数论应用选讲

李学武

1．求解二元一次不定方程

我们的任务是解二元一次不定方程

ax+by=c (*)

其中a,b,c都是整数，所求的解(x,y)也是整数。

由于方程(*)如果有解，则解不是唯一确定的，所以称为不定方程。二元一次不定方程是一类重要的方程，应用很广。关于方程(*)的可解性，有下面的两个重要的结论：

（1）设gcd(a,b)表示整数a,b的最大公约数。方程(*)有解的充分必要条件是gcd(a,b)|c。（记号“x|y”表示x能整除y，即存在整数k，使y=kx）。

（2）如果(x_0,y_0)是方程(*)的一组解，则对任何整数t,(x_0+bt,y_0-at)也都是方程(*)的解。

许多讲数论的书都对这两个结论做了严格的论证。

下面我们讨论具体求解的方法。

为了避免计算中对负数和0的讨论，我们假定a>0，b>0，并且a>=b。

假定方程(*)有解，即系数满足：gcd(a,b)|c，这时，c’=c/gcd(a,b)一定是个整数。我们先讨论下面的方程：

ax+by=f (**)

其中f=gcd(a,b)，右端项恰好是左边两系数的最大公因子。

显然，如果(x_0,y_0)是方程(**)的一组解，则$(c'x_0,c'y_0)$也是方程(*)的一组解，即$a(c'x_0)+b(c'y_0)=(c'f)=c$。

在方程(**)中，取a=107,b=73,c=1，显然满足gcd(107,73)=1，方程

107x+73y=1 (***)

有解。我们用类似于求最大公约数的辗转相除的方法求这个解。利用辗转相除，可以得到：

107=73+1+34,
73=34*2+5,
34=5*6+4,
5=4*1+1,
4=1*4。

写成一般的形式：$s_i=t_i*q_i+r_i$，

$q_i=s_i/t_i$，$r_i=s_i\%t_i$，$s_{i+1}=t_i$，$t_{i+1}=r_i$。

认真分析上面的规律，可以归纳出具体的求解方法。我们先用下面的表格列出相应的关系：

在下表中，i为辗转计算的次数。q[i]=s[i]/t[i]为相除得到的商。表中没有列出r[i]，它就是后一列的t[i]。

i	0	1	2	3	4(end)	5
s[i]	107	73	34	5	4	
t[i]	73	34	5	4	1	0
q[i]	1	2	6	1	4	
x[i]	0	1	2	13	**15**	
y[i]	1	q[0]=1	3	19	**22**	

表14-1

关键算法是x[k]，y[k]的递推计算公式：

x[0]=0，x[1]=1； x[i+1]=x[i]*q[i]+x[i-1]，当i>1时。

y[0]=1，y[1]=q[0]； y[i+1]=y[i]*q[i]+y[i-1]，当i>1时。

当t[k]≠0且r[k]=s[k]%t[k]=0时，k就是最后一轮计算，这时，
x[k]=15，y[k]=22就是所要的结果，但要加上适当的符号后，才能得到原方程的解(x,y)：

$x=(-1)^{k-1}x[k]$， $y=(-1)^{k}y[k]$。

关于x[i]、y[i]的递推公式的推导较烦琐，就不在这里介绍了。

对于方程(***)，用这种方法可以求得x=-15，y=22，显然，107*（-15）+73*22=1，满足方程。

程序：

```
#include <stdio.h>
void result_one(int a,int b,int c,int *x2,int *y2)
/* 函数1：计算不定方程的一组解 */
{int q[200],x[200],y[200];
 int d1,d2,i,r,t,j,gcd;
 x[0]=0;y[0]=1;
 d1=a;d2=b;
 for(i=0;i<200;i++)
  {q[i]=d1/d2;
   r=d1%d2; d1=d2;d2=r;
   if(r==0)
    {gcd=d1; break;
    }
   if(i==0)
     {x[1]=1; y[1]=q[0];
     }
   else
    {x[i+1]=q[i]*x[i]+x[i-1];
     y[i+1]=q[i]*y[i]+y[i-1];
    }
  }
  for(t=-1,j=1;j<i;j++) t=-t;
  *x2=-t*x[i];
  *y2=t*y[i];
  /* 以上求方程ax+by=gcd(a,b)的一组解 */
   if(c%gcd!=0)
    {printf("No solution!\n");
     exit(0);
    }
   t=c/gcd;
   *x2=*x2*t;  *y2=*y2*t;
 /* 以上求方程ax+by=c的一组解 */
}
```

```
void test1(int a,int b,int c,int x,int y)
/* 函数2：检验解的正确性 */
 {if(a*x+b*y==c)
    printf("Ok!\n");
  else
    printf("Result error!\n");
 }
main()
/* 函数3：主函数 */
 {int a,b,c,x2,y2;
  printf("Input a(>0),b(>0),c:\n");
  scanf("%d%d%d",&a,&b,&c);
  result_one(a,b,c,&x2,&y2);
  test1(a,b,c,x2,y2);
  printf("x=%d y=%d \n",x2,y2);
 }
```

师：真正弄懂算法后，上面的程序并不难理解。关于算法，我再讲两点：

（1）如果a,b中有一个小于0，例如a<0，可以令x'=-x，解方程

ax'+by=c。

求解后，再令x=-x'就可以了。

（2）如果只求正数解，利用前面讲过的性质："如果(x_0,y_0)是方程(*)的一组解，则对任何整数t,(x_0+bt,y_0-at)也都是方程(*)的解"。可以通过解关于t的不等式组：

$x_0+bt>0$，$y_0-at>0$

就可以得到全部正整数解。

另外，这个程序还有可以进一步简化的余地，例如在函数result_one()中，几个数组都可以不要，因为算出q[i]后，q[i-2]及前面的元素都没用了，可以只用3个变量q1,q2,q3。程序效率会更高一些，对于x[i]，y[i]也是这样。

习题： 现有容量为M，N升的两个罐子（依次记为A,B）没有任何刻度，要求从水池中量出K升水放到另一个容器里。其中M,N,K都是正整数。例如，对于M=7,N=3,K=1，可以这样操作，先用A罐量M升水，再利用B罐从A罐中量两次N升水，A罐中剩余的就是所要的1升水。编程输出操作过程，或输出"不可能"。

2．佩尔方程

题：由键盘输入一个整数N($N<10^9$)，求不超过N的最大整数n,使

$$\frac{1^2+2^2+\cdots+n^2}{n}$$

是一个完全平方数。

分析：由

$1^2+2^2+\cdots+n^2=\dfrac{n(n+1)(2n+1)}{6}$,

$\therefore \dfrac{1^2+2^2+\cdots+n^2}{n}=\dfrac{(n+1)(2n+1)}{6}=\dfrac{2n^2+3n+1}{6}$,

于是，问题归结为：求整数m，使

$$2n^2+3n+1=6m^2.$$

即：$(4n+3)^2-3y^2=1$,

这是佩尔方程：$x^2-3y^2=1$

满足：$x\equiv 3 \pmod{4}$, $y\equiv 0 \pmod{4}$ 的解。

由佩尔方程的理论：$x^2-3y^2=1$

的正整数解(x_k, y_k)应满足：

$x_k+y_k\sqrt{3}=(2+\sqrt{3})^k$

再利用：$x_k\equiv 3 \pmod{4}$, $y_k\equiv 0 \pmod{4}$

就可以确定所需要的解。

3．Fibonacci 数列

（1995年NOI）已知整数m,n满足：

$(n^2-nm-m^2)^2=1$， (1)

求m^2+n^2的最大值，其中$m,n<10^6$由键盘输入。

分析：如果m=n，只能有m=n=1，因此可假定 m≠n，不妨设m<n。

令$n=m+u_k$，由(1)可导出$(m^2-mu_k-u_k^2)^2=1$，

再令$m=u_k+u_{k-1}$，由(1)可导出$(u_k^2-u_ku_{k-1}-u_{k-1}^2)^2=1$，

由此可得序列：n，m，u_k，u_{k-1}，…，u_1，u_0．切满足 $u_1=u_0=1$，以及

$u_k=u_{k-1}+u_{k-2}$,，即$\{u_k\}$为Fibonacci数列。