中国人民大学
RENMIN UNIVERSITY OF CHINA

期中复习专题-4

（统计分析）

余力

buaayuli@ruc.edu.cn

#195 平均成绩排序

有 n 位学生，每位学生修读的科目数不尽相同，已知所有学生的各科成绩，要求按学生平均成绩由高到低输出学生的学号、平均成绩；当平均成绩同时，按学号从低到高排序。对平均成绩，只取小数点后前 2 位，从第 3 位开始舍弃（无需舍入）。

输入格式

□□输入为 n+1 行，第一行为 n 表示学生人数。

□□从第二行开始的 n 行，每行为一名学生的成绩信息，包括：学号、科目数，各科成绩。其中 n、学号、成绩均为整数，它们的值域为：0≤n≤10000，1≤学号≤1000000，0≤成绩≤100。学生的科目数都不超过 100 门。

输出格式

□□最多 n 行，每行两个数，学号在前，后为平均成绩，空格分隔。若 n 为 0，输出 NO；若某学生所修科目不到 2 门，则不纳入排序，若无人修满 2 门，也输出 NO。

输入样例	输出样例
5	2003 93.75
1001 2 89 78	1001 83.50
2003 4 88 99 100 88	1004 72.66
4004 3 72 80 66	4004 72.66
1004 3 70 66 82	
3001 1 100	

```
int main() {
    int id[10000];
    double aver[10000], tmpf;
    int n, i, j, StuNo, KemuNum, tmp, sum, count = 0;

    scanf("%d", &n);
    // 输入 n 位学生信息
    for (i = 0; i < n; i++) {
        scanf("%d %d", &StuNo, &KemuNum);
        for (sum = 0, j = 0; j < KemuNum; j++) {
            scanf("%d", &tmp);
            sum = sum + tmp; }
        if (KemuNum >= 2) {
            id[count] = StuNo;
            aver[count] = sum * 1.0 / KemuNum;
            count++; }
    }
```

① 输入

② 统计

```
if (count == 0) {
    printf("NO");
    return 0; }
else {
    for (i = 0; i < count - 1; i++)
        for (j = 0; j < count – i-1; j++)
            if ((fabs(aver[j] - aver[j + 1]) < 1e-7 && id[j] > id[j + 1]) || aver[j] < aver[j + 1]) {
                tmp = id[j]; id[j] = id[j + 1]; id[j + 1] = tmp;
                tmpf = aver[j]; aver[j] = aver[j + 1]; aver[j + 1] = tmpf; }
    for (i = 0; i < count; i++)
        printf("%d %.2lf\n", id[i], int(aver[i] * 100) / 100.0 );

    return 0;
    }
```

③

排序

+Chp05_平均成绩排序(#195).cpp

```
scanf ("%d", &n);
for (i = 0; i < n; i++) {// 输入n个信息
    scanf ("%d %d", &StuNo, &KemuNum);
    for (sum = 0, j = 0; j < KemuNum; j++) {
        scanf ("%d", &tmp);
        sum = sum + tmp;  }
    if ( ** ) { id[count]; aver[count];count++;}
}
// 输出结果
if ( count  == 0 )
        printf ("NO");
else {
        for (i = 0; i <count-1; i++)
        for (j = 0; j <count-1-i; j++)
                {                       }

        for (i = 0; i < count; i++)
            printf(    );
    }
```

典型

统计排序

程序框架

a[0]---a[n-1] n 个数排序

```
for (i = 0; i < n - 1; i++)
    for (j = 0; j < n - 1 - i; j++)
        if (a[j] < a[j + 1] )
            { tmp = a[j]; a[j] = a[j + 1]; a[j + 1] = tmp; }
```

i	0	n-1
j	0	n-1-i

为避免搞混，
大家可以
就记这一种

a[1]---a[n] n 个数排序

```
for (i = 1; i <= n-1; i++)   // i=1,2,3,...,n-1
    for (j = 1; j <= n – i; j++)   // j=1,2,...,n-i
        if (a[j] < a[j + 1] )
            { tmp = a[j]; a[j] = a[j + 1]; a[j + 1] = tmp; }
```

排序模块

多排序依据

```
for (i = 0; i < n - 1; i++)
    for (j = 0; j < n - 1 - i; j++)
        if (aver[j] < aver[j + 1] || (fabs(aver[j] - aver[j + 1]) < 1e-7 && id[j] > id[j + 1])) {
            tmp = id[j]; id[j] = id[j + 1]; id[j + 1] = tmp;
            tmpf = aver[j]; aver[j] = aver[j + 1]; aver[j + 1] = tmpf; }
```

(aver[j] < aver[j+1] || (fabs(aver[j] -

aver[j+1]) < 1e-7 && id[j] > id[j+1]))

#308 猪场分配

老赵经营着一家现代化畜牧养殖企业，在全国各地建有 n 家专业养猪场。但是由于受非洲猪瘟的影响，2019 年损失较为惨重。在国家政策和市场需求的激励下，老赵决定分三个批次购入仔猪，恢复生产。为了杜绝疫病交叉感染的潜在风险，同一个批次的只能放在同一个养殖场。由于不同场地的养殖成本与容量不同，现在他需要考虑如何安排养殖场，以实现最佳的经济效益。假定各批次出栏时，市场预期价格一致，根据输入的养殖场现状信息，编程找出一种总成本最少的猪场分配方案（不是成本最少的所有方案，见输出说明）。

【输入格式】

第 1 行 3 个整数，分别表示三个批次的仔猪数量。

第 2 行 1 个整数，表示老赵经营的养殖场数 n。

第 3 行开始，共 n 行，每行 5 个整数，依次表示：养殖场的编号、运营状态、最大养殖容量、运营基础成本和每头仔猪的出栏养殖成本。其中运营状态用 0、1 表示，1 表示已在运营，不能安排其它批次仔猪进场。比如：112 1 3000 5000 300。

【输出格式】

如果找到最少成本的方案，输出两行，第 1 行只 1 个数，为最少成本；第 2 行 3 个数，为对应该成本的分配方案，即依仔猪批次顺序，输出养殖场的编号。这些编号是在所有可能最佳方案中，各批次可能分配的猪场最小编号。

如果找不到，输出 NO。

枚举+统计排序

```
int Farm_Total, Farm_Count, P1_Num, P2_Num, P3_Num, P1_id, P2_id, P3_id;
int ID[3001], rongliang[3001], base_cost[3001], each_cost[3001], data[3001][5];
int i, j, k, find;
int cost_sum, min_cost = 10000000;

scanf("%d%d%d%d", &P1_Num, &P2_Num, &P3_Num, &Farm_Total);

for (i = 0, Farm_Count = 0; i < Farm_Total; i++) {
    for (j = 0; j < 5; j++)
        scanf("%d", &data[i][j]);
    if (data[i][1] == -1) continue;
    ID[Farm_Count] = data[i][0];
    rongliang[Farm_Count] = data[i][2];
    base_cost[Farm_Count] = data[i][3];
    each_cost[Farm_Count] = data[i][4];
    Farm_Count++;//记录可用场子数
}
```

①

```
for (i = 0; i < Farm_Count; i++) {
    if (rongliang[i] < P1_Num)  continue;
    for (j = 0; j < Farm_Count; j++) {
        if (rongliang[j] < P2_Num || j == i)  continue;
        for (k = 0; k < Farm_Count; k++) {
            if (rongliang[k] < P3_Num || k == j || k == i)  continue;
            cost_sum = base_cost[i] + base_cost[j] + base_cost[k];
            cost_sum = cost_sum + each_cost[i] * P1_Num + each_cost[j] * P2_Num + each_cost[k] * P3_Num;
            find = 0;
            if (cost_sum < min_cost)    find = 1;

            else if (cost_sum == min_cost)
                if (ID[i] < P1_id) find = 1;
                else if (ID[i] == P1_id)
                    if (ID[j] < P2_id) find = 1;
                    else if (ID[j] == P2_id && ID[k] < P3_id)    find = 1;
            if (find == 1) {
                min_cost = cost_sum;
                P1_id = ID[i]; P2_id = ID[j]; P3_id = ID[k];}
        }
    }
}
```

不符合条件的直接过滤

②

计算方案

比较方案

设置为最优方案

#260 二分查找

第一行一个数 n，表示需要输入的正整数个数。

第二行一个数 m (1≤m≤2^30)，表示待查找的整数。

□□第三行包含 n 个正整数（≤2^30），每两个整数之间用一个空格隔开，是给定集合中的 n 个元素，输入数据保证这 n 个整数互不相等。

输出格式

□□输出共两行。

□□第一行包含一个整数 k，表示待查找的数在排序后的**集合中的位置**（位置从 0 ~ n-1 标号），如果没有找到则输出-1。

□□第二行包含一个整数,表示查找成功前的比较次数。

输入样例 1

10

24

42 24 10 29 27 12 58 31 8 16

输出样例 1

4

1

过程模拟

特殊提示

【样例 1 说明】

排序后的集合 a 为：8 10 12 16 24 27 29 31 42 58

待查找的数 24，第一次与 a[4]比较，a[4]是 24，找到元素，算法结束，所以比较次数为 1。

```
#include <stdio.h>
int main() {
    int n, i, j, mid, got=0, times=0;
    long int a[5000], m, t;
    scanf("%d%ld", &n, &m);
    for (i = 1; i <= n; i++)
        scanf("%ld", &a[i]);
    for (i = 1; i <= n; i++)
        for (j = 1; j <=n- i; j++)
            if (a[j] > a[j+1])
                {t=a[j]; a[j]= a[j+1]; a[j+1]=t;}
    for(i=1, j=n, mid = (i+j)/2; j>=i; ) {
        times++;
        if (a[mid ] < m)  i=mid+1;
        else if (a[mid ] > m)  j=mid-1;
        else { got = 1;  break; }
        mid  = (i + j) / 2; }
    if (got==1) printf("%d\n%d", mid-1, times);
    else  printf("-1\n%d", times);
    return 0;
}
```

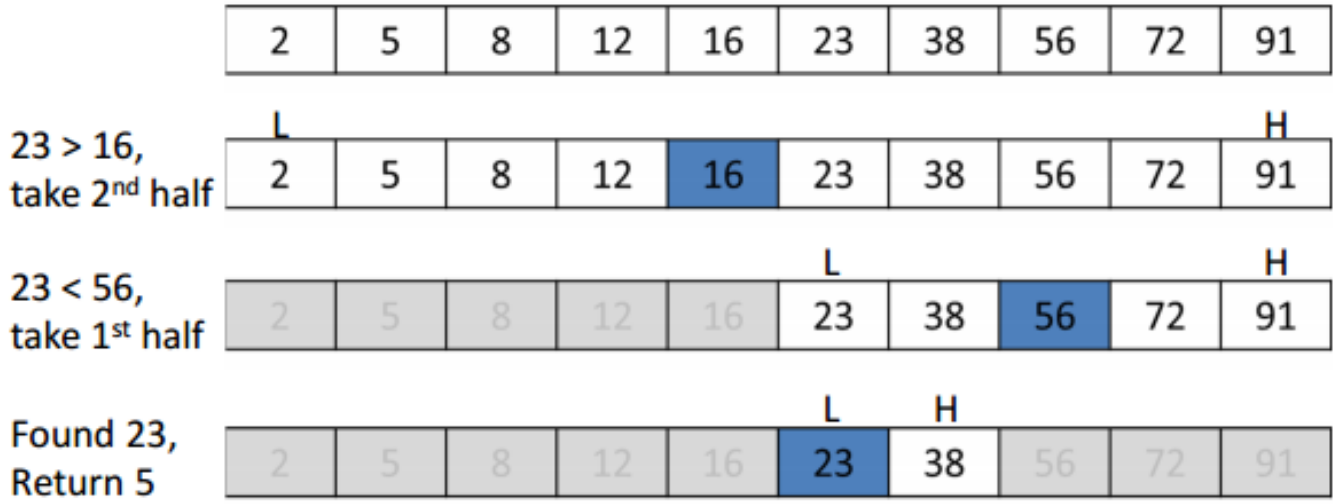
①

②

③

```
for(i=1, j=n, mid = (i+j)/2; j>=i; ) {
    times++;
    if (a[mid ] < m)  i=mid+1;
    else if (a[mid ] > m)  j=mid-1;
    else { got = 1;  break; }
    mid  = (i + j) / 2; }
```

If searching for 23 in the 10-element array:




```
int L=0
int  R=n-1;
int cmp=0;
int found=-1;
while（L<=R）{
        int mid=(L+R)/2;
        cmp++;
        if（m<arr[mid] ) R=mid-1;
        else if (m>arr[mid] ) L=mid+1;
                else  { found=mid; break;  }
}
```

```
int main() {
    int n, i, j, k, x, y; //x y 表示每次查找的区间
    long long int a[1000], m, tmp;
    scanf("%d%lld", &n, &m);
    for (i = 0; i < n; i++)
        scanf("%lld", &a[i]);
    for (i = 0; i <= n - 2; i++)
        for (j = 0; j <= n - 2 - i; j++)
            if (a[j] > a[j + 1]) { tmp = a[j]; a[j] = a[j + 1]; a[j + 1] = tmp; }
    x = 0, y = n-1, k = 0;
    while ( a[(y+x)/2] !=m && (x != y) )
        if (a[(y+x)/2]>m) { y = (y+x)/2-1; k++; }
        else {x = (y+x)/2+1; k++;}
    if ((x == y) && (m != a[x]))  printf("-1\n%lld", k + 1);
    if (a[(y + x) / 2] == m)   printf("%d\n%d", (y + x) / 2, k + 1);
}
```

测试点 #7 2

输入文件 (data8.in)

```
603
26536
7616 9307 19781 13830 17461 10120 24159
<3336 bytes omitted>
```

输出文件 (data8.out)

```
-1
9
```

选手输出

#307 生辰八字

大富商杨家有一女儿到了出阁的年纪，杨老爷决定面向全城适龄男士征婚。杨老爷遵循传统，决定按生辰八字作为选女婿的依据。每个应征的男士须提供自己的生辰八字，亦即八个正整数，每个数的取值范围均在[1,24]。杨老爷特意聘请了黄半仙来算命，选择和女儿最契合的前 k 个男士作为候选。黄半仙采用的算命方法是西洋传入的余弦相似度，具体做法如下：

给定两个生辰八字 A=[a_1, a_2,...,a_8]，B=[b_1, b_2,...,b_8]，则

$$Similarity(A,B) = \sum_{i=1}^{8} a_i * b_i / \left(\sqrt{\sum_{i=1}^{8}(a_i)^2} * \sqrt{\sum_{i=1}^{8}(b_i)^2}\right)$$

例如，若 A=[10,1,1,1,1,1,1,1]，B=[1,1,1,1,1,1,1,1]，则 Similarity(A,B)=(10*1+1*1+...1*1)/(sqrt(10^2+1^2+...1^2)+sqrt(1^2+1^2+...+1^2))=1.29

黄半仙认为一个男士和小姐两人的生辰八字的余弦相似度越大，两人就越契合。

【输入格式】

第 1 行两个整数，n 和 k，(1≤n≤100,1≤k≤n)，表示有 n 个男士应征，以及需要选择 k 人进入候选名单。

第 2 行 8 个整数，表示杨小姐的生辰八字。

后面 n 行，每行 9 个整数，第一个数字是某男士的身份证号，8 位整数；后面 8 个数字是该男士的生辰八字。整数之间均以空格隔开。

【输出格式】

k 个整数，表示契合度最好的前 k 位男士的身份证号，按相似度从高到低排列。

注意：若两个男士和小姐的相似度相同，则身份证号码大的一个排在前面。当相似度相差小于 1e-10 时，认为相同。

【样例输出】

```
1012345678
```

【样例输入】

```
2 1
1 1 1 1 1 1 1 1
1012345678 1 1 1 1 1 1 1 1
1087654321 1 1 1 1 8 8 8 8
```

```
int main() {
    int top_k, num, i, j, k, woman[9], man[9], man_id[101], tmp_id;
    double sim[101], tmp1, tmp2, tmp3, tmp_sim;
    scanf("%d%d", &num, &top_k);
    for (k = 1; k <= 8; k++)
        scanf("%d", &woman[k]);  //女生生辰八字
    for (i = 1; i <= num; i++) {
        scanf("%d", &man_id[i]); // 男生身份证号
        for (k = 1; k <= 8; k++)
            scanf("%d", &man[k]); // 男生生辰八字
        tmp1 = 0; tmp2 = 0; tmp3 = 0;
        for (k = 1; k <= 8; k++) {
            tmp1 = tmp1 + woman[k] * man[k];
            tmp2 = tmp2 + woman[k] * woman[k];
            tmp3 = tmp3 + man[k] * man[k];
        }
        sim[i] = tmp1 / (sqrt(tmp2) * sqrt(tmp3)); // 相似度
    }
    for (i = 1; i <= num; i++)
        for (j = 1; j <= num - i; j++)
            if (sim[j] < sim[j + 1] || (fabs(sim[j] - sim[j + 1]) < 1e-10 && man_id[j] < man_id[j + 1])) {
                tmp_id = man_id[j]; man_id[j] = man_id[j + 1]; man_id[j + 1] = tmp_id;
                tmp_sim = sim[j]; sim[j] = sim[j + 1]; sim[j + 1] = tmp_sim; }
    for (i = 1; i <= top_k; i++)
        printf("%d ", man_id[i]);
    return 0;
}
```

①

②

```
    scanf("%d%d", &num, &top_k);
    for (k = 1; k <= 8; k++)
        scanf("%d", &woman[k]);    //女生生辰八字
    for (i = 1; i <= num; i++) {
        scanf("%d", &man_id[i]); // 男生身份证号
        for (k = 1; k <= 8; k++)
            scanf("%d", &man[k]); // 男生生辰八字
        tmp1 = 0; tmp2 = 0; tmp3 = 0;
        for (k = 1; k <= 8; k++) {
            tmp1 = tmp1 + woman[k] * man[k];
            tmp2 = tmp2 + woman[k] * woman[k];
            tmp3 = tmp3 + man[k] * man[k];
        }
        sim[i] = tmp1 / (sqrt(tmp2) * sqrt(tmp3)); // 相似度
    }
```

①

②

相似性
计算模块

向量的内积（点乘）

$$a=[a_1,a_2,\ldots a_n] \quad b=[b_1,b_2,\ldots b_n]$$

a和b的点积公式为：

$$a \bullet b = a_1b_1 + a_2b_2 + \ldots + a_nb_n$$

$$a \bullet b = |a||b|\cos\theta$$

$$? = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$d_{Euclidean}(x,y) = d_{Euclidean}(y,x) = \sqrt{|x_1 - y_1|^2 + |x_2 - y_2|^2 + \cdots + |x_n - y_n|^2}$$

$$= \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}$$

```
for (i = 1; i <= num; i++)
    for (j = 1; j <= num - i; j++)
        if (sim[j] < sim[j + 1] || (fabs(sim[j] - sim[j + 1]) < 1e-10 && man_id[j] < man_id[j + 1])) {
            tmp_id = man_id[j]; man_id[j] = man_id[j + 1]; man_id[j + 1] = tmp_id;
            tmp_sim = sim[j]; sim[j] = sim[j + 1]; sim[j + 1] = tmp_sim; }
```

if(sim[j]<sim[j+1]||(fabs(sim[j]- sim[j+1])<1e-10

&& man_id[j]<man_id[j+1]))

#91 相似乐曲

所谓“与 Q 最相似的 k 首乐曲”，即与 Q 的欧几里得距离最小的前 k 首乐曲。

【输入格式】

- 第 1 行，表示查询乐曲，一个正整数 n0（1≤n0≤100），表示乐曲长度，后面有 n0 个整数，每个整数在[0, 255]内，表示一个频率。
- 第 2 行，两个整数 n 和 k，用空格隔开。表示有 n 首乐曲，1≤n≤100，查找最相似的 k (1≤k≤n)首乐曲。
- 第 3 行到 n+2 行，表示编号从 0 到 n-1 的 n 首乐曲。每行一个正整数 ni (1≤ni≤100), 表示该乐曲长度，后面 ni 个整数，每个整数在[0, 255]内，表示一个频率。

【输出格式】

输出 k 个整数，与查询乐曲最相似的乐曲的编号，注意乐曲编号范围是[0, n-1]。按相似度从高到低（即：欧式距离从小到大）的顺序输出。若两首乐曲与 Q 的距离相同，则编号小的排名靠前。

【输入样例】

```
4 10 12 245 245
3 1
6 2 4 2 250 250 250
1 189
4 10 12 245 245
```

【输出样例】

```
2
```

```
[0, 15]:2
[16, 31]:0
.....
[240, 255]:2
```

```
#include <stdio.h>
#include <math.h>
int main() {
    int num, top, i, j, k, Len, sum, tmp_id;
    int ID[100], freq[100], count_0[16], count_x[16];
    float tmp_sim, sim[100];

    scanf("%d", &Len); //曲长
    for (i = 0; i < Len; i++)
        scanf("%d", &freq[i]);
    for (j = 0; j < Len; j++)
        for (k = 0; k < 16; k++)
            if (freq[j] >= 0 + k * 16 && freq[j] <= 15 + k * 16)
                count_0[k]++;//识别曲每个区间里的频率
```

count_0[freq[j]%16]++;

```
scanf("%d %d", &num, &top); //n 首曲子, 找 k 个最相近
for (i = 0; i < num; i++) {
    scanf("%d", &Len);
    for (j = 0; j < Len; j++)
        scanf("%d", &freq[j]);
    for (j = 0; j < Len; j++)
        for ( k = 0; k < 16; k++ )
            if ( freq[j] >= 0 + k * 16 && freq[j] <= 15 + 16 * k )
                count_x[k]++;//每首曲子每个区间的频率
        //count_x[freq[j]%16]++
    for (j = 0, sum = 0; j < 16; j++)
        sum = sum + (count_0[j] - count_x[j]) * (count_0[j] - count_x[j]);
    sim[i] = sqrt(sum); //第 i 首曲子的相似度
    ID[i] = i; //编号
}
```

①

②

③

```
for (i = 0; i < num - 1; i++)
    for (j = 0; j < num - 1 - i; j++)
        if (sim[j]<sim[j+1]||((abs(sim[j+1]-sim[j])<1e-6)&&ID[j]>ID[j+1]))
        {tmp_sim = sim[j]; sim[j] = sim[j + 1]; sim[j + 1] = tmp_sim;
        tmp_id = ID[j]; ID[j] = ID[j + 1];ID[j + 1] = tmp_id; }
for (i = 0; i < top; i++)
    printf("%d ",ID[i]);//要有空格
return 0;
```

#90 刷题高手

老师给了题目的列表，要求同学们在 YOJ 上做题。一段时间后，老师想统计一下同学们刷题，找出刷题高手。

【输入格式】

第一行，多个整数，第一个整数 n 表示老师要求的题目个数(1≤n≤100)。后面有 n 个整数，表示老师要求的题号。

第二行，两个整数 m 和 k，表示有 m 个学生 (1≤m≤100)，以及老师希望找出刷题最多的 k 个同学 (1≤k≤m)。

后面 m 行，每行格式如下：一个正整数 sno，表示同学的学号，8 位数字；一个正整数 p，表示该同学刷了多少题，0≤p≤100；后面 p 个正整数，表示该同学成功通过的题目编号。

【输出格式】输出为一行，至少 k 个整数，表示刷题数量最多的前 k 名同学的学号，按刷题数量从高到低排列，输出的学号之间用一个空格分隔。

注意 1) 若同学们刷的题目不是老师所要求的，则不计入成绩。2) 允许并列。例如若 k=3，且有多位同学并列排名第 2，则成绩第 3 的同学也应该输出。当有并列情况时，有可能出现所有学生的排名均小于 k。此时，将所有同学学号按成绩高低输出即可。3) 出现并列时，学号较小的同学先输出。

【输入样例】

5 1 3 5 7 9 //老师要求的题号

3 1

10016655 5 2 4 6 8 10

10055236 1 5

10001799 4 1 2 4 6

【输出样例】

10001799 10055236

```
int main() {
    struct Student {
        int StuNo; //8 位学号
        int num;    //刷题个数
        int PassNo[100];  //通过的题目编号
        int count;        //符合要求的题目数
    };
    int Required_N, i, j, k;
    int Total, Top_k; //共 Total 个学生，刷题最多 Top_k 个同学
    scanf("%d", &Required_N);  //老师要求的题目个数
    int Test[Required_N];
    for (i = 0; i < Required_N; i++)
        scanf("%d", &Test[i]);  //老师要求的题号
    scanf("%d %d", &Total, &Top_k);
    struct Student stu[Total];
    struct Student temp;
    for (i = 0; i < Total; i++) {
        scanf("%d", &stu[i].StuNo);
        scanf("%d", &stu[i].num);
        for (j = 0; j < stu[i].num; j++)
            scanf("%d", &stu[i].PassNo[j]);
        stu[i].count = 0;
    }
```

结构体风格

①

检查两个数组
的元素是重叠

```
//找出每个同学通过的符合老师要求的题目数
for (i = 0; i < Total; i++)
    for (j = 0; j < stu[i].num; j++)
        for (k = 0; k < Required_N; k++)
            if (stu[i].PassNo[j] == Test[k]) stu[i].count++;
//冒泡排序，从高到低；并列情况按学号排序
for (i = 0; i < Total - 1; i++)
    for (j = 0; j < Total - i - 1; j++)
        if ((stu[j].count < stu[j + 1].count) ||
        ·····(stu[j].StuNo > stu[j + 1].StuNo && stu[j].count == stu[j + 1].count)) {
            temp = stu[j];stu[j] = stu[j + 1];stu[j + 1] = temp;}
//输出前 Top_k 结果，关键!!!
for (i = 0, j = 0; i < Top_k && j < Total; j++) {
    printf("%d ", stu[j].StuNo);
    if (stu[j].count > stu[j + 1].count) i++;
}
return 0;
}
```

②

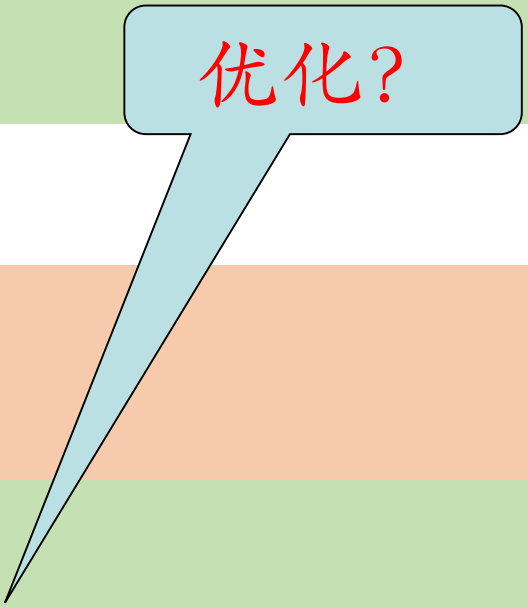
双排序

③

④

双标准输出

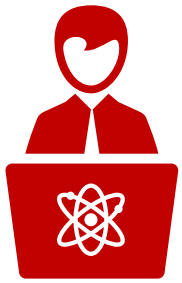
```
#include <stdio.h>
int main() {
    int ques_num, stu_num, ans_num, top_k, i, j, k, tmp;
    int sno[102], ans[102], ques[102], ture[102] = {0};

    scanf("%d", &ques_num);
    for (i = 0; i < ques_num; i++)
        scanf("%d", &ques[i]);
    scanf("%d %d", &stu_num, &top_k);

    for (i = 0; i < stu_num; i++) {
        scanf("%d %d", &sno[i], &ans_num);
        for (j = 0; j < ans_num; j++)
            scanf("%d", &ans[j]);
        for (j = 0; j < ans_num; j++)
            for (k = 0; k < ques_num; k++)
                if (ques[k] == ans[j]) ture[i]++;
    }
```



```
for (i = 0; i < stu_num; i++)
    for (j = 0; j < stu_num - 1 - i; j++)
        if ((ture[j + 1] > ture[j]) || (sno[j] > sno[j + 1] && ture[j + 1] == ture[j])) {
            tmp = ture[j];ture[j] = ture[j + 1];ture[j + 1] = tmp;
            tmp = sno[j];sno[j] = sno[j + 1];sno[j + 1] = tmp; }
```

```
for (i = 0, j = 0; i < top_k && j < stu_num; j++) {
    printf("%d ", sno[j]);
    if (ture[j] > ture[j + 1]) i++;}
```

中国人民大学
RENMIN UNIVERSITY OF CHINA



谢谢大家！