

浙江师范大学数理与信息工程学院

ACM/ICPC Team

算
法
设
计
入
门
教
材

2006 年 9 月

算法设计讲稿

第一章 算法初步

第一节 程序设计与算法

一、算法

算法是解决问题方法的精确描述，但是并不是所有问题都有算法，有些问题经研究可行，则相应就有算法，但这并不是说问题就有结果。上述的“可行”，是指对算法的研究。

1. 待解问题的描述

待解问题表述应精确、简练、清楚，使用形式化模型刻画问题是最恰当的。例如，使用数学模型刻画问题是最简明、严格的，一旦问题形式化了，就可依据相应严格的模型对问题求解。

2. 算法设计

算法设计的任务是对各类具体问题设计良好的算法及研究设计算法的规律和方法。常用的算法有：穷举搜索法、递归法、回溯法、贪心法、分治法等。

3. 算法分析

算法分析的任务是对设计出的每一个具体的算法，利用数学工具，讨论各种复杂度，以探讨某种具体算法适用于哪类问题，或某类问题宜采用哪种算法。

算法的复杂度分时间复杂度和空间复杂度。

- 时间复杂度：在运行算法时所耗费的时间为 $f(n)$ (即 n 的函数)。

- 空间复杂度：实现算法所占用的空间为 $g(n)$ (也为 n 的函数)。

称 $O(f(n))$ 和 $O(g(n))$ 为该算法的复杂度。

二、程序设计

1. 程序

程序是对所要解决的问题的各个对象和处理规则的描述，或者说是数据结构和算法的描述，因此有人说，数据结构+算法=程序。

2. 程序设计

程序设计就是设计、编制和调试程序的过程。

3. 结构化程序设计

结构化程序设计是利用逐步求精的方法，按一套程式化的设计准则进行程序的设计。由这种方法产生的程序是结构良好的。所谓“结构良好”是指：

- (1) 易于保证和验证其正确性；

- (2) 易于阅读、易于理解和易于维护。

按照这种方法或准则设计出来的程序称为结构化的程序。

“逐步求精”是对一个复杂问题，不是一步就编成一个可执行的程序，而是分步进行。

- 第一步编出的程序最为抽象；

- 第二步编出的程序是把第一步所编的程序（如过程、函数等）细化，较为抽象；

-

- 第 i 步编出的程序比第 $i-1$ 步抽象级要低；

.

. 直到最后, 第 n 步编出的程序即为可执行的程序。

所谓“抽象程序”是指程序所描述的解决问题的处理规则, 是由那些“做什么”操作组成, 而不涉及这些操作“怎样做”以及解决问题的对象具有什么结构, 不涉及构造的每个局部细节。

这一方法原理就是: 对一个问题(或任务), 程序人员应立足于全局, 考虑如何解决这一问题的总体关系, 而不涉及每局部细节。在确保全局的正确性之后, 再分别对每一局部进行考虑, 每个局部又将是一个问题或任务, 因而这一方法是自顶而下的, 同时也是逐步求精的。采用逐步求精的优点是:

(1) 便于构造程序。由这种方法产生的程序, 其结构清晰、易读、易写、易理解、易调试、易维护;

(2) 适用于大任务、多人员设计, 也便于软件管理。

逐步求精方法有多种具体做法, 例如流程图方法、基于过程或函数的方法。

[例]求两自然数, 其和是 667, 最小公倍数与最大公约数之比是 120:1(例如 (115, 552)、(232, 435))。

[解]两个自然数分别为 m 和 $667-m$ ($2 \leq m \leq 333$)。

处理对象: m (自然数)、 l (两数的最小公倍数)、 g (两数的最大公约数)。

处理步骤: 对 m 从 2 到 333 检查 l 与 g 的商为 120, 且余数为 0 时, 打印 m 与 $667-m$ 。

第一层抽象程序:

```
Program TwoNum;
Var   m, l, g: integer;
Begin for m:=2 to 333 do
    begin l:=lcm(m, 667-m);      {求最小公倍数}
          g:=gcd(m, 667-m);      {求最大公约数}
          if (l=g*120) and (l mod g=0) then
              writeln(m:5, 667-m:5);
          end;
End.
```

第二层考虑函数 lcm(最小公倍数)、gcd(最大公约数)的细化。

最大公约数问题是对参数 a 、 b , 找到一个数 i 能整除 a 与 b , i 就是 gcd 的函数值。

```
Function gcd(a, b: integer): integer;
var   i: integer;
begin for i:=a downto 1 do
    if not((a mod i=0) or (b mod i=0)) then gcd:=i;
end;
```

而最小公倍数的计算是: 若干个 b 之和, 若能被 a 整除, 则该和便是 a 、 b 的最小公倍数。

```
Function lcm(a, b: integer): integer;
var   i: integer;
begin i:=b;
    while i mod a=0 do i:=i+b;
    lcm:=i;
end;
```

第二节 编程入门题例

编程入门题（一）

1、位数对调：输入一个三位自然数，把这个数的百位与个位数对调，输出对调后的数。例如：Input 3 bit nature data:234

n=432

[解]1. 先确定输入数 n 是否三位数，即 $n > 99$ 且 $n \leq 999$ 。

2. 位数对调： $n=abc \rightarrow cba=x$

①百位数 $a=n$ 整除 100；②十位数 $b=(n-a*100)$ 整除 10；③个位数 $c=n$ 除以 10 的余数；

3. 得对调后的数： $x=c*100+b*10+a$

[程序]

```
{I-} {输入的数据为整数}
program Threebit;
var x,n,a,b,c:INTEGER;
BEGIN write('Input 3 bit nature data:'); readln(n);
      IF (n>99) and (n<1000) then
        begin a:=n DIV 100;    {求百位数}
              b:=(n-a*100) DIV 10; {求十位数}
              c:=n mod 10;      {求个位数}
              x:=c*100+b*10+a;   {得新数 X}
              writeln('Number=',x:3);
        end
      ELSE writeln('Input error!');
END.
```

2、求三角形面积：给出三角形的三个边长为 a, b, c, 求三角形的面积。

提示：根据海伦公式来计算三角形的面积：

$$S = \frac{a+b+c}{2}; \text{Area} = \sqrt{S(S-a)(S-b)(S-c)}$$

[解]1. 输入的三角形三边长 a, b, c 要满足“任意两边长的和大于第三边长”。

2. 按海伦公式计算： $s = (a+b+c)/2$; $x = s*(s-a)*(s-b)*(s-c)$

这时若 $x \geq 0$, 则求面积： $\text{area} = \sqrt{x}$, 并输出 area 的值。

[程序]

```
PROGRAM h1;
VAR a,b,c,s,x,area:real;
BEGIN
  write('Input a,b,c:'); readln(a,b,c);
  If (a>0) and (b>0) and (c>0) and (a+b>c) and (a+c>b) and (b+c>a) Then
    Begin s:=(a+b+c)/2; x:=s*(s-a)*(s-b)*(s-c);
      If x>=0 Then Begin Area:=SQRT(x);writeln('Area=',area:8:5); End;
    End
  Else writeln('Input error!')
END.
```

3、模拟计算器：试编写一个根据用户键入的两个操作数和一个运算符，由计算机输出运算结果的程序。这里只考虑加(+)、减(-)、乘(*)、除(/)四种运算。

例1：Input x,y: 15 3

Input operator(+,-,*,/): +

15.00+ 3.00= 18.00

例2：Input x,y: 5 0

Input operator(+,-,*,/): /

divide is zero!

[解]该题关键是判断输入的两数是作何种运算(由输入的运算符 operator 决定，如 '+'、'-'、'*'、'/' 分别表示加、减、乘、除法的运算)。其中要进行除(/)运算时，要先进行合法性检查，即除数不能为 0。

[程序]

```

PROGRAM Oper;
Var x,y,n:real;
    operator:char;
Begin
  write(' Input x,y:');readln(x,y);
  write(' Input operator:');readln(operator);
  Case operator of
    '+' :n:=x+y;      {加法运算}
    '-' :n:=x-y;      {减法运算}
    '*' :n:=x*y;      {乘法运算}
    '/' :If y=0 then {除法运算}
      begin writeln(' Divide is zero!');halt;end
    Else n:=x/y;
    else begin writeln(' Input operator error!');halt;end;
  End;
  writeln(x:6:2,operator,y:6:2,'=',n:6:2);
End.

```

4、念数字：编一个“念数字”的程序，它能让计算机完成以下工作：当你输入一个 0 至 99 之间的数后，计算机就会用汉字拼音印出这个数的念结束。

例 1：Input data:35

SAN SHI WU

例 2：Input data:0

LING

如果输入的数不在 0 到 99 之间，就印出“CUO LE”（错了），请求重新输入。

注：为了使不熟悉汉语拼音的同学也能做这个题，把“零，一，二，三，……，九，十”的拼音法写在下面。

零 LING 一 YI 二 ER 三 SAN 四 SHI 五 WU
 六 LIU 七 QI 八 BA 九 JIU 十 SHI

[解]输入数在 0~99 之间，若 x 为两位数则拆分为十位数、个位数。然后调用念数过程 ReadDigit 用汉字拼音印出各位数（0~9）的念。

[程序]

```

{$I-}
Program NinShu;
Var x,a,b:Integer;
Procedure ReadDigit(n:Integer);{念数过程:n=0~9}
Begin
  Case n of
    0:write(' LING ');
    1:write(' YI ');
    2:write(' ER ');
    3:write(' SAN ');
    4:write(' SHI ');
    5:write(' WU ');
    6:write(' LIU ');
    7:write(' QI ');
    8:write(' BA ');
    9:write(' JIU ');
  End;
End; {ReadDigit}

Begin {main}
  Repeat write(' Input data:');readln(x);
    if (x<0) or (x>99) then writeln('Cuo Le');
  Until (x>=0)and(x<=99);
  If (x>=0)and(x<=9) then ReadDigit(x) {调用念数过程}
  Else Begin a:=x DIV 10; b:=x mod 10; {位数拆分}
    If a<>1 then ReadDigit(b);
    writeln(' Shi');
  End;
End.

```

```

    if b<>0 then ReadDigit(b);
  End;
  writeln;
End.

```

5、数列找数：数组 A(N) 的各下标变量中 N 个互不相等的数，键盘输入正整数 M ($M \leq N$)，要求打印数组中第 M 大的下标变量的值。

例如：数组 A (10) 的数据为：

A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)	A(9)	A(10)
16	57	20	19	38	41	6	13	25	32

运行结果：INPUT AN NUMBER: 3

A(5)=38 (即第 3 大的数是 A(5)=38)

[解]该题要从 N 个互不相等的数中找第 M 大的值。有以下两种解法：

解法一：初始时：A 数组存放 N 个互不相等的数；B 数组用于存放数组 A 的下标。见下表一。

下标值 i	1	2	3	4	5	6	7	8	9	10
数组 A	16	57	20	19	38	41	6	13	25	32
数组 B	1	2	3	4	5	6	7	8	9	10



降序处理（冒泡排序法）：

数组 A 的元素由大到小进行排序，同时数组 B 的元素排列也随之变化。

下标值 I	1	2	3	4	5	6	7	8	9	10
数组 A	57	41	38	32	25	20	19	16	13	6
数组 B (原数组 A 的下标)	2	6	5	10	9	3	4	1	8	7

例题中 M=3，由表二知 A[3]=38，B[3]=B[M]=5（原数组 A 的下标值）即为所求。

[程序]

```

Program Max01; {冒泡排序法}
var i, j, n, m, x: integer;
    A, B: ARRAY[1..100] of integer;

Procedure Init; {读数初始化过程}
Var i, j: integer; fd: boolean;
Begin
  write(' Input N: '); readln(N); {读入 N}
  if N<1 then begin writeln(' Input error! '); halt; end;
  write(' Input ', N:3, ' Data: ');
  For i:=1 to n Do
    begin read(A[i]); B[i]:=i; end; {读入 A[i], 且 B[i] 初值置为 i}
  Readln;
  i:=1; fd:=false; {数组中的数有相同的标志}
  while NOT fd and (i<=N-1) do
    Begin j:=i+1;
      While NOT fd and (j<=N) do
        begin fd:=A[i]=A[j]; {若 A[i]=A[j], 则 fd=true; 否则 fd=false}
          j:=j+1;
        end;
      i:=i+1;
    End;
  If fd then {fd 为 True, 则表示数组中至少有两个相等的数}
    begin writeln(' More equal number! '); halt; end;
  write(' Input M: '); readln(M);
  If M>N then {表明所找的数 M 已超过输入数的总数 N}
    begin writeln(' Input error! '); halt; end;
End; {Init}

Begin {MAIN}

```

```

Init; { 读数过程 }
for i:=1 to N-1 do { 冒泡排序 }
  for j:=i+1 to N do
    if A[i]<A[j] then
      begin x:=A[i];A[i]:=A[j];A[j]:=x; { 交换 A[i] 与 A[j] 的值 }
            x:=B[i];B[i]:=B[j];B[j]:=x; { 交换 B[i] 与 B[j] 的值 }
      end;
  writeln('A(',B[M],')=',A[M]); { 输出第 M 大的数、原下标值 }
End.

```

解法二(搜索法):用 $B[i]$ 表示在 $A[1]$ 、 $A[2]$ 、 $A[3]$ 、……、 $A[N]$ 中比 $A[i]$ 大的个数 ($i=1, 2, 3, \dots, N$)。

搜索的结果见下表三:

下标值 i	1	2	3	4	5	6	7	8	9	10
A 数组	16	57	20	19	38	41	6	13	25	32
B 数组	8	1	6	7	3	2	10	9	5	4

例题中 $M=3$, 由表三知 $B[5]=3=M$, $A[5]=38$ 即为所求。

[程序]

```

Var i, j, k, m, n: integer;
    A, B: ARRAY[1..100] of integer;

Procedure Init;
{ 具体程序见解法一 }
Begin { MAIN }
  Init; { 读数过程 }
  for i:=1 to n do
    begin B[i]:=1; { B[i] 初始值为 1, 即假定所有 A[i] 都可能为最大数 }
      for j:=1 to n do
        if (i<>j) and (A[i]<A[j]) then B[i]:=B[i]+1;
        if B[i]=M then k:=i;
      end;
    writeln('A(',k,')=',A[k]);
  End.

```

在上述表三中, 我们可以发现: 例中 $M=3$, 在搜索过程中当下标 $i=5$ 时, $B[5]=M=3$, 此时已找到所求, 即 $A[5]=38$ 。这样 $i>5$ 时就不用再搜索下去。因此, 可对解法二的算法优化如下:

1. 读数至 A 数组
2. $i:=1$; $fd:=false$; { fd : 判断是否已找到所求数的标志 }
3. 当 $(fd=false) \text{ and } (i \leq n)$ 时, 做
 - (1) $B[i]:=1$ { 表示数列中比 $A[i]$ 小的数的总个数+1, 初始值为 1 }
 - (2) $j:=1$
 - (3) 当 $j \leq n$ 时, 做
 - ① 如果 $(i < j) \text{ and } (A[i] < A[j])$, 则 $B[i]$ 的值加 1。
 - ② $j:=j+1$
 - (4) 如果 $B[i]=M$, 则输出所求: $A[i]$, 且 $fd:=true$ 。
 - (5) $i:=i+1$
4. 程序结束。

[主程序]

```

Begin { MAIN }
  Init; { 读数过程 }
  i:=1; fd:=false; { 找到所求解的标志值 }
  while not fd and (i<=N) do
    begin B[i]:=1; { B[i] 初始值为 1, 即假定所有的 A[i] 都可能为最大的数 }
      j:=1;
      while j<=n do
        begin
          if (i<>j) and (A[i]<A[j]) then B[i]:=B[i]+1;
          j:=j+1;
        end;
    end;
  End.

```

```

end; {while j}
if B[i]=M then {找到所求便输出，并退出比较}
begin writeln('A(',i,')=',A[i]);fd:=true; end;
i:=i+1;
end;{while i}
End.

```

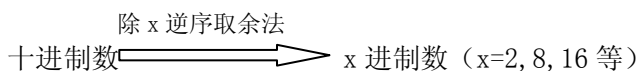
6、数制转换：编程输入十进制N（N：-32767~32767），请输出它对应的二进制、八进制、十六进制数。例如：

```

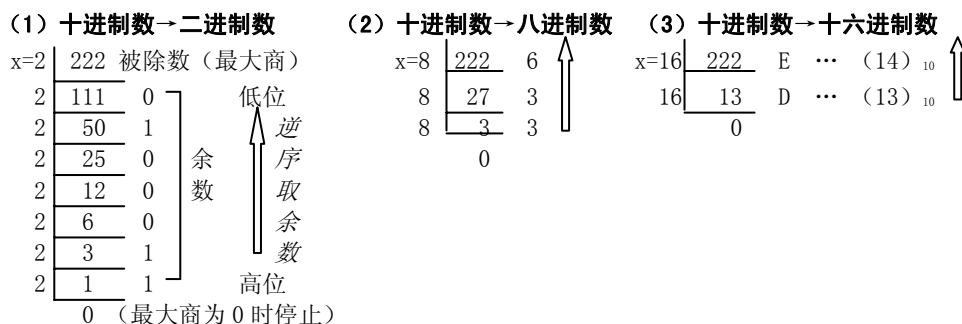
INPUT N(-32767~32767):222
222 TURN INTO 2: 11011110
222 TURN INTO 8: 336
222 TURN INTO 16: DE

```

【解】十进制数转化为某进制数的转换方法，如下图示：



例中 n=222（十进制数），转换成 x 进制数的过程如下图示：



将每次所得的余数由下至上排列(逆序取余数)，即有：

(222)₁₀ 转换成二进制数得到：1100010

(222)₁₀ 转换成八进制数得到：336

(222)₁₀ 转换成十六进制数得到：13、14

这时得到的逆序余数串（在数组 B[1]、B[2]、……、B[k]中）的每位数均为十进制数。程序中利用字符串 A 来计算 x 进制数的位数（即 COPY (A, B[i]+1, 1)），见下表：

数组 B	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
字符串 A	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
下标 i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

由上表得：(222)₁₀ = (1100010)₂ = (336)₈ = (DE)₁₆

【程序】

```

{$I-}
Program jj;
var N,k:integer;
    B:array[1..1000] of integer;

Procedure chg10(N,x:integer); {将十进制数 N 转换成 x 进制数}
Begin k:=0; {转换后的 x 进制数位的长度}
  while N<>0 do {N 不为 0 时}
  begin B[k]:=N mod x; {除以 x 取余数}
      N:=N Div x; {取最大商 N}
      k:=k+1; {x 进制数位数加 1}
  end;
end; {chg10}

```



```

Procedure Sput(N,x:integer); {进制数输出}
VAR i:integer; A:string;
Begin A:='0123456789ABCDEF'; {表示 x 进制数的位数串}
  write(N:6, ' Turn into ', x:2, ':');
  for i:=k-1 downto 0 do write(copy(A,B[i]+1,1)); {逆序输出 x 进制数}
  writeln;
end; {Sput}
begin {MAIN}
  write(' Input N(-32767 to 32767):');readln(N);
  if (N<=-32767)or(N>32767) then
    begin writeln(' Input error!');halt;end;
  chg10(N,2);Sput(N,2); {十进制数转换成 2 进制数, 并输出}
  chg10(N,8);Sput(N,8); {十进制数转换成 8 进制数, 并输出}
  chg10(N,16);Sput(N,16); {十进制数转换成 16 进制数, 并输出}
end. →

```

编程入门题(二)

1、**求素数**: 求 2 至 N ($2 \leq N \leq 500$) 之间的素数。例如:

输入: N=100

输出: 2 3 5 7 11 13
17 19 23 29 31 37
41 43 47 53 59 61
71 73 79 83 89 97

total=24 {表示 2 至 100 之间的素数有 24 个}

[解法一]素数是指除 1 及本身以外不能被其他数整除的自然数。下面介绍用穷举法求素数。

- 2 是素数; $t=0$;
- $I=2 \sim n$, 则:
 - 如果 i 是素数, 则其必须是奇数且不能被 $2 \sim \sqrt{i}$ 中的任一个数整除。
 - 如果 I 是素数, 则输出该素数且计数器 $t=t+1$;
- 输出 $2 \sim N$ 之间素数的总数: $total=t$;
- 程序结束

[程序]

```

program exa;
uses crt;
var i,k,n,w,t:integer;
    yes:boolean;
Begin t:=0;clrscr;write('N=');readln(n);
  if (n<2)and(n>500) then
    begin writeln('input error!');halt;end;
  write(2:5);t:=1;
  for i:=2 to n do
    if odd(i) then
      begin yes:=true;
        w:=trunc(sqrt(i));k:=2;
        while (k<=w)and yes do
          begin if i mod k=0 then yes:=false; k:=k+1;end;
          if yes then begin write(i:5);t:=t+1; end;
        end;
      writeln('total=',t); end.

```

[解法二]筛法求素数:

- 建立一个包括 2, 3, ..., N 的数据“集合”R;
- 顺序取 R 中的数 (从素数 2 开始), 同时将其及能整除它的数从 R 中划去。“筛法求素数”的过程如下图示:

素 数	数据集合 R
R[1]	
2	{ 2 , 3, 4 , 5, 6 , 7, 8 , 9, 10 , 11, 12 ,}

3	{ 3 , 5, 7, 9 , 11, }
5	{ 5 , 7, 11, }
.....

这样便可得到所求的素数：2, 3, 5,。

[程序]

```

program Sushu;
var i, j, k, n, w, t: integer;
    R, P: array [1..500] of integer;
begin
  write('N='); readln(n);
  if (n<2) or (n>500) then begin writeln('Input N error!'); halt end;
  for i:=2 to n do R[i-1]:=i; writeln('Result:'); t:=0; k:=n-1;
  while k>0 do
    begin P:=R; {数组 P 保存在 P 中}
      write(R[1]:5); {输出素数} t:=t+1; w:=R[1]; j:=0;
      for i:=1 to k do
        if P[i] mod w<>0 then {划去 w 及 W 的倍数}
          begin j:=j+1; R[j]:=P[i]; end; {重新构造集合 R}
      k:=j; {新建后 R 的元素个数}
      end; writeln; writeln('Total=', t);
  end.

```

2、矩阵相乘：已知 $N \times M_1$ 矩阵 A 和 $M_1 \times M$ 矩阵 B ($1 \leq M, M_1, N \leq 10$)，求矩阵 C ($=A \times B$)。例如：

输入：N, M1, M=4 3 4

A= 1 2 3

3 4 5

4 5 6

5 -1 -2

B= 1 6 4 2

2 3 4 1

-1 5 7 -3

输出：C= 2 27 33 -5

6 55 63 -5

8 69 78 -5

5 17 2 15

提示：所谓矩阵相乘（如 $A \times B = C$ ），是指

$$C_{ij} = \sum (A_{ik} \times B_{kj}) \quad (i=1 \sim N, j=1 \sim M, k=1 \sim M)$$

例如：

$$C_{11} = A_{11} \times B_{11} + A_{12} \times B_{21} + A_{13} \times B_{31}$$

$$= 1 \times 1 + 2 \times 2 + 3 \times (-1)$$

$$= 2$$

$$C_{42} = A_{41} \times B_{12} + A_{42} \times B_{22} + A_{43} \times B_{32}$$

$$= 5 \times 6 + (-1) \times 3 + (-2) \times 5$$

$$= 17$$

[解] 该题主要考查矩阵的表示及其运算。矩阵 A ($N \times M_1$) 与矩阵 B ($M_1 \times M$) 相乘得矩阵 C ($N \times M$)，其解法如下：

$\left[\begin{array}{l} i=1 \sim N, \text{ 重复做} \\ j=1 \sim M, \text{ 重复做} \end{array} \right. \quad (1) C_{ij}=0; \quad (2) k=1 \sim M_1, \text{ 重复做 } C_{ij}=C_{ij}+A_{ik} \times B_{kj}; \quad (3) \text{ 输出 } C_{ij}$

[程序]

```

program JiZheng;
var i, j, k, N, M1, M: integer;
    A, B, C: array [1..10, 1..10] of integer;
begin
  write('N, M1, M='); readln(N, M1, M);
  if (N>=1) and (N<=10) and (M1>=1) and (M1<=10) and (M>=1) and (M<=10) then
    begin {输入 N×M1 矩阵 A} write('A=');
      for i:=1 to N do for j:=1 to M1 do read(A[i, j]); readln;
      {输入 M1×M 矩阵 B} write('B=');
      for i:=1 to M1 do for j:=1 to M do read(B[i, j]); readln;
      write('C=');
      for i:=1 to N do
        begin for j:=1 to M do {计算 Cij=∑ (Aik×Bkj)}
          begin C[i, j]:=0; for k:=1 to M1 do C[i, j]:=C[i, j]+A[i, k]*B[k, j];
            write(C[i, j]:5); {输出矩阵 Cij}
          end;
        writeln; write(' ');
      end;
    end;

```

```

end;
writeln;
end
else writeln(' Input N,M1,M error!');
end.

```

3、找数字对：输入 N ($2 \leq N \leq 100$) 个数字 (在 0 与 9 之间)，然后统计出这组数中相邻两数字组成的链环数字对出现的次数。例如：

输入：N=20 {表示要输入数的数目}

0 1 5 9 8 7 2 2 2 3 2 7 8 7 8 7 9 6 5 9

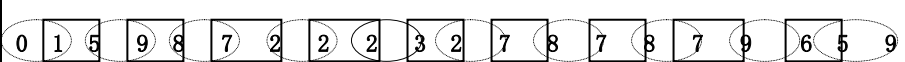
输出：(7, 8) =2 (8, 7) =3 {指 (7, 8)、(8, 7) 数字对出现次数分别为 2 次、3 次}

(7, 2) =1 (2, 7) =1

(2, 2) =2

(2, 3) =1 (3, 2) =1

【解】该题主要是数组的存储表示及运用。数组 A 存放所输入的 N 个数，用数组 B 表示 $A[i]$ 与 $A[i+1]$ 相邻数字对出现的次数。计算过程见下表：

数组 A																			
次数 i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
求链环 数字的 过程	I= 1: B[0, 1]=1				I=2: B[1, 5]=1				I=3: B[5, 9]=1				I=4: B[9, 8]=1						
	I= 5: B[8, 7]=1				I=6: B[7, 2]=1				I=7: B[2, 2]=1				I=8: B[2, 2]=2						
	I=9: B[3, 2]=1				I=10: B[2, 3]=1				I=11: B[2, 7]=1				I=12: B[7, 8]=1						
	I=13: B[8, 7]=2				I=14: B[7, 8]=2				I=15: B[8, 7]=3				I=16: B[7, 9]=1						
	I=17: B[9, 6]=1				I=18: B[6, 5]=1				I=19: B[5, 9]=1										

【程序】

```

Program Getdouble;
var i,N:integer; fd:boolean;
    A:array[1..100] of integer;
    B:array[0..9,0..9] of integer;
Begin fillchar(B,sizeof(B),0); {数组 B 初值置为 0} write(' N=');readln(N);
    if (N>1)and(N<=100) then
        begin write('Enter ',N,' numbers:');
            for i:=1 to N do
                begin read(A[i]); {输入 A[i]}
                    if (A[i]<0)or(A[i]>9) then {A[i]的值在 0 与 9 之间}
                        begin writeln(' Input error!');halt;end;
                    end;
            for i:=1 to N-1 do
                B[A[i],A[i+1]]:=B[A[i],A[i+1]]+1; {相邻数字对B[A[i],A[i+1]]值加 1}
            fd:=true;writeln(' Result:');
            for i:=1 to N-1 do
                if (B[A[i],A[i+1]]>0)and(B[A[i+1],A[i]]>0) then
                    begin write(' (',A[i],',',A[i+1],')=',B[A[i],A[i+1]],' ');
                        if A[i+1]=A[i] then writeln {相邻数字相同, 如 (2, 2)}
                        else writeln(' (',A[i+1],',',A[i],')=',B[A[i+1],A[i]]);
                        B[A[i+1],A[i]]:=0;
                        fd:=false;
                    end;
                if fd then writeln(' Not found!');
            end
        else writeln(' Input N error!');
    end.

```

4、蛇形矩阵：生成一个按蛇形方式排列自然数 1, 2, 3, 4, 5, ……., N^2 的 ($1 < N \leq 10$) 阶方阵。例如：

输入：N=4

N=7

输出：1 3 4 10
2 5 9 11
6 8 12 15

1 3 4 10 11 21 22
2 5 9 12 20 23 34
6 8 13 19 24 33 35

7 13 14 16

7 14 18 25 32 36 43
15 17 26 31 37 42 44
16 27 30 38 41 45 48
28 29 39 40 46 47 49

[解] 本题考查重点是矩阵的表示及上下标运算，旨在培养观察和分析思维能力。

例如，当 $N=7$ 时，把 1, 2, 3, …, 49 逐一填入数组 A 中，搜索填数的方向如右图示：

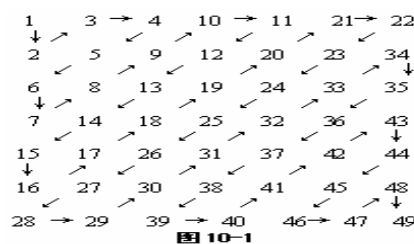
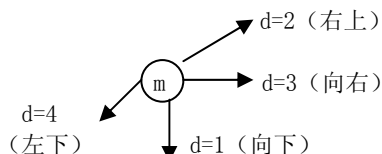


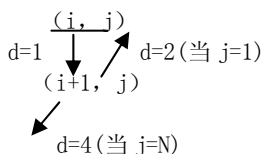
图 10-1

从图中，我们可以看出：对每个数字 m (1, 2, …, $\sim N \times N$) 来说，可能按以下四种搜索方向之一填数：

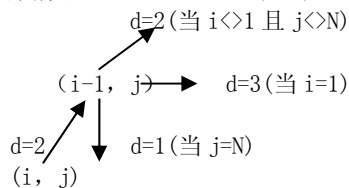


按照图 10-1 的搜索方向填数时，先要把当前数字 m 填入数组 $A[i, j]$ 中，接下来就是要确定下一次的填数的坐标及其搜索方向（即 $d=?$ ）。现分析如下：

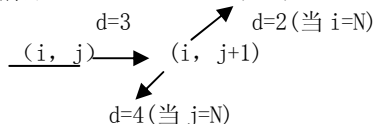
第一种情形 ($d=1$ ，如 $m=2, 7, 15; 35, 44$):



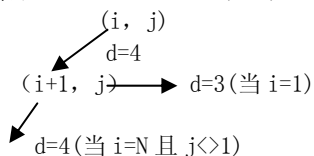
第二种情形 ($d=2$ ，如 $m=2, 10, 21$; 或 $34, 43, 48$; 或 $7, 8, 9, 16, 17, 18, 19$ 等):



第三种情形 ($d=3$ ，如 $m=29, 40, 47$; 或 $4, 11, 22$):



第四种情形 ($d=4$ ，如 $m=28, 39, 46$; 或 $6, 15$; 或 $5, 12, 13, 14$ 等):



[程序]

```
Program she;
const max=10;
var d,i,j,m,N:integer;
    A:array [1..10,1..10] of integer;
begin
    write('N=');readln(N);
    if (N>=1) and (N<=max) then
        begin i:=1;j:=1;m:=1;d:=1;
            repeat A[i,j]:=m; {填数}
            case d of
                1: {第一种情形}begin i:=i+1; if j=1 then d:=2 else d:=4; end;
                2: {第二种情形}begin i:=i-1;j:=j+1;
```

```

        if j=N then d:=1 else if i=1 then d:=3;end;
3: {第三种情形}begin j:=j+1; if i=N then d:=2 else d:=4;end;
4: {第三种情形}begin i:=i+1;j:=j-1;
        if i=N then d:=3 else if j=1 then d:=1;end;
end;
m:=m+1;
until m>N*N;
writeln(' OUTPUT:');
for i:=1 to N do begin for j:=1 to N do write(A[i,j]:4); {输出填数} writeln;end;
end
else writeln(' Input N error!');
end.

```

5、编码问题(95 年全国分区联赛题): 设有一个数组 A: array [0..N-1] of integer; 存放的元素为 $0 \sim N-1$ ($1 < N \leq 10$) 之间的整数, 且 $A[i] \neq A[j]$ ($i \neq j$)。例如当 $N=6$ 时, 有: $A = (4, 3, 0, 5, 1, 2)$ 。此时, 数组 A 的编码定义如下:

A[0]编码为 0;

A[i]编码为: 在 $A[0], A[1], \dots, A[i-1]$ 中比 $A[i]$ 的值小的个数
($i=1, 2, \dots, N-1$)

\therefore 上面数组 A 的编码为: $B = (0, 0, 0, 3, 1, 2)$

要求编程解决以下问题:

(1) 给出数组 A 后, 求出其编码;

(2) 给出数组 A 的编码后, 求出 A 中的原数据

程序样例:

例一:

输入: Stat=1 {表示要解决的第 (1) 问题}

N=8 {输入 8 个数}

A=1 0 3 2 5 6 7 4

输出: B=0 0 2 2 4 5 6 4

例二:

输入: Stat=2 {表示要解决的第 (2) 问题}

N=7

B=0 1 0 0 4 5 6

输出: A=2 3 1 0 4 5 6

[解] 第 1 个问题的解法: 用穷举搜索法。

$B[0]$ 为 0

$B[i]$ 表示在 $A[1], A[2], \dots, A[i-1]$ 中比 $A[i]$ ($i=1, 2, \dots, N$) 小的个数。

第 2 个问题的解法: 先构建数组 P, 初始值为 $0, 1, 2, 3, \dots, N-1$ 。然后从 $B[N-1], B[N-2], \dots, B[1], B[0]$ 逆向从数组 P 中取数求数组 A。以题中例二为例, 求解过程如下图:

下标值 i	$B_0 B_1 B_2 B_3 B_4 B_5 B_6$	$p_0 p_1 p_2 p_3 p_4 p_5 p_6$	数组 A
6	0 1 0 0 4 5 6 ↑	0 1 2 3 4 5 (6)	$A[6]=p[B_6]=p_6=6$, 划去 p_6
5	0 1 0 0 4 5 6 ↑	0 1 2 3 4 (5)	$A[5]=p[B_5]=p_5=5$, 划去 p_5
4	0 1 0 0 4 5 6 ↑	0 1 2 3 (4)	$A[4]=p[B_4]=p_4=4$, 划去 p_4
3	0 1 0 0 4 5 6 ↑	(0) 1 2 3	$A[3]=p[B_3]=p_0=0$, 划去 p_0
2	0 1 0 0 4 5 6 ↑	(1) 2 3	$A[2]=p[B_2]=p_1=1$, 划去 p_1
1	0 1 0 0 4 5 6 ↑	2 (3)	$A[1]=p[B_1]=p_3=3$, 划去 p_3
0	0 1 0 0 4 5 6 ↑	(2)	$A[0]=p[B_0]=p_2=2$

从上述求解过程中，我们得到：A= (2, 3, 1, 0, 4, 5, 6)。

[程序]

```
program code;
var i, j, k, m, n, stat: integer;
    A, B, P: array [0..10] of integer;
begin
    write('Stat(1, or 2)='); readln(stat);
    case stat of
        1: begin write('N='); readln(n); write('A=');
            for i:=0 to n-1 do read(A[i]); readln; {读入数组 A}
            for i:=0 to n-2 do
                for j:=i+1 to n-1 do
                    if (A[i]=A[j]) or (A[i]>n-1) then {数组 A 中有否相等}
                        begin writeln('Input error!'); halt; end;
            for i:=0 to n-1 do B[i]:=0; {编码数组 B 初始值为 0}
            for i:=1 to n-1 do
                for j:=0 to i-1 do
                    if A[i]>A[j] then B[i]:=B[i]+1; {求数组编码}
            for i:=0 to n-1 do write(B[i]:5); writeln;
            end;
        2: begin write('N='); readln(n); write('B=');
            for i:=0 to n-1 do read(B[i]); readln; {读编码数组 B}
            for i:=0 to n-1 do P[i]:=i; {建立取数数列 P}
            for i:=n-1 downto 0 do {由编码数组 B 逆向求原数组 A}
                begin A[i]:=P[B[i]]; {A[i] 为数组 P 中第 B[i] 号元素}
                for j:=B[i] to i-1 do P[j]:=P[j+1]; {从 P 数组中删去 P[B[i]]}
            end;
            write('A=');
            for i:=0 to n-1 do write(A[i]:5); writeln; {输出数组 A}
        end;
    end;
end.
```

第二章 算法应用

一、穷举搜索法

穷举搜索法是穷举所有可能情形，并从中找出符合要求的解。

穷举所有可能情形，最直观的是联系循环的算法。

[例] 找出 n 个自然数 $(1, 2, 3, \dots, n)$ 中 r 个数的组合。例如，当 $n=5, r=3$ 时，所有组合为：

5	4	3
5	4	2
5	4	1
5	3	2
5	3	1
5	2	1
4	3	2
4	3	1
4	2	1
3	2	1
total=10 {组合的总数}		

[解] n 个数中 r 的组合，其中每 r 个数中，数不能相同。另外，任何两组组合的数，所包含的数也不应相同。例如，5、4、3 与 3、4、5。为此，约定前一个数应大于后一个数。

将上述两条不允许为条件，当 $r=3$ 时，可用三重循环进行搜索。

[程序]

```

Program zuhell;
const n=5;
var i,j,k,t:integer;
begin t:=0;
  for i:=n downto 1 do
    for j:=n downto 1 do
      for k:=n downto 1 do
        if (i>j)and(j>k) then
          begin t:=t+1;writeln(i:3,j:3,k:3);end;
        writeln(' total=',t);
      end.
    end.
  end.
或者
Program zuhel2;
const n=5;r=3;
var i,j,k,t:integer;
begin t:=0;
  for i:=n downto r do
    for j:=i-1 downto r-1 do
      for k:=j-1 downto 1 do
        begin t:=t+1;writeln(i:3,j:3,k:3);end;
      writeln(' total=',t);
    end.
  end.

```

这两个程序，前者穷举了所有可能情形，从中选出符合条件的解，而后者比较简洁。但是这两个程序都有一个问题，当 r 变化时，循环重数改变，这就影响了这一问题的解，即没有一般性。

但是，很多情况下穷举搜索法还是常用的。

二、递归法

递归法也是常用的方法。

【例】 仍以前节例题为例，找 n 个数的 r 个数的组合。要求：

输入： $n, r=5 \quad 3$

```

输出： 5      4      3
       5      4      2
       5      4      1
       5      3      2
       5      3      1
       5      2      1
       4      3      2
       4      3      1
       4      2      1
       3      2      1
total=10      {组合的总数}

```

【解】 分析所提示的 10 组数。首先固定第一位数（如 5），其后是在另 4 个数中再“组合” 2 个数。这就将“5 个数中 3 个数的组合”推到了“4 个数中 2 个数的组合”上去了。第一位数可以是 $n-r$ （如 5-3） \rightarrow 个数中 \rightarrow 数组组合递推到 $n-1$ 个数中 $r-1$ 个数有组合，这是一个递归的算法。即：

```

Procedure comb(n,r:integer);
var i:integer;
begin for i:=n downto r do
  begin {固定 i 的输出位置}
    comb(i-1,r-1); {原过程递推到 i-1 个数的 r-1 数组组合}
  end;
end;

```

再考虑打印输出格式。

【程序】

```

Program zuhe2;
var k,n,r:integer;

```

```

Produce  comb(n,r:integer);
var i,temp:integer;
begin for i:=n downto r do
  if (i<>n)and(k<>r) then {k 为过程外定义的}
    begin for temp:=1 to (k-r)*3 do write(' '); {确定 i 的输出位置}
    write(i:3);
    if i>1 then comb(i-1,r-1); {递推到下一情形}
    else writeln;
end; end;
Begin {main}
write(' n,r=');readln(n,r);
if r>n then
  begin writeln(' Input n,r error!');halt; end;
comb(n,r); {调用递归过程}
End;

```

三、回溯法

回溯法是一种选优搜索法，按选优条件向前搜索，以达到目标。但当探索到某一步时，发现原先选择并不优或达不到目标，就退回一步重新选择，这种走不通就退回再走的技术为回溯法，而满足回溯条件的某个状态的点称为“回溯点”。

【例】再以前例说明，找 n 个数中 r 个数的组合。

【解】将自然数排列在数组 A 中：

$A[1]$	$A[2]$	$A[3]$
5	4	3
5	4	2
...		
3	2	1

排数时从 $A[1] \rightarrow A[2] \rightarrow A[3]$ ，后一个至少比前一个数小 1，并且应满足 $r_i + A[r_i] > r$ 。若 $r_i + A[r_i] \leq r$ 就要回溯，该关系就是回溯条件。为直观起见，当输出一组组合数后，若最后一位为 1，也应作一次回溯（若不回，便由上述回溯条件处理）。

【程序】

```

program zuhe3;
type tp=array[1..100] of integer;
var n,r:integer;

procedure comb2(n,r:integer;a:tp);
var i,ri:integer;
begin ri:=1;a[1]:=n;
  repeat
    if ri<>r then {没有搜索到底}
      if ri+a[ri]>r then {是否回溯}
        begin a[ri+1]:=a[ri]-1;
              ri:=ri+1;
        end
      else
        begin ri:=ri-1; a[ri]:=a[ri]-1;end; {回溯}
    else
      begin for j:=1 to r do write(a[j]:3);writeln; {输出组合数}
        if a[ri]=1 then {是否回溯}
          begin ri:=ri-1; a[ri]:=a[ri]-1;end; {回溯}
        else a[ri]:=a[ri]-1; {递推到下一个数}
      end;
  until a[1]<r-1;

```



```

end;

begin {MAIN}
  write('n,r=');readln(n,r);
  if r>n then
    begin writeln('Input n,r error!');halt; end
  comb2(n,r);
end.

```

第三章 综合题解

综合测试题（一）

1、寻找数：求所有这样的三位数，这些三位数等于它各位数字的立方和。

例如， $153=1^3+5^3+3^3$ 。

[解]穷尽三位数，用一个循环语句。其数码可用“模取”运算 MOD 完成。

[程序]

```

PROGRAM lifang;
uses crt;
var i,a,b,c:integer;
begin
  clrscr;
  for i:=100 to 999 do
    begin c:=i mod 10; {取个位数}
      b:=(i div 10) mod 10; {取十位数}
      a:=i div 100; {取百位数}
      if i=a*a*a+b*b*b+c*c*c then writeln(i:6);
    end;
end.

```

2、最小自然数：求具有下列两个性质的最小自然数 n：

(1) n 的个位数是 6；

(2) 若将 n 的个位数移到其余各位数字之前，所得的新数是 n 的 4 倍。

[解]仍用穷举法寻找，当找到一个符合条件者便停止。“找到便停止”的重复，宜采用 repeat-until 循环。

由于不知道 n 是几位数，个位数移到前面去应借助一个指定位数的数。设为 e。

[程序]

```

program minnum;
var n,e:integer;
begin e:=1;n:=1;
  repeat e:=10*e;
    repeat n:=n+1;
      until not((n=e)or((10*n+6)*4=(6*e+n)));
    until ((10*n+6)*4<>6*e+n);
    writeln(10*n+6);
  end.

```

3、找素数：寻找 160 以内的素数，它的倒序数（如 123 的倒序数为 321）、数码和、数码积不是素数便是 1。

[解]倒序数、数码和、数码积都需对原数分解，分解后再查它们是否符合条件。

[程序]

```

program sushu;
uses crt;
var i, j, n, s, p, f: integer;
function cond(k: integer): boolean;
var j: integer; b: boolean;
begin
  b:=true; {为 1 时表示 k 为素数}
  if k<>1 then for j:=2 to k-1 do if k mod j=0 then b:=false;
  if k=0 then cond:=false else cond:=b;
end;
begin {MIAN} clrscr;
  for i:=2 to 160 do
  if cond(i) then
    begin
      j:=i; n:=0; s:=0; p:=1;
      while j<>0 do
        begin
          f:=j mod 10;
          j:=j div 10;
          n:=n*10+f; {计算倒序数}
          s:=s+f; {计算数码和}
          p:=p*f; {计算数码积}
        end;
      if (cond(n)) and (cond(s)) and (cond(p)) then write(i:5);
    end;
  writeln; readln
end.

```

4、完全平方数：寻找具有完全平方数，且不超过 7 位数码的回文数。所谓回文数是指这样的数，它的各位数码是左右对称的。例如 121、676、94249 等。

[解]判断一个数是否回文数，可以将其转化成字符判断。也可以分解数，所谓分解就是将数的后半段倒置后再与前半段比较。这里采用分解的方法，其函数为 symm。

[程序]

```

program wcpfs;
uses crt;
var i: longint; s: longint;
function symm(n: longint): boolean;
var i, j, k: longint; m: longint;
begin
  i:=n; j:=1; {计算 n 的位数}
  repeat j:=j+1; i:=i div 10; until (i<=9) and (i>=0);
  k:=j div 2; m:=0;
  for i:=1 to k do {分解前后两位数}
    begin m:=m*10+(n mod 10); n:=n div 10; end;
  if j mod 2=1 then {n 是奇数位, 中间位不要} n:=n div 10;
  symm:=(m=n);
end;
begin {MAIN} clrscr;
  for i:=11 to round(sqrt(999999999)) do begin if symm(i*i) then writeln(i*i); end;
end.

```

1000000

5、成等差的素数：寻找 6 个成等差级数且小于 160 的素数。

[解]设级数为：n, n-d, n-2d, n-3d, n-4d, n-5d。若这 6 个数全为素数，则为要求的解。这里 d、n 均是要寻找的。仍用穷尽法，d 最大可为 33。判断素数函数为 isprime。

[程序]

```

PROGRAM dcss(input,output);
  var n,d:integer;
FUNCTION isprime(m:integer):integer; {判断素数函数}
  var b,i:integer;
  begin b:=1;
    if m<=0 then b:=0
    else for i:=2 to trunc(sqrt(m)) do if m mod i=0 then b:=0;
    isprime:=b;
  end;
begin {main}
  for d:=1 to 31 do for n:=160 downto 7 do
    if (isprime(n)=1)and(isprime(n-d)=1) then
      if (isprime(n-2*d)=1)and(isprime(n-3*d)=1) then
        if (isprime(n-4*d)=1)and(isprime(n-5*d)=1) then
          writeln(n:6,n-d:6,n-2*d:6,n-3*d:6,n-4*d:6,n-5*d:6);
  end.

```

***6、取数列：**取 $\{2^m, 3^n \mid m \geq 1, n \geq 1\}$ 中由小到大排列的前 70 项数。

[解]这个数的集合事实上存在两个数列：一个是 2^m ，另一个是 3^n 。若依次从两数列中取出小的进入新的数列，该新数列便为所求（这里不用数组，而直接打印输出）。

[程序]

```

program mn23;
const n=70;
var m2,n3:real; k:integer; f:text;
begin
  assign(f,'exmn.txt');rewrite(f); {输出结果在文件 exmn.txt 中}
  m2:=2;n3:=3;k:=0;
  while k<n do
    begin if m2<n3 then begin writeln(f,m2:40:0);m2:=m2*2; end
      else begin writeln(f,n3:40:0);n3:=n3*3; end;
      k:=k+1;
    end;
  close(f);
end.

```

7、发奖章：运动会连续开了 n 天，一共发了 m 枚奖章，第一天发 1 枚并剩下 $(m-1)$ 枚的 $1/7$ ，第二天发 2 枚并剩下 $(m-3)$ 枚的 $1/7$ ，以后每天按此规律发奖章，在最后一天即第 n 天发了剩下的 n 枚奖章。问运动会开了多少天？一共发了几枚奖章？

[解]由于题目涉及 $m-1$ 的 $1/7$ ，于是 $m-1$ 应是 7 的倍数，即 $m=7x+1$ 。递推 x ，寻找 m 、 n 。

[程序]

```

PROGRAM sport;
var m,n,x,b:integer;
begin x:=0;
  repeat
    x:=x+1;
    m:=7*x+1;
    n:=1;
    b:=1;
  while (m>n) and (b=1) do
    begin
      m:=m-n;
      if (m mod 7=0) then
        m:=(m div 7)*6
      else
        b:=0;
    end;
    n:=n+1;
  end;

```

```

    end;
until b=0;
writeln('n=',n,'    m=',7*x+1);
end.

```

8、猜名次：五个学生 A、B、C、D、E 参加某一项比赛。甲、乙两人在猜测比赛的结果。甲猜的名次顺序为 A、B、C、D、E，结果没有猜中任何一个学生的名次，也没有猜中任何一对相邻名次（所谓一对相邻名次，是指其中一对选手在名次上邻接。例如 1 与 2，或者 2 与 3 等）。乙猜的名次顺序为 D、A、E、C、B，结果猜中了两个学生的名次，并猜对了两对学生名次是相邻的。问比赛结果如何？答案为：E、D、A、C、B。乙猜对 C、B 为最后两名，两对相邻为 (D、A)、(C、B))。

[解] 设五名选手 A、B、C、D、E 的编号分别为 1、2、3、4、5。用五个变量 c1、c2、c3、c4、c5 标记第一名至第五名。算法仍用穷尽法。其中处理相邻问题用一个两位数表示，即 DA、AE、EC、CB 分别用 41、15、53、32 表示，并按两位数比较判断相邻问题。

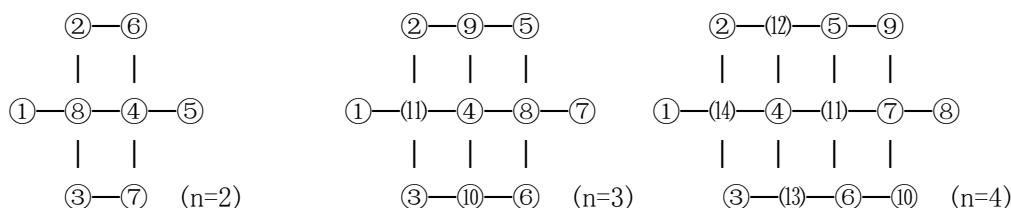
[程序]

```

program mingci;
var c1,c2,c3,c4,c5,s1,s2,t:integer;
begin for c1:=2 to 5 do
    for c2:=1 to 5 do
        if (c2<>2)and((c2-c1)<>1) then
            for c3:=1 to 5 do
                if (c3<>3) and ((c3-c2)<>1) then
                    for c4:=1 to 5 do
                        if (c4<>4) and ((c4-c3)<>1) then
                            for c5:=1 to 4 do
                                if (c5-c4)<>1 then
                                    begin s2:=0;
                                        if c1=4 then s2:=s2+1;
                                        if c2=1 then s2:=s2+1;
                                        if c3=5 then s2:=s2+1;
                                        if c4=3 then s2:=s2+1;
                                        if c5=2 then s2:=s2+1;
                                        s1:=0;
                                        t:=10*c1+c2;
                                        if (t=41)or(t=15)or(t=53)or(t=32) then s1:=s1+1;
                                        t:=10*c2+c3;
                                        if (t=41)or(t=15)or(t=53)or(t=32) then s1:=s1+1;
                                        t:=10*c3+c4;
                                        if (t=41)or(t=15)or(t=53)or(t=32) then s1:=s1+1;
                                        t:=10*c4+c5;
                                        if (t=41)or(t=15)or(t=53)or(t=32) then s1:=s1+1;
                                        if (s1=2)and(s2=2) then
                                            writeln(c1:3,c2:3,c3:3,c4:3,c5:3);
                                    end;
                                end;
                            end;
                        end;
                    end;
                end;
            end;
        end;
    end;
end.

```

9、填自然数：设有如图所示的 $3n+2$ 个球互连，将自然数 $1-3n+2$ 分别为这些球编号，使如图相连的球编号之差的绝对正好是数列 $1, 2, \dots, 3n+2$ 中各数。



[解] 填自然数的一种算法是：

(1) 先自左向右，第 1 列中间 1 个填数，然后第 2 列上、下 2 个填数，每次 2 列；但若 n 是奇数，最后 1 次只排第 1 列中间 1 个数。

(2) 自右向左, 先右第 1 列中间填数; 若 n 是奇数, 再右第 2 列中间填数。然后依次右第 1 列上、下 2 个填数, 再右第 2 列中间 1 个填数, 直到左第 2 列为止。

[程序]

```
program ziyangshu;
uses crt;
const size=25;
var a:array[0..2,0..size] of integer; i,k,m,n:integer;
begin clrscr;write('Input then n:');readln(n);k:=1;
  for i:=0 to n div 2 do
    begin a[1,2*i]:=k;k:=k+1;
      if ((i=n div 2)and(n mod 2=1)or(i<n div 2)) then
        begin a[0,2*i+1]:=k;k:=k+1; a[2,2*i+1]:=k;k:=k+1; end;
      end;
    if n mod 2=1 then begin a[1,n+1]:=k;k:=k+1;m:=n end
    else m:=n+1;
  writeln(m);
  for i:=0 to n div 2 do
    begin a[1,m-2*i]:=k;k:=k+1; writeln(1,' ',m-2*i,' ',k);
      a[0,m-2*i-1]:=k;k:=k+1; a[2,m-2*i-1]:=k;k:=k+1;
    end;
  write(' ':3);
  for i:=1 to n do write(a[0,i]:3);writeln;
  for i:=0 to n+1 do write(a[1,i]:3);writeln;write(' ':3);
  for i:=1 to n do write(a[2,i]:3);writeln;
end.
```

综合测试题 (二)

1、回文问题: 递归法判断所输入的一行字符是否回文。这里所说的回文是指输入的一行字符,

以“-”字符为中心, 其两边的字符是左右对称的。例如:

输入: ABCDE-EDCBA ↓

输出: It is symmetry. {输入一行字符是回文}

[解] 设一行字符为 $M-W$, 对于 M 分解成由 $ch1$ 标记的一个字符与一子串 m ; 对 w 分解成一字符子串 w 和由 $ch2$ 标记的一个字符, 因此 $M-W$ 这“回文”取决于: (1) $m-w$ 是回文; (2) $ch1=ch2$ 。即将原问题递推到 $m-w$ 的解。递归终止条件是 M 与 W (或 m 与 w) 长度为 0。“回归”时, 若 $m-w$ 是回文且 $ch1=ch2$, 则 $M-W$ 是回文; 否则 $M-W$ 就不是回文。程序使用递归过程 pp 。

[程序]

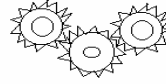
```
PROGRAM MW;
var ch:char;
function pp:boolean;
var ch1,ch2:char; bl:boolean;
begin read(ch1);
  if ch1='-' then bl:=true
  else begin
    if pp then begin read(ch2);bl:=ch1=ch2;end
    else bl:=false;
  end;
  pp:=bl;
end;
begin {MAIN}
```

```

write('Input string:');
if pp then begin read(ch);
  if ord(ch)=13 then begin writeln;writeln('It is symmetry.');

```

- 2、三齿轮问题：**三个齿轮啮合。如图在齿轮箱里三个齿轮互相衔接，某瞬间两对齿相遇，问各转多少圈后，这两对齿同时重逢。如图示。
(说明：用 a, b, c 分别表示三个齿轮的齿数。)



[解] 这一问题是最小公倍数问题。设三齿轮齿数分别是 na 、 nb 、 nc ， $[na, nb, nc]$ 为最小公倍数，相遇各齿轮所转的圈数为最小公倍数除以自己的齿数。

[程序]

```

{$I-}
program cl;
var na,nb,nc,ma,mb,mc,l3:integer;

function gcd(x,y:integer):integer; { 求最大公约数函数 }
var r:integer;
begin
  repeat r:=x mod y;x:=y;y:=r;until r=0;
  gcd:=x;
end;

function lcm(x,y:integer):integer; { 求最小公倍数函数 }
begin lcm:=(x*y div gcd(x,y));
end;

function lcm3(a1,a2,a3:integer):integer; { 求三个数的最小公倍数函数 }
begin lcm3:=lcm(lcm(a1,a2),a3); end;

begin {main}
  write('na,nb,nc=');readln(na,nb,nc); 读入三齿轮齿数
  if (na<1)or(nb<1)or(nc<1) then begin writeln('Input error!');exit;end;
  l3:=lcm3(na,nb,nc); {求 na,nb,nc 的最小公倍数}
  ma:=l3 div na; {求各齿轮所转的圈数}
  mb:=l3 div nb; mc:=l3 div nc;
  writeln('For mesh must rotate about rings:',ma:3,mb:3,mc:3);
end.

```

- 3、计算合数：**一个整数 n ($n \leq 100$) 可以有多种分划，使其分划的一系列整数之和为 n 。例如：

输入： $n=6$

输出文件 hs.out，格式内容为

```

6
5 1

```

瞿有甜整理

```

4 2
4 1 1
3 3
3 2 1
3 1 1 1
2 2 2
2 2 1 1
2 1 1 1
1 1 1 1 1 1
total=11    {表示分划数有 11 种}

```

[解]采用递归算法。从最大合数开始分划，若当前分划数之和仍不大于 n ，则继续分解。否则回溯再寻找分划。

[程序]

```

{$S-}
program hs;
var i, j, m, k, t: integer;
    n: array[0..100] of integer;
    f: text;

procedure output_sum;
var j: integer;
begin t:=t+1; j:=0;
    while n[j]<>0 do {输出所有不为0的合数}
        begin write(f, n[j]:3); j:=j+1; end;
    writeln(f);
end;

procedure sum(i: integer);
begin
    if m-n[i]<=n[i] then {整数分解后的余数不大于已分解的合数}
        begin n[i+1]:=m-n[i]; m:=m-n[i]; i:=i+1; n[i+1]:=0; end
    else {整数分解后的余数大于已分解的合数}
        begin n[i+1]:=n[i]; m:=m-n[i]; i:=i+1; end;
    if m<>n[i] then sum(i) {未分解完继续分解}
    else output_sum; {输出合数}
    if n[i]>1 then {若合数大于1可继续分解, 否则回溯寻找大于1的合数}
        begin n[i]:=n[i]-1; sum(i); end
    else
        begin while (n[i]=1) and (i>0) do
            begin i:=i-1; m:=m+n[i]; end;
            if i<>0 then {找到大于1的合数, 则继续分解}
                begin n[i]:=n[i]-1; sum(i); end;
        end;
    end;
end;
begin {MAIN}
    assign(f, 'hs.out'); rewrite(f);
    瞿有甜整理

```

```

write('Input a number:');readln(n[0]);
t:=0;m:=n[0];k:=n[0]; {第1个合数设为要分解的数}
for i:=1 to k do
  n[i]:=0; {除第1个合数外所有合数初值均为0}
output_sum; {输出第1次分解的合数}
while n[0]<>1 do {第1个合数不为1,则继续分解寻找合数}
  begin n[0]:=n[0]-1;i:=0;
    sum(0); m:=k;
  end;
writeln(f,'total=',t); close(f)
end.

```

4、旅行路线选择: 设有 n 个城市（或景点），今从某市出发遍历各城市，使之旅费最少（即找出一条旅费最少的路径）。

输入部分：各城市间的旅费表由输入文件提供。

输出部分：旅费最少的一条路径及总费用。

例如：

输入文件名：ex14501.dat

输出文件名：1.out

其中，输入文件 ex14501.dat 的内容如下：

0	17	13	24	10
10	0	20	9	6
17	29	0	21	28
12	10	22	0	19
12	18	31	20	0

输出文件 1.out 的内容如下：

The route path is:0->2->3->1->4->0 {最少旅费城市路径}
 Total of traveling expense: 62 {最少旅费数}

[解] 设矩阵元素 a_{ij} 表示从第 i 号城市到第 j 号城市之旅费。并设城市间往返旅费可以不等（即 $a_{ij} \neq a_{ji}$ ）。 a_{ii} 是没有意义的，由于问题是求最少，因此 a_{ii} 不应为零，今试为无穷（ ∞ ）。各城市间旅费如下表：

∞	17	13	24	10
10	∞	20	9	6
17	29	∞	21	28
12	10	22	∞	19
12	18	31	20	∞

问题的算法是在表每行中找最小元素，并用该数减该行非 ∞ 元素。再对每列也施同样工作，形成一个新表（保证每行、每列均不少于1个为零），所有减数累加为min（其含义为旅费下界，即旅费不会少于min）。旅行路程因成环路，故可设起点是第0号城市。若选第 i 号到第 j 号城市，则表上 b_{ij} 表示还需旅费，同时由于选了 $i \rightarrow j$ ，则 i 不可能再选向其它城市，则第 i 行全填 ∞ ，同理，由于 j 已由 i 过来，则第 j 城市不可能再由其它城市过来，第 j 列也全填上 ∞ 。对新矩阵再施每行至少有一个0，每列至少有一个0，找出余下城市遍历所需旅费下界 m_j 。对于不同的 j ，比较 $m_j + b_{ij}$ 以最小的一个为选定从 i 到达的城市，并将选择路径记下。如此重复直到选完。下列表表示了分枝选择和每次选择的旅费下界。

初始表： min=61

*	7	0	11	0
4	*	11	0	0
0	12	*	1	11
2	0	9	*	9
0	6	16	5	*

从 0 号城市出发的 4 种可能:

0→1 min=9

*	*	*	*	*
4	*	4	0	0
0	*	*	1	11
0	*	0	*	7
0	*	9	5	*

0→3 min=9

*	*	*	*	*
4	*	2	*	0
0	12	*	*	11
0	0	0	*	9
0	6	7	*	*

0→2 min=0

*	*	*	*	*
4	*	4	0	0
0	*	*	1	11
0	*	0	*	7
0	*	9	5	*

0→2 min=0

*	*	*	*	*
4	*	2	0	*
0	12	*	1	*
2	0	0	*	*
0	6	7	5	*

从中选择应是 0→2。再从 2 出发, 有 3 种选择:

2→1 min=2

*	*	*	*	*
4	*	*	0	0
*	*	*	*	*
0	*	*	*	7
0	*	*	5	*

2→3 min=0

*	*	*	*	*
4	*	*	*	0
*	*	*	*	*
2	0	*	*	9
0	6	*	*	*

2→4 min=0

*	*	*	*	*
4	*	*	*	*
0	*	*	*	*
2	0	*	*	*
0	6	*	*	*

从中应选 2→3。再从 3 出发, 2 种选择:

3→1 min=0

*	*	*	*	*
4	*	*	*	0
*	*	*	*	*
*	*	*	*	*
0	*	*	*	*

3→4 min=10

*	*	*	*	*
0	*	*	*	*
*	*	*	*	*
*	*	*	*	*
0	0	*	*	*

从中选 3→1。再从 2 出发可选:

1→4 min=0

*	*	*	*	*
*	*	*	*	*
*	*	*	*	*
*	*	*	*	*
0	*	*	*	*

[程序]

```

program mm;
uses crt;
const n=5;
      max=1000;
type tp=array[0..n,0..n] of integer;
var  f1,f2:text; fn1,fn2:string;
      i,j,k,min,m1,m,jj,kk,si,sj:integer;
      a,b,c,d:tp;
      path,s:array[0..n] of integer;

procedure p(var b:tp;var m:integer);
var pi,pj,pk:integer;
    px,py:integer;

```

```

begin m:=0;
  for pi:=0 to n-1 do
    begin pk:=max;
      for pj:=0 to n-1 do
        if b[pi,pj]<pk then pk:=b[pi,pj];
      if (pk>0)and(pk<>max) then
        begin m:=m+pk;
          for pj:=0 to n-1 do
            if b[pi,pj]<>max then
              b[pi,pj]:=b[pi,pj]-pk;
          end;
        end;
      end;
    end;
  for pj:=0 to n-1 do
    begin pk:=max;
      for pi:=0 to n-1 do
        if b[pi,pj]<pk then pk:=b[pi,pj];
        if (pk>0)and(pk<>max) then
          begin m:=m+pk;
            for pi:=0 to n-1 do
              if b[pi,pj]<>max then
                b[pi,pj]:=b[pi,pj]-pk;
            end;
          end;
        end;
      end;
    end;
  end;

begin {MAIN}
  clrscr;write(' Input filename:');readln(fn1);
  write(' Output filename:');readln(fn2);
  assign(f1,fn1);reset(f1);assign(f2,fn2);rewrite(f2);
  writeln(f2,'Traveling expenses table:');
  for i:=0 to n-1 do {读城市旅费表}
    begin for j:=0 to n-1 do
      begin read(f1,a[i,j]);write(f2,a[i,j]:6);end;
      writeln(f2);
    end;
  for i:=0 to n-1 do a[i,i]:=max; {a[i,i] 无意义的, 设为 max}
  k:=0;path[0]:=0;i:=0;s[0]:=max; {从 0 号城市出发}
  for j:=1 to n-1 do s[j]:=0; {s[j] 为 0 时表示未到达 j 号城市, 否则已走过}
  for si:=0 to n-1 do {矩阵转置}
    for sj:=0 to n-1 do
      b[si,sj]:=a[si,sj];
  p(b,min); {调用函数, 计算旅费下界及最低旅费}
  repeat
    m1:=max;
    for j:=0 to n-1 do

```

```

if (s[j]=0)and(b[i, j]<>max) then 可到达的城市却未遍历
begin
  for si:=0 to n-1 do
    for sj:=0 to n-1 do
      c[si, sj]:=b[si, sj]; {矩阵转置}
    for kk:=0 to n-1 do {从 i 城市出发已不可能, 从其他城市到 j 城市也不可能}
      begin c[i, kk]:=max; c[kk, j]:=max; end;
    p(c, m); {调用函数, 计算从 i 到 j 城市的旅费下界}
    if m+b[i, j]<m1 then {求最小旅费路径}
      begin m1:=m+b[i, j]; jj:=j;
        for si:=0 to n-1 do
          for sj:=0 to n-1 do
            d[si, sj]:=c[si, sj]; {保存最小旅费表}
          end;
        end;
      for sj:=0 to n-1 do
        b[si, sj]:=d[si, sj]; {将所选择的最小旅费表存入初始表}
      min:=min+m1; i:=jj; {jj 号城市作为出发点}
      k:=k+1; path[k]:=jj;
      s[jj]:=max; sj:=max;
      for si:=0 to n-1 do {判断所有城市是否都已到达}
        if s[si]<>max then sj:=0;
      until sj=max; {直至所有城市都已走过为止}
      write(f2, 'The route path is:');
      for i:=0 to k do write(f2, path[i], '->');
      writeln(f2, '0');
      writeln(f2, 'Total of traveling expense:', min:5);
      close(f1); close(f2);
    end.

```