中所有元素为 0 的行，同样零列也就是一列中所有元素为 0 的列。那么，解决本题就进行一次矩阵相加即可。用两个数组存储两个矩阵，再用一个数组存储这两个矩阵相加后的结果。然后对结果进行两次遍历，分别求出零行的数目和零列的数目。

注意这里有多组测试数据，那么怎样停止程序呢？每次读入第一个整数，判断是否为 0，如果为 0 则说明输入结束了。

步骤如下：

1）读入数据（两个矩阵）；

2）对读入的两个矩阵求和，结果存储在另一个矩阵（结果矩阵）中；

3）对结果矩阵进行两次遍历，获得零行和零列的总数。

AC 代码：

```
#include <stdio.h>

// 定义三个矩阵，分别存储将要输入的两个矩阵以及两个矩阵的和
int a[11][11];
int b[11][11];
int c[11][11];

/*
 * 将矩阵 a 和矩阵 b 相加
 * 传入的是矩阵的行数 m 以及列数 n
 * 将矩阵 a 和矩阵 b 的结果存储在矩阵 c 中
 */
void addM(int m,int n){
```

```
    for(int i=0;i<m;i++){
        for(int j=0;j<n;j++){
            c[i][j] = a[i][j] + b[i][j];
        }
    }
}

/*
 * 获得答案：遍历查看有多少零行和零列
 */
int getAns(int m, int n){
    int ans = 0;
    for(int i=0;i<m;i++){  // 先遍历零行
        int j=0;
        for(j=0;j<n;j++){  // 每行中从第一列开始
            if(c[i][j]){
                break;
            }
        }
        if(j == n){        // 如果整行遍历完了没有发现非零值，则这一行是零行
            ans ++;        // 这样的话结果增加 1
        }
    }

    for(int i=0;i<n;i++){  // 然后遍历列，方法与遍历行是一样的
        int j=0;
        for(j=0;j<m;j++){
            if((c[j][i])){
                break;
            }
        }
        if(j == m){
            ans ++;
        }
    }

    return ans;    // 返回遍历的结果
}

int main(){

    int m, n;
    while(scanf("%d", &m), m){    // 读入行数，如果某次行数为0，则说明输入结束了
```

```
        scanf("%d", &n);
        for(int i=0;i<m;i++){      // 读入第一个矩阵的数据
            for(int j=0;j<n;j++){
                scanf("%d",&a[i][j]);
            }
        }
        for(int i=0;i<m;i++){      // 读入第二个矩阵的数据
            for(int j=0;j<n;j++){
                scanf("%d",&b[i][j]);
            }
        }
        addM(m,n);                 // 求两个矩阵的和
        printf("%d\n",getAns(m,n));    // 输出结果
    }

    return 0;
}
```

第二题：Grading

Grading hundreds of thousands of Graduate Entrance Exams is a hard work. It is even harder to design a process to make the results as fair as possible. One way is to assign each exam problem to 3 independent experts. If they do not agree to each other, a judge is invited to make the final decision. Now you are asked to write a program to help this process.

For each problem, there is a full-mark P and a tolerance T(<P) given. The grading rules are:

• A problem will first be assigned to 2 experts, to obtain G1 and G2. If the difference is within the tolerance, that is, if $|G1 - G2| \le T$, this problem's grade will be the average of G1 and G2.

• If the difference exceeds T, the 3rd expert will give G3.

• If G3 is within the tolerance with either G1 or G2, but NOT both, then this problem's grade will be the average of G3 and the closest grade.

• If G3 is within the tolerance with both G1 and G2, then this problem's grade will be the maximum of the three grades.

• If G3 is within the tolerance with neither G1 nor G2, a judge will give the final grade GJ.

Input Specification:

Each input file may contain more than one test case.

Each case occupies a line containing six positive integers: P, T, G1, G2, G3, and GJ, as described in the problem. It is guaranteed that all the grades are valid, that is, in the interval [0, P].

Output Specification:

For each test case you should output the final grade of the problem in a line. The answer must be accurate to 1 decimal place.

Sample Input:

```
20 2 15 13 10 18
```

Sample Output:

```
14.0
```

解题思路：

如果能够耐心读懂本题题目，实际上本题并不难。题目中需要求的是打分问题。首先给定的是总分 P 和一个允许的差额 T。然后给定最先打出的两个分数 G1 和 G2。

1）如果 G1 与 G2 的差值不大于 T，则最终分数为 G1 与 G2 的平均分；

2）否则给出第三个分数 G3，计算 G3 与前面两个分数的差值。如果两个差值都不大于 T，则最终结果为 3 个分数中的最大值；

3）如果只有一个差值不大于 T，则最终结果为 G3 与那个同 G3 差值不大于 T 的分数的平均分；

4）如果两个差值都大于 T，则给定第四个分数 GJ，最终成绩就为 GJ。

题目本身不难，我就不过多解释了，需要注意的有三点：

1）如何读入数据，由于测试数据有多组，都是浮点数。那么可以先读取一个浮点数判断是否读到了文件结尾，如果没有读到文件结尾则说明还有一组数据，则读入剩下的数据。或者直接将六个浮点数一起读入，判断是否读到文件结尾；

2）差值是非零实数，在不知道孰大孰小的情况下做差可以使用 math.h 中的 fabs 函数。或者做个判断。

3）结果需要保留一位小数，可以使用

printf("%.1lf");

来输出，第一个是数字“1”，第二个是小写“L”。（如果使用 double 来存储浮点数的话）

AC 代码：

```
#include <stdio.h>
#include <math.h>

int main(){
```

```
    // 定义变量存储总分、差值以及四个分数
    double p, t, g1, g2, g3, gj;
    while(scanf("%lf", &p) != EOF){        // 如果没有读到文件结尾，说明还有测试数据

        // 读入剩下的测试数据
        scanf("%lf%lf%lf%lf%lf", &t, &g1, &g2, &g3, &gj);
        if(fabs(g1-g2) <= t){                  // 如果 g1 和 g2 的差值不大于 t，则输出 g1 和 g2 的平均分
            printf("%.1lf\n", (g1+g2)/2);
        }else{
            double tmp1 = fabs(g3-g1);         // 计算 g3 与 g1 的差值
            double tmp2 = fabs(g3-g2);         // 计算 g3 与 g2 的差值
            if(tmp1<=t && tmp2<=t){                // 如果两个差值均不大于 t，则输出三个分数中的最大值
                printf("%.1lf\n", g1>g2 ? (g1>g3?g1:g3) : (g2>g3?g2:g3));
            }else if(tmp1 <= t){               // 如果 g3 与 g1 的差值不大于 t，则输出 g1 与 g3 的平均分
                printf("%.1lf\n", (g1+g3)/2);
            }else if(tmp2 <= t){               // 如果 g3 与 g2 的差值不大于 t，则输出 g2 与 g3 的平均分
                printf("%.1lf\n", (g2+g3)/2);
            }else{                             // 否则两个差值均大于t，则输出另外给出的分数 gj
                printf("%.1lf\n", gj);
            }
        }
    }

    return 0;
}
```

第三题：Median

Given an increasing sequence S of N integers, the median is the number at the middle position. For example, the median of S1={11, 12, 13, 14} is 12, and the median of S2={9, 10, 15, 16, 17} is 15. The median of two sequences is defined to be the median of the non-decreasing sequence which contains all the elements of both sequences. For example, the median of S1 and S2 is 13.

Given two increasing sequences of integers, you are asked to find their median.

Input Specification:

Each input file may contain more than one test case.

Each case occupies 2 lines, each gives the information of a sequence. For each sequence, the first positive integer N (≤1000000) is the size of that sequence. Then N integers follow, separated by a space.

It is guaranteed that all the integers are in the range of long int.

Output Specification:

For each test case you should output the median of the two given sequences in a line.

Sample Input:

```
4 11 12 13 14

5 9 10 15 16 17
```

Sample Output:

```
13
```

解题思路：

本题需要解决的问题是给定两个有序序列，求出这两个序列合并在一起以后中间的那个值。由于两个序列都是有序的，自然我们会想到归并排序的归并过程。于是问题迎刃而解了。

解题步骤：

1）读入两个有序序列的值；

2）计算中间值得位置；

3）定位中间值；

4）输出结果。

需要注意的是两个序列的元素都非常多。因而可以考虑用全局变量来存储：定义在函数中的局部变量存储在系统的栈当中，全局变量或静态变量则存储在堆当中。栈的存取比堆快，但栈的空间少于堆。

AC 代码：

```
#include <stdio.h>

int a[1100000], b[1100000];    // 定义储存两列序列的数组，由于元素很多，所以使用全局变量来存储

int main(){
```

```
    int m;      // 第一个序列的元素数目
    while(scanf("%d", &m) != EOF){
        for(int i=0;i<m;i++){ // 读入第一个序列中的元素
            scanf("%d", &a[i]);
        }
        int n; // 第二个序列中的元素数目
        scanf("%d", &n);
        for(int i=0;i<n;i++){ // 读入第二个序列中的元素
            scanf("%d", &b[i]);
        }
        int t = (m+n-1)/2; // 在总的序列中中间的那个数前面有多少元素
        int i=0, j=0;
        while(t--){// 定位到中间的那个数
            if(a[i]<b[j] && i<m || j>=n){
                i++;
            }else{
                j++;
            }
        }
        printf("%d\n",a[i]<b[j] && i<m ? a[i] : b[j]); // 输出结果
    }

    return 0;
}
```

第四题：Graduate Admission

It is said that in 2011, there are about 100 graduate schools ready to proceed over 40,000 applications in Zhejiang Province. It would help a lot if you could write a program to automate the admission procedure.

Each applicant will have to provide two grades: the national entrance exam grade GE, and the interview grade GI. The final grade of an applicant is (GE + GI) / 2. The admission rules are:

- The applicants are ranked according to their final grades, and will be admitted one by one from the top of the rank list.
- If there is a tied final grade, the applicants will be ranked according to their national entrance exam grade GE. If still tied, their ranks must be the

same.

• Each applicant may have K choices and the admission will be done according to his/her choices: if according to the rank list, it is one's turn to be admitted; and if the quota of one's most preferred shcool is not exceeded, then one will be admitted to this school, or one's other choices will be considered one by one in order. If one gets rejected by all of preferred schools, then this unfortunate applicant will be rejected.

• If there is a tied rank, and if the corresponding applicants are applying to the same school, then that school must admit all the applicants with the same rank, even if its quota will be exceeded.

Input Specification:

Each input file may contain more than one test case.

Each case starts with a line containing three positive integers: N (≤40,000), the total number of applicants; M (≤100), the total number of graduate schools; and K (≤5), the number of choices an applicant may have.

In the next line, separated by a space, there are M positive integers. The i-th integer is the quota of the i-th graduate school respectively.

Then N lines follow, each contains 2+K integers separated by a space. The first 2 integers are the applicant's GE and GI, respectively. The next K integers represent the preferred schools. For the sake of simplicity, we assume that the schools are numbered from 0 to M-1, and the applicants are numbered from 0 to N-1.

Output Specification:

For each test case you should output the admission results for all the graduate schools. The results of each school must occupy a line, which contains the applicants' numbers that school admits. The numbers must be in increasing order and be separated by a space. There must be no extra space at the end of each line. If no applicant is admitted by a school, you must output an empty line correspondingly.

Sample Input:

```
11 6 3

2 1 2 2 2 3

100 100 0 1 2

60 60 2 3 5
```

100 90 0 3 4

90 100 1 2 0

90 90 5 1 3

80 90 1 0 2

80 80 0 1 2

80 80 0 1 2

80 70 1 3 2

70 80 1 2 3

100 100 0 2 4

Sample Output:

0 10

3

5 6 7

2 8

1 4

解题思路：

这一题是解决学校录取考生的问题。只要对考生按一定的顺序排序，然后依次判断考生能否进入自己所选择的学校。

排序是按统考分数 GE 与面试分数 GI 的均分非递增进行的，如果均分相同则按统考分数 GE 非递减进行排序。每个学校都有一定的名额限制，如果有考生与最后一名的排名相同（均分及统考分均相同）则学校需要扩招。

解题步骤如下：

1）读取考生信息，注意记录考生均分及序号；

2）对考生进行排序；

3）对每位考生进行遍历，判断该考生是否能够被自己所选择的学校录取，如果能够录取则

对相应的学校添加记录；
4）输出每个学校的信息，按考生的序号非递增输出。

AC 代码：

```
#include <cstdio>
#include <algorithm>
using namespace std;

// 定义每个考生的信息结构
struct StuType{
    double ge, gi, a;  // 统考分数、面试分数，以及均分
    int id;    // 考生的序号
    int s[5];  // 考生选择的学校（最多 5 个）
};

// 定义比较函数，用来对考生信息进行排序
bool cmp(StuType a, StuType b){
    if(a.a == b.a){            // 如果均分相同
        return a.ge > b.ge;        // 则比较统考分数
    }
    return a.a > b.a;          // 否则比较均分
}

int schools[110][44000];       // 记录学校招收的考生序号
StuType students[44000];       // 记录所有的考生信息

int main(){

    int n;
    while(scanf("%d",&n) != EOF){ // 读取考生数目
        int m, k;
        scanf("%d%d", &m, &k);     // 读取学校数目以及考生可选学校的数目
        int numOfSchool[110];      // 存储每个学校能够招收考生的数目
        for(int i=0;i<m;i++){
            scanf("%d",&numOfSchool[i]);  // 读取每个学校能够招收考生的数目
        }
        for(int i=0;i<n;i++){
            scanf("%lf%lf", &students[i].ge, &students[i].gi);   // 读取考生统考成绩以及面试成绩
            students[i].a = (students[i].ge+students[i].gi)/2;   // 计算统考成绩和面试成绩的均分
            for(int j=0;j<k;j++){
                scanf("%d",&students[i].s[j]);        // 读入考生选取的学校序号
```

```
            }
            students[i].id = i;        // 记录考生的序号
        }
        sort(students, students+n, cmp);  // 按先均分后统考分数的顺序进行排序
        int numOfSche[110] = {0};          // 记录每个学校已经招收的考生
        double gradeOfLast[110][2];            // 如果需要扩招的话，记录扩招时需要的均分和统考分数，
                                           // 因为只有均分和统考分数不小于（实际上是相等）扩招时才能够被扩招
        for(int i=0;i<n;i++){              // 对每个考生进行遍历
            for(int p=0;p<k;p++){          // 遍历考生选择的每个学校
                int j = students[i].s[p]; // 获得考生选择学校的编号
                if(numOfSche[j]<numOfSchool[j]){ // 判断目前该学校是否招生已满，如果没满，则招收之
                    schools[j][numOfSche[j]++] = students[i].id; // 记录学校招收考生的序号
                    if(numOfSche[j] == numOfSchool[j]){              // 如果正好招生已满，
                        gradeOfLast[j][0] = students[i].a;           // 则记录最后一个考生的均分
                        gradeOfLast[j][1] = students[i].ge;          // 以及统考分
                    }
                    break;    // 注意，一旦考生被选择的某个学校录取了，后面的学校就不能再录取了
                }else if(students[i].a==gradeOfLast[j][0] && students[i].ge==gradeOfLast[j][1]){  // 如果招生已满，则判断该考生的均分和统考分是否与最后一名相同，如果相同则应当扩招进来
                    schools[j][numOfSche[j]++] = students[i].id; // 记录学校招收考生的序号
                    break;
                }
            }
        }
        for(int i=0;i<m;i++){      // 将学校的考生信息输出
            sort(schools[i], schools[i]+numOfSche[i]);   // 注意要求按考生序号递增输出
            for(int j=0;j<numOfSche[i];j++){
                if(j){             // 除了第一个考生序号前面不需要输出空格，后面的需要前面都需要输出一个空格
                    putchar(' ');
                }
                printf("%d",schools[i][j]);   // 输出考生序号
```

```
            }
            putchar('\n');
        }
    }

    return 0;
}
```