



中國人民大學
RENMIN UNIVERSITY OF CHINA

期中复习专题-1

(数值计算、统计分析)

余力

buaayuli@ruc.edu.cn

期中复习专题 (10-29上机)

- #161 数据加密 (循环)
- #310 同构数 (循环)
- #195 平均成绩排序 (数组、排序)
- #307 生辰八字 (数组、排序)
- #260 二分查找 (数组、排序、循环)

数值计算类

统计排序类

对称循环类

#161 数据加密

某个公司采用公用电话传递数据，数据是四位的整数，数据在传递过程中是加密的：每位数字都加上 5，得到的结果除以 10 的余数代替该数字，再将第一位和第四位交换，第二位和第三位交换。请你编写程序按照上述规则加密数据。

输入格式

□□输入只有一行，包括一个 4 位数的正整数 d ($1000 \leq d \leq 9999$)，表示加密前的数据。

输出格式

□□输出只有一行，也是一个 4 位数的正整数，表示加密后的数据。

输入样例

1235

输出样例

876

特殊提示

□□【样例 1 说明】

□□1235 每位上数字加 5 后模 10 得到的新数字是 6780，按照要求第一位第四位交换，第二位第三位交换后是 876（先导 0 不输出）。

读取每一位

#161 数据加密

```
#include <stdio.h>
int main()
{
    int n, m=0;
    scanf("%d", &n);
    m += (n%10+5)%10*1000;
    m += ((n/10%10)+5)%10*100;
    m += ((n/100%10)+5)%10*10;
    m += ((n/1000)+5)%10;
    printf("%d", m);
    return 0;
}
```

数位模块

#161 数据加密

```
#include <stdio.h>+  
main()+  
{  
    int a[4],m,n,i,c;+  
    scanf("%d",&c);+  
    for(i=0;i<4;i++)+  
    {  
        a[i]=(c%10+5)%10;+  
        c=c/10;    }+  
    m=a[0];    n=a[1];+  
    a[0]=a[3];    a[1]=a[2];+  
    a[3]=m;    a[2]=n;+  
    printf("%d",a[0]+10*a[1]+100*a[2]+1000*a[3]);+  
}
```

#161 数据加密

```
int main() {
```

```
→ int i, j, n;
```

```
→ scanf("%d", &n);
```

```
→ int x[4], y[4];
```

```
→ x[3] = n % 10;
```

```
→ x[2] = n / 10 % 10;
```

```
→ x[1] = n / 100 % 10;
```

```
→ x[0] = n / 1000 % 10;
```

```
→ for (i = 0; i <= 3; i++) {
```

```
→     x[i] += 5;
```

```
→     if (x[i] >= 10) x[i] = x[i] % 10;
```

```
→     x[i] = x[i] % 10;
```

```
→ }
```

数位模块

```
y[0] = x[3];
```

```
y[1] = x[2];
```

```
y[2] = x[1];
```

```
y[3] = x[0];
```

```
for (i = 0; i <= 3; i++)
```

```
→ if (y[i] == 0) → continue;
```

```
→ else → break;
```

```
for (j = i; j <= 3; j++)
```

```
→ printf("%d", y[j]);
```

```
return 0;
```

条件输出模块

#310 同构数

【问题描述】↵

计算正整数[a,b]之间的全部“同构数”之和。所谓“同构数”，是指一个正整数 n 是它平方数的尾部，则称 n 为同构数。如 6 的平方是 36，6 出现在 36 的右端，6 就是同构数。76 的平方数是 5776，76 是同构数。↵

【输入格式】↵

输入只有一行，输入两个正整数 a 和 b，中间由一个空格分隔，其中：1≤a≤b≤10000。↵

【输出格式】↵

输出一行，一个正整数，为 a、b 之间同构数之和。↵

【输入样例 1】↵ 1 100↵	【输入样例 2】↵ 80 100↵
【输出样例 1】↵ 113↵	【输出样例 2】↵ 0↵

【数据规模说明】↵

1≤a≤b≤10000。↵

76的平方数是5776，76是同构数

数位模块

```
#include <stdio.h>
#include <math.h>
int main() {
    → int from, to, n, m, weishu, sum=0, i;
    → scanf("%d%d", &from, &to);
    → for(n=from; n<=to; ++n) {
        → i=n;
        → weishu=0;
        → do {i=i/10;
        →     → weishu++;
        →     } while (i!=0);
        → m=n*n;
        → for(i=0; i<weishu; i++)
        →     → m=m/10;
        → for(i=0; i<weishu; i++)
        →     → m=m*10;
        → if(n==n*n-m) sum+=n;
    }
    → printf("%d", sum);
    → return 0;
}
```



```
for(n=from; n<=to; ++n) {
    → i=n;
    → weishu=0;
    → do {i=i/10;
    →     → weishu++;
    → } while (i!=0);
    → m=n*n;
    → for(i=0; i<weishu; i++)
    →     → m=m/10;
    → for(i=0; i<weishu; i++)
    →     → m=m*10;
    → if(n==n*n-m) sum+=n;
}
```


筛法求素数

例：使用筛法求100以内的所有素数。

思路

1. 想象将100个数看作沙子和小石头子，让小石头子权称素数；让沙子当作非素数。弄一个筛子，只要将沙子筛走，剩下的就是素数了。
2. 非素数一定是 2、3、4 的倍数。
3. 使用数组，让下标就是100以内的数，让数组元素的值作为筛去与否的标志。比如筛去以后让元素值为1。

		0	0	1	0	1	0	...	1	1
0	1	2	3	4	5	6	7		99	100

方法的依据

- 1至100这些自然数可以分为三类：
 - **单位数**：仅有一个数1。
 - **素数**：是这样一个数，它大于1，且只有1 和它自身这样两个正因数。
 - **合数**：除了1和自身以外，还有其他正因数。

**1不是素数，除1以外的自然数，当然只有素数与合数。
筛法实际上是筛去合数，留下素数。**

筛法思路

- 第一块是一个计数型的循环语句，功能是将prime数组清零。
 $\text{prime}[c] = 0; \quad c = 2, 3, \dots, 100$
- 第二块是正因数d初始化为 $d = 2$
 - 所有d的倍数将会被筛掉
- 第三块是循环筛数。这里用了一个 do while 语句，属于一种直到型循环，其一般形式为：

```
do
{
    循环体语句块
}
while ( 表达式 )
```

代码

```
int main() {  
    int n, prime[10000];  
    scanf("%d", &n);  
    prime[1] = 0;  
    for (int i = 2; i <= n; i++)  
        prime[i] = 1;  
    for (int i = 2; i <= n; i++) {  
        if (prime[i] == 1)  
            for (int j = 2; j * i <= n; j++)  
                prime[i * j] = 0;  
    }  
    for (int i = 2; i <= n; i++)  
        if (prime[i])  
            printf("%d ", i);  
    return 0;  
}
```

素数模块

筛法求素数.cpp

“数值计算类” 题目

#493-5· 回文素数位和

【知识点】获取数位；素数；回文数；

【问题描述】

如果一个数从左边读和从右边读都是同一个数，就称为回文数，既是素数又是回文数的数，称为回文素数，例如 500 到 1000 之间的回文素数有 6 个：727、757、787、797、919、929，其中数位和最大的是 797（数位和为 $7+9+7=23$ ）。编程求 $[m, n]$ 区间数位和最大的回文素数。

【输入格式】

输入一行，两个正整数 m 和 n ，用空格隔开。

【输出格式】

输出一行，包含 2 个整数，依次是区间 $[m, n]$ 中数位和最大的回文素数及其数位和，2 个整数之间一个用空格隔开。如果给定的 $[m, n]$ 区间没有素数回文数，则输出 0 0。

【输入样例 1】 【输入样例 2】

500 1000 20 50

【输出样例 1】 【输出样例 2】

797 23 0 0

“数值计算类” 题目

#287. 拆分数字↵

给出一个不多于 4 位的十进制非负整数 N，求它是几位数，并按个十百千顺序打印出各位数字。↵

输入格式↵

□□一行，只包含一个十进制非负整数 N。↵

输出格式↵

□□一行，分为两部分，首先按个十百千输出各位数字；然后输出位数。数字之间以逗号分隔。↵

输入样例↵

123↵

输出样例↵

3,2,1,3↵

“数值计算类” 题目

#304 整数计数

编写一个程序计算整数区间 $[a, b]$ 内，其个位数是 n ，且能被 k 整除的 m 位正整数共有多少个。

【输入格式】

输入只有一行，输入 5 个整数 a 、 b 、 n 、 k 、 m ，空格分隔，其中： $1 \leq a \leq b \leq 1,000,000$ ，且 $0 \leq n, k, m \leq 9$ 。

【输出格式】

输出一行，为符合要求的整数个数。

【样例输入 1】

1019 1 2 2

【样例输出 1】

0

【样例输入 2】

1050 4 2 2

【样例输出 2】

4

#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： $0 \leq n \leq 10000$ ， $1 \leq \text{学号} \leq 1000000$ ， $0 \leq \text{成绩} \leq 100$ 。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↵

```
5↵
1001 2 89 78↵
2003 4 88 99 100 88↵
4004 3 72 80 66↵
1004 3 70 66 82↵
3001 1 100↵
```

输出样例↵

```
2003 93.75↵
1001 83.50↵
┌───────────┐
1004 72.66↵
4004 72.66↵
```



```
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++;     }
```

```
→ }
```

```

→ 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;
→     → }

```

```
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 (KemuNum >= 2) { count++ ; }
```

```
}
```

```
// 输出结果
```

```
if ( count == 0 )
```

```
    printf ("NO");
```

```
else {
```

```
    for (i = 0; i < count - 1; i++)
```

```
    for (j = 0; j < count - i - 1; j++)
```

```
        {                                }
```

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

```
        printf(  );
```

```
}
```

统计分析类
框架

排序模块

$a[0] \dots a[n-1]$ n 个数排序

```
for (i = 0; i < n - 1; i++) // i = 0, 1, 2, ..., n-2
    for (j = 0; j < n - 1 - i; j++) // j = 0, 1, 2, ..., n-2-i
        if (a[j] < a[j + 1])
            { tmp = a[j]; a[j] = a[j + 1]; a[j + 1] = tmp; }
```

例如: $a[0] \dots a[9]$ 10 个数排序

```
for (i = 0; i < 9; i++) // i = 0, 1, 2, 3, ..., 8
    for (j = 0; j < 9 - i; j++) // j = 0, 1, 2, ..., 8-i
        if (a[j] < a[j + 1])
            { tmp = a[j]; a[j] = a[j + 1]; a[j + 1] = tmp; }
```

为避免搞混，
大家可以
就记这一种

$a[1] \dots a[n]$ n 个数排序

```
for (i = 1; i < n; i++) // i = 1, 2, 3, ..., n-1
    for (j = 0; j < n - i; j++) // j = 0, 1, 2, ..., n-1-i
        if (a[j] < a[j + 1])
            { tmp = a[j]; a[j] = a[j + 1]; a[j + 1] = tmp; }
```

例如: $a[1] \dots a[10]$ 10 个数排序

```
for (i = 1; i < 10; i++) // i = 1, 2, 3, ..., 9
    for (j = 0; j < 10 - i; j++) // j = 0, 1, 2, ..., 9-i
        if (a[j] < a[j + 1])
            { tmp = a[j]; a[j] = a[j + 1]; a[j + 1] = tmp; }
```

**j 的下标一定
保证从是0开始**

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]))**

#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 $A = [a_1, a_2, \dots, a_8]$, $B = [b_1, b_2, \dots, b_8]$, 则

【样例输出】

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

1012345678

例如，若 $A = [10, 1, 1, 1, 1, 1, 1, 1]$ ， $B = [1, 1, 1, 1, 1, 1, 1, 1]$ ，则 $\text{Similarity}(A, B) = (10*1 + 1*1 + \dots + 1*1) / (\text{sqrt}(10^2 + 1^2 + \dots + 1^2) + \text{sqrt}(1^2 + 1^2 + \dots + 1^2)) = 1.29$

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 ，($1 \leq n \leq 100, 1 \leq k \leq n$)，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。

【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 1 8 8 8

【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。

```
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(j = 1; j <= num; j++){
    →     → 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(j = 1; j <= top_k; j++){
    →     → printf("%d", man_id[j]);
    → }
    → return 0;
```

```
}.
```

→ 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

→ **}**

相似性模块

向量的内积（点乘）

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

a和b的点积公式为：

$$a \bullet b = a_1 b_1 + a_2 b_2 + \dots + a_n b_n$$

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

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

$$\begin{aligned} d_{Euclidean}(x, y) &= d_{Euclidean}(y, x) = \sqrt{|x_1 - y_1|^2 + |x_2 - y_2|^2 + \dots + |x_n - y_n|^2} \\ &= \sqrt{\sum_{i=1}^n |x_i - y_i|^2} \end{aligned}$$

```

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

一个音乐搜索引擎，用户输入一首乐曲 Q、一个正整数 k。从音乐库中查找与 Q 最相似的 k 首乐曲。乐曲由一组正整数组成，每个正整数表示声音的频率。计算乐曲相似度的方法是**直方图方法**。下面是详细的计算过程。↵

第一步，将整个频率的取值区域 $[0, 255]$ 均分为 16 个子区域，即 $[0, 15]$ 、 $[16, 31]$ 、.....、 $[240, 255]$ 。扫描一首乐曲，计算组成该乐曲的所有频率值出现在每个子区域的次数。例如一首乐曲 M1 由 4 个频率组成 10、12、245、245，则它的直方图就是↵

$[0, 15]: 2$ ↵

$[16, 31]: 0$ ↵

.....↵

$[240, 255]: 2$ ↵

即，该乐曲的频率在 $[0, 15]$ 和 $[240, 255]$ 这两个子区域各出现 2 次，其他子区域均出现 0 次。↵

第二步，分别得到两个乐曲 M1 和 M2 的直方图之后，将两个直方图看作两个向量，采用欧几里得公式计算两个直方图的距离。↵

欧几里得公式：给定两个向量 $v1 = [x1, x2, ..., xn]$ ，以及 $v2 = [y1, y2, ..., yn]$ ，它们的欧几里得距离为 $\sqrt{(x1-y1)^2 + (x2-y2)^2 + + (xn-yn)^2}$ ↵

例如，若另一首乐曲 M2 的直方图为↵

$[0, 15]: 2$ ↵

$[16, 31]: 0$ ↵

.....↵

$[240, 255]: 0$ ↵

则 M1 和 M2 的欧几里得距离为 $\sqrt{(2-2)^2 + 0 + ... + (0-2)^2} = 2$ ↵

#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 n_0 ($1 \leq n_0 \leq 100$)，表示乐曲长度，后面有 n_0 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， $1 \leq n \leq 100$ ，查找最相似的 k ($1 \leq k \leq n$) 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 n_i ($1 \leq n_i \leq 100$)，表示该乐曲长度，后面 n_i 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

```

int main(){
    ... int n,k,n0,i,j,count0[16]={0},m;
    ... int a[100],b[100][3],d[100][100],count[100][16]={0}; //a 存曲子.
        float sim[100]; //c[i]存相似度.
    ...

```

```

    ... scanf("%d",&n0); //曲长.
    ... for(i=0;i<n0;i++){
        ... scanf("%d",&a[i]);
        scanf("%d %d",&n,&k); //n 首曲子找 k 个最相近.
    ...

```

```

    ... for(i=0;i<n;i++){
        ... b[i][2]=i; //编号.
    ...

```

```

    ... for(i=0;i<n;i++){
        ... scanf("%d",&b[i][1]);
        ... for(j=0;j<b[i][1];j++){
            ... scanf("%d",&d[i][j]);
        ... }
    ...

```

```

....for(i=0;i<16;i++)..
.....for(j=0;j<n0;j++){..
.....if(a[j]>=0+i*16&& a[j]<=15+i*16)..
.....count0[i]++;//识别曲每个区间里的频率,
.....}..

```

..

```

....for(i=0;i<n;i++){..
.....for(j=0;j<16;j++){..
.....for(m=0;m<b[i][1];m++){..
.....if(d[i][m]>=0+j*16&& d[i][m]<=15+16*j)..
.....count[i][j]++;//每首曲子每个区间的频率,
.....}..

```

//上面的可以简化换成如下的,

```

....//for(i=0;i<n;i++){..
....//for(m=0;m<b[i][1];m++){..
....//.....j=d[i][m]%16..
....//.....count[i][j]++;//每首曲子每个区间的频率,
....//.....}..

```

```

    ... int sum;
    ... for(i=0;i<n;i++){
    ...     sum=0;
    ...     for(j=0;j<16;j++){
    ...         sum=sum+(count0[j]-count[i][j))*(count0[j]-count[i][j]);
    ...     sim[i]=sqrt(sum);//第i首曲子的相似度
    ... }

```

```

    ... float max; int t;
    ... for(i=0;i<n-1;i++){
    ...     for(j=0;j<n-1-i;j++){
    ...         if(sim[j]>sim[j+1]||((abs(sim[j+1]-sim[j])<1e-6)&&b[j][2]>b[j+1][2])){
    ...             max=sim[j]; sim[j]=sim[j+1]; sim[j+1]=max;
    ...             t=b[j][2]; b[j][2]=b[j+1][2]; b[j+1][2]=t;
    ...         }
    ...     }

```

```

    ... for(i=0;i<k;i++){
    ...     printf("%d ",b[i][2]);//要有空格
    ... return 0;
}

```

#91 相似乐曲.cpp

“统计分析类” 题目

#90·刷题高手（数组、双排序）+++

老师给了一个题目的列表，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的数量，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($1 \leq n \leq 100$)。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 ($1 \leq m \leq 100$)，以及老师希望找出刷题最多的 k 个同学 ($1 \leq k \leq m$)。

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， $0 \leq p \leq 100$ ；

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


```
#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++;
}

```

“统计分析类” 题目

#484. 重复元素【排序、数组】↵

【知识点】排序、数组↵

【问题描述】↵

统计数列有重复的元素及其个数。已知有不超过短整型数值范围的 n 个数 ($n \leq 1000$)，请查找统计有重复的数列项，按行从小到大输出有重复的项及其重复次数。↵

【输入格式】↵

两行，第一行表示数列元素的个数 n ；第二行为该数列的元素，以空格分隔。↵

【输出格式】↵

若干行，每行两个数，冒号分隔，分别为有重复的数项值及其重复的次数。输出顺序由数列项的值确定。若数列中没有重复项，则输出 NO。↵

【样例输入 1】↵

10↵
36 30 68 38 2 30 36 30 68 30↵

【样例输出 1】↵

30:4↵
36:2↵
68:2↵

“统计分析类” 题目

#308·猪场分配 (多重循环、排序、难) ↵

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。↵

枚举

```
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
```

```
}

```

```

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];
            }
        }
    }
}

```

#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 ($1 \leq m \leq 2^{30}$)，表示待查找的整数。

第三行包含 n 个正整数 ($\leq 2^{30}$)，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 $0 \sim n-1$ 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

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, q=0, times=0;
    → long int a[5000], m, t;
    → scanf("%d%d", &n, &m);
    → for(i = 1; i <= n; i++)
    →     → scanf("%d", &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 { q = 1; → break; }
    →     → mid = (i + j) / 2; }
    → if(q == 1) → printf("%d\n%d", mid - 1, times);
    → else → printf("-1\n%d", times);
    → return 0;
}

```

```

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 { q = 1; → break; }
    → mid = (i + j) / 2; }

```


二分查找

If searching for 23 in the 10-element array:

2	5	8	12	16	23	38	56	72	91
---	---	---	----	----	----	----	----	----	----

23 > 16,
take 2nd half

L					H				
2	5	8	12	16	23	38	56	72	91

23 < 56,
take 1st half

L						H			
2	5	8	12	16	23	38	56	72	91

Found 23,
Return 5

L						H			
2	5	8	12	16	23	38	56	72	91

二分查找

```
int L = 0, R = n-1;
int cmp = 0, 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;
    }
}
```

#110 回文判断

■ 回文

- Able was I ere I saw Elba
- 落败孤岛孤败落

■ 写一个程序，让用户任意输入一个字符串，判断是否是回文

■ 分析思路

- 判断位置 i 的字符与 $n-1-i$ 的字符是否相等
- 循环开始和终止的边界？

#110 回文判断

```
int main()
{
    int n, j, i;
    char a[1000];
    gets(a);
    n = strlen(a);
    for (i = 0, j = n - 1; i < (n + 1) / 2; i++, j--)
        if (a[i] != a[j])
            { printf("No"); break; }
    if (i == (n + 1) / 2) printf("Yes");
    return 0;
}
```

#110 回文判断.cpp

“对称循环类” 题目

[209]→#461·回文数等式↵

回文数 x 是指正读和反读都一样的正整数，如 11，121，1221。↵

输入一个 k ($1 \leq k \leq 1000$)，打印所有不超过 k 的数 i ($1 \leq i \leq k$) 的平方是回文数 x 的等式，即 $i*i=x$ 。输出每行一个等式。↵

输入格式↵

一行，只有一个整数 k ↵

输出格式↵

$i*i=x$ ↵

输入样例↵

30↵

输出样例↵

1*1=1↵

2*2=4↵

3*3=9↵

11*11=121↵

22*22=484↵

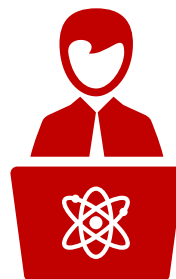
26*26=676↵

```
int int_rev(int x) {
    int y = 0;
    while (x) {
        y = y * 10 + x % 10; //反转的整数扩大10倍+原数的余数
        x /= 10;
    }
    return y;
}

int main() {
    int k;
    scanf("%d", &k);
    for (int i = 1; i <= k; i++)
        if (i * i == int_rev(i * i))
            printf("%d*%d=%d\n", i, i, i * i);
    return 0;
}
```



中國人民大學
RENMIN UNIVERSITY OF CHINA



谢谢大家!

