National Olympiad in Informatics

信息学奥赛之

数学一本通

青少年信息学奥林匹克竞赛实战辅导丛书

主编　林厚从 / 主审　王　宏

ACM-ICPC

NOIP-NOI-IOI

东南大学出版社
SOUTHEAST UNIVERSITY PRESS

信息学奥赛之数学一本通

林厚从　主编
王　宏　主审

东南大学出版社
SOUTHEAST UNIVERSITY PRESS
·南京·

内容提要

数学是计算机程序设计的灵魂。利用数学方面的知识、数学分析的方法以及数学题解的技巧，可以使得程序设计变得轻松、美观、高效，而且往往能反映出问题的本质。在国内外各项程序设计比赛(比如，ACM、NOI)活动中，越来越多地用到各种复杂的数学知识，对选手的数学修养要求越来越高。编写本书的目的就在于给广大 ACM 队员、NOI 选手以及编程爱好者，系统分析一些程序设计中常用的数学知识和数学方法。

本书的适用对象包括：中学信息学奥林匹克竞赛选手及辅导老师、大学 ACM 程序设计比赛选手及教练、高等院校计算机相关专业的师生、程序设计爱好者等。

图书在版编目(CIP)数据

信息学奥赛之数学一本通 / 林厚从主编. — 南京 : 东南大学出版社，2016.7

ISBN 978-7-5641-6576-5

Ⅰ. ①信… Ⅱ. ①林… Ⅲ. ①中学数学课—教学参考资料 Ⅳ. ①G634.603

中国版本图书馆 CIP 数据核字(2016)第 136163 号

信息学奥赛之数学一本通

出版发行	东南大学出版社
社　　址	南京市玄武区四牌楼 2 号
网　　址	http://www.seupress.com
出 版 人	江建中
责任编辑	张　煦
封面设计	余武莉
责任印制	张文礼

经　　销	全国各地新华书店
印　　刷	扬中市印刷有限公司
开　　本	787 mm×1092 mm　1/16
印　　张	24.50
字　　数	612 千字
版 印 次	2016 年 7 月第 1 版　2016 年 7 月第 1 次印刷
书　　号	ISBN 978-7-5641-6576-5
定　　价	58.00 元

本社图书若有印装质量问题，请直接与营销部联系。电话(传真)：025—83791830

前　言

N. Wirth 给出了程序设计的一个重要公式，程序＝算法＋数据结构。

作为一个程序员或者程序设计爱好者来说，不应该只把程序设计作为一门技术，更应该看成是一种艺术。其实，算法本身也是一门艺术，数据结构本身也是一门艺术。程序也好，算法也好，数据结构也好，其中都蕴涵了很多的数学，而数学更是一门艺术。如果把数学与程序设计完美地结合在一起，则是艺术的巅峰！

从某种意义上来说，计算机源于数学。而作为计算机科学核心技术的程序设计，与数学之间的关系更是密不可分，可以这样说，数学是计算机程序设计的灵魂。利用数学方面的知识、数学分析的方法以及数学解题的技巧，可以使得程序设计变得轻松、美观、高效，而且往往能反映出问题的本质。

在 ACM 国际大学生程序设计竞赛（ACM-ICPC）和全国青少年信息学奥林匹克竞赛（NOI）系列活动中，越来越多地出现了数学的影子，也用到了越来越多数学方面的知识，对选手的数学修养要求越来越高。本书的目的就在于给广大编程爱好者和信息学参赛者，介绍和总结一些程序设计中常用的数学知识和数学方法，希望能起到抛砖引玉的作用。

本书由全国青少年信息学奥林匹克竞赛金牌指导教师、常州市第一中学林厚从老师主编。2008 年，笔者编写出版了《数学与程序设计》一书，深受读者欢迎。相比而言，《信息学奥赛之数学一本通》知识内容更加丰富、体系结构更加完整、例题习题更加新颖，更加突出实战性和应用性。在编写过程中，吸收了很多 OIer 的灵感和智慧，参考了部分国家集训队员和教练员的论文，包括朱全明、贾志鹏、陈瑜希、董宏华、金策、高融、唐文斌、余林韵、朱泽园、俞华程、Matrix67 的博客、ACdreamers 的博客、pi9nc 的博客。福州市第三中学黄志刚老师和吴钰晗、闫书弈两位同学，以及常州市第一中学吴睿海、孔瑞阳等同学为本书的编写做了大量调试和校对工作。全国青少年信息学奥林匹克竞赛（NOI）科学委员会主席、清华大学计算机科学与技术系王宏老师，在百忙之中审定了全书。在此，一并表示感谢！由于水平有限，书中难免存在不当之处，恳请谅解，也欢迎广大读者批评指正，不胜感激！

书中所有例题的参考程序均采用 Dev-C＋＋实现。如果需要本书中所有题目的测试数据和 PASCAL 标程，请发邮件与我联系（hc. lin@163. com）。

笔　者

2016 年 6 月

目　录

① 本书中加“*”号内容为提高性知识，一般在省队选拔及 NOI 比赛中才会涉及。

第1章 数 论

数论被誉为数学的皇后，它的研究对象是我们经常接触的整数，例如求两个整数的最大公约数和最小公倍数，求一个正整数的素因子分解，求一个方程的整数解等等。

古希腊人把关于数的抽象关系的研究与用数进行计算的实际技能区别开，前者称为算术(arithmetic)，后者称为算术计算术(logistic)。这种分类法，从中世纪延续下来一直用到15世纪末，教科书中开始用单一的名称“算术”来论述数的理论方面和实用方面。有趣的是，今天的欧洲大陆，arithmetic有其原始意义，而在英国和美国，一般把arithmetic作为logistic的同义词。并且，在这两个国家，使用描述性的术语number theory来表示对于数的研究的抽象方面。

毕达哥拉斯和他的学派在数学上有很多创造，尤其对整数的变化规律感兴趣。例如，把全部因数之和(除其本身以外)等于本身的数称为完全数(Perfect Number，又称完美数)；而将本身大于其因数之和的数称为盈数；将小于其因数之和的数称为亏数。不错，上帝6天创造世界，6就是一个完全数，因为6=1+2+3。另一方面，阿尔克温说：整个人类是诺亚方舟上的神灵下凡，这一创造是不完善的，8是个亏数，因为8>1+2+4。直到1952年，人类才知道12个完全数，它们都是偶数，其中前三个是6，28和496。

一般认为，毕达哥拉斯及其后继者，连同这个团体的哲学，是数论发展的先驱，是后来把数论发展为神秘主义的基础。例如，亚姆利库，这位公元320年左右有影响的新柏拉图派哲学家，就曾把亲和数(Amicable Numbers)的发现归功于毕达哥拉斯。两个数是亲和的，即一个数的真因子的和等于另一个数。例如284和220就是亲和的，因为284的真因子是1，2，4，71，142，其和为220，而220的真因子有1，2，4，5，10，11，20，22，44，55，110，其和为284。后来又增添了神秘的色彩和迷信的意思，即分别写上这两个数的护身符会使两数的佩戴者保持良好的友谊，这种数在魔术、法术、占星学和占卦上，都起着重要的作用。奇怪的是，后来很长一段时间再没有发现新的亲和数，直到17世纪，费马手持一本《算术》，并在其空白处写写画画，竟把数论引上了近代的轨道，并于1636年宣布17 296和18 416为另一对亲和数。后来得知这只不过是重新发现，这对亲和数在13世纪末14世纪初就曾被斑纳发现过，并且也许曾被泰比特·伊本柯拉用于其公式。又过了两年，法国数学家笛卡儿给出了第三对。再后来，瑞士数学家欧拉着手于系统地寻找亲和数，于1747年给出了一个30对亲和数的表，后来又扩展到超过60对。在这种数的研究历史中，还有一件奇怪的事：一个十六岁的意大利男孩帕加尼尼在1886年发现了被人们忽视、比较小的一对亲和数：1 184和1 210。

出现了现代计算机后，数论的研究得到了迅猛发展，因为很多数论的证明和猜想都可以通过高速计算机来解决和验证，甚至不断被穷举出来。本章就介绍一些数论知识及其在计算机编程中的实际应用。

1.1 整除

设 a 是非零整数，b 是整数。如果存在一个整数 q，使得 $b=a*q$，那么就说 b 可被 a 整除，记作 $a|b$，且称 b 是 a 的倍数，a 是 b 的约数(因子)。

例如 3|12，21|63。整除具有以下一些性质：

1. 如果 $a|b$ 且 $b|c$，那么 $a|c$。

2. $a|b$ 且 $a|c$ 等价于对任意的整数 x 和 y，有 $a|(b*x+c*y)$。

3. 设 $m\neq 0$，那么 $a|b$ 等价于 $(m*a)|(m*b)$。

4. 设整数 x 和 y 满足下式：$a*x+b*y=1$，且 $a|n$、$b|n$，那么 $(a*b)|n$。

证明：因为 $a|n$ 且 $b|n$

根据性质 3 可得：$(a*b)|(b*n)$ 且 $(a*b)|(a*n)$

再由性质 2 可得：$(a*b)|(a*n*x+b*n*y)$

其中：$a*n*x+b*n*y=n*(a*x+b*y)=n*1=n$　所以：$(a*b)|n$

5. 若 $b=q*d+c$，那么 $d|b$ 的充要条件是 $d|c$。

另外还有一些有用的例子，比如：若 2 能整除 a 的最末位(约定 0 可以被任何数整除)，则 $2|a$；若 4 能整除 a 的最后两位，则 $4|a$；若 8 能整除 a 的最后三位，则 $8|a$；……若 3 能整除 a 的各位数字之和，则 $3|a$；若 9 能整除 a 的各位数字之和，则 $9|a$；若 11 能整除 a 的偶数位数字之和与奇数位数字之和的差，则 $11|a$。同时，能被 7、11、13 整除的数的特征是：这个数的末三位数与末三位以前的数字所组成的数之差能被 7、11、13 整除，这个数就能被 7、11、13 整除。

【例 1.1-1】 教堂(church. *[①]，64 MB[②]，1 秒[③])

【问题描述】

ROMA 城中有一些古典的印度式建筑，这些建筑和周围的欧洲建筑风格格格不入。这些伪装成教堂的建筑其实是某国特工的基地。Tomas 接受了一项任务，就是从某个教堂出发，逐个访问这些教堂，搞清楚每一个教堂的内部结构，并回到出发的地方。这些教堂很有规律地构成了一个 $m*n$ 的矩形，每个教堂和它的八个方向的教堂有直接的路径相连。水平或垂直方向相邻的教堂之间的路程均为 1。请问 Tomas 至少需要走多远的路，才能完成这个危险而艰巨的任务呢？

【输入格式】

输入一行两个整数 m 和 n($m,n\leqslant 10\ 000$)。

【输出格式】

输出一行一个实数，表示最少需要走的路程，保留两位小数。

【输入和输出样例】

church. in	church. out
2 3	6.00

① church. * 表示该题输入文件名为 church. in，输出文件名为 church. out，源程序名为 church. cpp、church. pas 等，下同。

② 64 MB 为空间限制，下同。

③ 1 秒为每个测试定的时间限制，下同。

【样例说明】

如图1.1-1所示(虚线)的路线1—2—3—6—5—4—1是最短的,但如果按照1—2—6—3—5—4—1的顺序则需要走6.83的距离。所有教堂的编号是逐行进行的,同一行教堂按从左向右的次序编号。

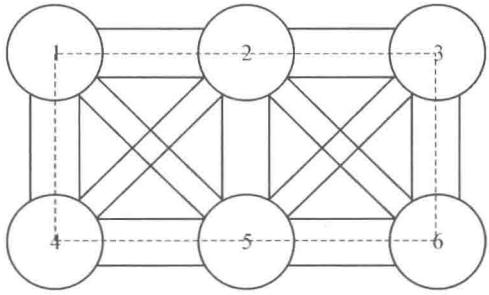

图1.1-1 最短路程示例

【问题分析】

列举一些小数据会发现,答案与m、n的奇偶性有关。具体编程时,还要注意一些特殊情况,如m或者n等于1的情况。

【参考程序】

```
#include<stdio.h>
using namespace std;
int n,m;
int main(){
    scanf("%d%d",&n,&m);
    if(n==1||m==1)
      printf("%.2lf\n",(double)(m+n-2)*2);
    else
      if(m*n%2==1)
        printf("%.2lf\n",(double)m*n-1+1.414);
      else
        printf("%.2lf\n",(double)n*m);
    return 0;
}
```

【例1.1-2】 密码(strongbox.*,64 MB,4秒)

【问题描述】

有一个密码箱,0到$n-1$中的某些整数是它的密码。且满足:如果a和b都是它的密码,那么$(a+b)\%n$也是它的密码(a,b可以相等,%表示整除取余数,下同),某人试了k次密码,前$k-1$次都失败了,最后一次成功了。

问:该密码箱最多有多少不同的密码。

【输入格式】

输入第一行两个整数分别表示n,k。

第二行为 k 个用空格隔开的非负整数，表示每次试的密码。

数据保证存在合法解。

【输出格式】

输出一行一个数，表示结果。

【输入和输出样例】

strongbox. in	strongbox. out
42 5 28 31 10 38 24	14

【数据规模】

对于 10%的数据：$N\leqslant 10^4$，$k\leqslant 100$；

另有 10%的数据：$N\leqslant 10^9$，$k\leqslant 100$；

另有 10%的数据：$N\leqslant 10^9$，$k=1$；

对于前 60%的数据：$k\leqslant 1\ 000$；

对于 100%的数据：$1\leqslant k\leqslant 250\ 000$，$k\leqslant n\leqslant 10^{14}$；

【问题分析】

首先，二元一次不定方程的一般形式为 $a*x+b*y=c$。其中 a,b,c 是整数，$a*b\neq 0$。此方程有整数解的充分必要条件是 $GCD(a,b)\mid c$。

设 x_0,y_0 是该方程的一组整数解，那么该方程的所有整数解可表示为：

$$x=x_0+\frac{b}{GCD(a,b)}t,y=y_0-\frac{a}{GCD(a,b)}t$$

设 x 是密码，那么观察这个式子：$x*k-n*c=GCD(x,n)$。

可知：这个方程对于 k 和 c 是一定有正整数解的。

所以：一定存在一个 k，使得：$x*k\%n=GCD(x,n)$。

由题意，x 是密码，那么 $(x+x)\%n=2x\%n$ 也是密码，$(2x\%n+x)\%n=3x\%n$ 也是密码，以此类推，对于任意正整数 k，如果 x 是密码，那么 $x*k\%n$ 也是密码。

由上可知：$x*k\%n=GCD(x,n)$，$x*k\%n$ 也是密码。

所以：$GCD(x,n)$ 也是密码。

结论 1：如果 x 是密码，那么 $GCD(x,n)$ 也是密码。

设 x,y 是两个密码，易得 $(p*x+q*y)\%n$ 也是密码 $(p,q\geqslant 0)$。

$a*x+b*y=GCD(x,y)$ 一定有解，所以：

$a*x+b*y\equiv$[①] $GCD(x,y)\pmod n$ 一定有解[当 $a*x+b*y=GCD(x,y)$ 就可以了]。

因为：$a*x+b*y\equiv a*x+b*y+p*n*x+q*n*y\pmod n$

所以：$(a+p*n)*x+(b+q*n)*y\equiv a*x+b*y\pmod n$ $(p,q$ 为任意整数$)$

所以：$(a+p*n)*x+(b+q*n)*y\equiv GCD(x,y)\pmod n$ 一定有解，且 $(a+p*n)$，$(b+q*n)\geqslant 0$。

因为：$((a+p*n)*x+(b+q*n)*y)\%n$ 是密码，所以 $GCD(x,y)$ 也是密码。

① $a\equiv b\pmod n$ 表示 a 和 b 关于模 n 同余。

结论2：如果 x,y 是密码，那么 $GCD(x,y)$ 也是密码。

对于任意一个密码集合 A，分析：

1. 设 A 中所有数的 GCD 为 x；

2. 由结论2得 $x\in A$；

3. 如果密码集合中有比 x 小的数 y，则 $GCD(x,y)<x$，不符合设定，所以 x 是 A 中最小的数。

所以 A 中的所有数为 $x,2x,3x,4x\cdots$

约束条件如下：

1. 因为 A 中的数要尽量多，所以 x 要尽量小；

2. 根据结论1，集合中有两个数为 $a[k]$，$GCD(n,a[k])$，所以 $x|GCD(n,a[k])$；

3. 对于任何 $1\leqslant j<k$，x 不能整除 $a[j]$（否则 $a[j]$ 也是密码）。

实现方法：先设 $a[k]=GCD(n,a[k])$，可以先在根号时间复杂度内处理出 $a[k]$ 的所有因子，存在 q 数组中，接着去除所有为 $GCD(a[i],a[k])$ 的因数，设去除后最小的因子为 x，答案为 n/x。

【参考程序】

```
#include<stdio.h>
#include<algorithm>
using namespace std;
#define LL long long
LL n,k;
LL a[250005];

int gcd(int a,int b){
    return b? gcd(b,a%b):a;
}

int main(){
    scanf("%I64d%I64d",&n,&k);
    for(int i=1;i<=k;i++)
      scanf("%I64d",&a[i]);
    a[k]=gcd(a[k],n);
    for(int i=1;i<k;i++)
      a[i]=gcd(a[i],a[k]);
    for(LL i=1;i*i<=a[k];i++)
      if(a[k]%i==0){
        q[++cnt]=i;
        if(i*i!=a[k])q[++cnt]=a[k]/i;
      }
    sort(q+1,q+cnt+1);
```

```
    for(int i=1;i<k;i++)
      f[lower_bound(q+1,q+cnt+1,a[i])-q]=1;
    for(int i=1;i<=cnt;i++)
      if(f[i])
        for(int j=1;j<i;j++)
          if(q[i]%q[j]==0)
            f[j]=1;
    for(ans=1;f[ans];ans++);
    printf("%d\n",n/q[ans]);
    return 0;
}
```

1.2 同余

若 a,b 为两个整数，且它们的差 $a-b$ 能被某个自然数 m 所整除，则称 a 就模 m 来说同余于 b，或者说 a 和 b 关于模 m 同余，记为：$a\equiv b(\bmod m)$。它意味着：$a-b=m*k$（k 为某一个整数）。

例如，$32\equiv 2(\bmod 5)$，此时 k 为 6。

对于整数 a,b,c 和自然数 m,n，对模 m 同余具有以下一些性质：

1. 自反性：$a\equiv a(\bmod m)$
2. 对称性：若 $a\equiv b(\bmod m)$，则 $b\equiv a(\bmod m)$
3. 传递性：若 $a\equiv b(\bmod m)$，$b\equiv c(\bmod m)$，则 $a\equiv c(\bmod m)$
4. 同加性：若 $a\equiv b(\bmod m)$，则 $a+c\equiv b+c(\bmod m)$
5. 同乘性：若 $a\equiv b(\bmod m)$，则 $a*c\equiv b*c(\bmod m)$

   若 $a\equiv b(\bmod m)$，$c\equiv d(\bmod m)$，则 $a*c\equiv b*d(\bmod m)$
6. 同幂性：若 $a\equiv b(\bmod m)$，则 $a^n\equiv b^n(\bmod m)$
7. 推论 1：$a*b \bmod k=(a \bmod k)*(b \bmod k)\bmod k$
8. 推论 2：若 $a \bmod p=x$，$a \bmod q=x$，p、q 互质，则 $a \bmod p*q=x$。

证明：因为 $a \bmod p=x$，$a \bmod q=x$，p、q 互质

则一定存在整数 s,t，使得 $a=s*p+x$，$a=t*q+x$

所以，$s*p=t*q$

则一定存在整数 r，使 $s=r*q$

所以，$a=r*p*q+x$，得出：$a \bmod p*q=x$

但是，同余不满足同除性，即不满足：$a \operatorname{div} n\equiv b \operatorname{div} n(\bmod m)$。

【例 1.2-1】 指数取余(mod. *，64 MB，1 秒)

【问题描述】

输入整数 m,n,k，求 $m^n \bmod k$ 的值。$m,n,k*k$ 为长整型范围内的自然数。

【输入格式】

输入一行 3 个整数，分别为 m,n 和 k。

【输出格式】

输出一行一个整数，表示结果。

【输入和输出样例】

mod. in	mod. out
2 10 9	7

【问题分析】

举例来说，如何求 3^{89} mod 7 呢？

$3^1 \equiv 3 \pmod 7$

$3^2 \equiv 3^2 \pmod 7 \equiv 2 \pmod 7$

$3^4 \equiv (3^2)^2 \pmod 7 \equiv 2^2 \pmod 7 \equiv 4$

$3^8 \equiv (3^4)^2 \pmod 7 \equiv 4^2 \pmod 7 \equiv 2$

$3^{16} \equiv (3^8)^2 \pmod 7 \equiv 2^2 \pmod 7 \equiv 4$

$3^{32} \equiv (3^{16})^2 \pmod 7 \equiv 4^2 \pmod 7 \equiv 2$

$3^{64} \equiv (3^{32})^2 \pmod 7 \equiv 2^2 \pmod 7 \equiv 4$

$3^{89} \equiv (3^{64}) * (3^{16}) * (3^8) * (3^1) \pmod 7 \equiv 4 * 4 * 2 * 3 \pmod 7 \equiv 96 \pmod 7 \equiv 5 \pmod 7$

所以，首先将 n 分解成 2 的幂次方，存放在一个数组 r 中，$r[i]=1$ 表示有 m^i 这一项。然后用递推的方法从小到大逐个求出 m^i mod k 的值（j 从 i 到 1）存放在数组 d 中。

其实，我们还可以利用以下递归公式进行求解：

$$x\hat{}y=\begin{cases}1 & \text{如果 } y=0\\ x*x\hat{}(y-1)① & \text{如果 } y \text{ 是奇数}\\ (x\hat{}2)\hat{}(y \text{ div } 2) & \text{如果 } y \text{ 是偶数}\end{cases}$$

也就是递归地对 n 进行二进制分解，同时计算。等价于将表达式分解成下面的形式：

$3^{89}=(3^{44})^2*3=((3^{22})^2)^2*3=(((3^{11})^2)^2)^2*3=((((3^5)^2*3)^2)^2)^2*3$

$=(((((3^2)^2*3)^2*3)^2)^2)^2*3$

【参考程序】

```
#include<stdio.h>
using namespace std;
int m,n,k;
int main(){
    scanf("%d%d%d",&m,&n,&k);
    int ans=1;
    for(;n;n>>=1,m=(long long)m*m%k)
      if(n&1)
        ans=(long long)ans*m%k;
    printf("%d\n",ans);
    return 0;
}
```

① $x\hat{}(y-1)$ 表示 x^{y-1}，下同。

【例 1.2-2】 Semi-prime H-numbers(POJ3292)

【问题描述】

形如 $4n+1$ 的数被称为“H 数”，乘法在“H 数”组成的集合内是封闭的。在这个集合中只能被 1 和本身整除的数叫做“H-素数”(不包括 1)，其余的数被称为“H-合数”。一个“H-合成数”是一个能且只能分解成两个“H-素数”乘积的“H-合数”(可能有多种分解方案)。比如 $441=21*21=9*49$，所以 441 是“H-合成数”。$125=5*5*5$，所以 125 不是“H-合成数”。

求 0～h 范围内“H-合成数”的个数。

【输入格式】

输入若干行，每行一个小于等于 1 000 001 的整数 h，一个 0 表示结束。

【输出格式】

对于每一行输入，输出一个数，表示答案。

【输入和输出样例】

poj3292.in	poj3292.out
21	0
85	5
789	62
0	

【问题分析】

利用同余结论，拓展一下筛选法求素数。如果一个数 i 是 H-素数，那么 $5i+4i*x$ 一定是 H 数但不是 H-素数，因为 $(5i+4i*x) \bmod 4=5i \bmod 4=(5 \bmod 4)*(i \bmod 4)=1*1=1$，且 $5i+4i*x=i(4x+5)$。

【参考程序】

```
#include<stdio.h>
using namespace std;
#define MAX_H 1000001+16
bool is_H_prime[MAX_H], is_H_semiprime[MAX_H];
int H_prime[MAX_H];
int accumulate[MAX_H];
int n;
int main(){
    for(int i=5;i<MAX_H;i+=4){
        if(is_H_prime[i])continue;
        H_prime[++n]=i;
        for(int j=i*5;j<MAX_H;j+=i*4)
            is_H_prime[j]=true;
    }
    for(int i=1;i<=n;i++)
        for(int j=1;j<=i&&H_prime[i]*H_prime[j]<MAX_H;j++)
```

```
                is_H_semiprime[H_prime[i] * H_prime[j]] = true;
        for(int i = 1;i<MAX_H; + + i)
            accumulate[i] = accumulate[i - 1] + is_H_semiprime[i];
        int h;
        scanf(" % d",&h);
        while(h){
            printf(" % d\n",accumulate[h]);
            scanf(" % d",&h);
        }
        return 0;
    }
```

1.3 最大公约数

一般地，设 $a_1,a_2,a_3,\cdots a_k$ 是 k 个非零整数，如果存在一个非零整数 d，使得 $d|a_1$，$d|a_2,d|a_3,\cdots d|a_k$，那么 d 就称为 $a_1,a_2,a_3,\cdots a_k$ 的公约数。公约数中最大的一个就称为最大公约数，记为 $GCD(a_1,a_2,a_3,\cdots a_k)$，显然它是存在的，至少为 1。当 $GCD=1$ 时，称这 n 个数是互质的或既约的。公约数一定是最大公约数的约数。

一般地，设 $a_1,a_2,a_3,\cdots a_k$ 是 k 个非零整数，如果存在一个非零整数 d，使得 $a_1|d$，$a_2|d,a_3|d,\cdots a_k|d$，那么 d 就称为 $a_1,a_2,a_3,\cdots a_k$ 的公倍数。公倍数中最小的一个就称为最小公倍数，记为 $LCM(a_1,a_2,a_3,\cdots a_k)$，显然它也是存在的。公倍数一定是最小公倍数的倍数。

1.3.1 辗转相除法

辗转相除法用来求两个数的最大公约数，又称欧几里得算法，其原理就是：$GCD(x,y)=GCD(x,y-x)$。

原理的证明如下：

设 $z|x,z|y$，则 $z|(y-x)$。

设 z 不是 x 的因子，则 z 不是 $x,y-x$ 的公因子。

设 $z|x,z$ 不是 y 的因子，则 z 不是 $x,y-x$ 的公因子。

代码实现如下：

```
int GCD(int x,int y){
    return y = = 0 ? x : GCD(y,x % y);
}
```

1.3.2 二进制算法

如果想要进一步提高 GCD 的效率，可以通过不断去除因子 2 来降低常数，这就是所谓的“二进制算法”。

若 $x=y$，则 $GCD(x,y)=x$，否则：

(1) 若 x,y 均为偶数，则 $GCD(x,y)=2*GCD(x/2,y/2)$；

(2) 若 x 为偶数，y 为奇数，则 $GCD(x,y)=GCD(x/2,y)$；

(3) 若 x 为奇数，y 为偶数，则 $GCD(x,y)=GCD(x,y/2)$；

(4) 若 x,y 均为奇数，则 $GCD(x,y)=GCD(x-y,y)$。

代码实现如下：

```
inline int GCD(int x,int y){
    int i,j;
    if(x==0) return y;
    if(y==0) return x;
    for(i=0;0==(x&1);++i)x>>=1;          // 去掉所有的 2
    for(j=0;0==(y&1);++j)y>>=1;          // 去掉所有的 2
    if(j<i) i=j;
    while(1){
      if(x<y)x^=y,y^=x,x^=y;             // 若 x<y 交换 x,y
      if(0==(x-=y)) return y<<i;
      // 若 x==y, gcd==x==y (就是在辗转减,while(1)控制)
      while(0==(x&1))x>>=1;              // 去掉所有的 2
    }
}
```

1.3.3 最小公倍数

求两个数的最小公倍数可以使用“逐步倍增”法，如求 3 和 8 的最小公倍数，可以让 n 从 1 开始逐步加 1，不断检查 $8*n$ 是不是 3 的倍数，直到 $n=3$ 时，$8*3=24$ 是 3 的倍数了。还可以直接使用以下定理来求解。

定理：a、b 两个数的最大公约数乘以它们的最小公倍数就等于 a 和 b 本身的乘积。

比如，要求 3 和 8 的最小公倍数，则 $LCM(3,8)=3*8\ \text{div}\ GCD(3,8)=24$。

1.3.4 扩展欧几里得算法

扩展欧几里德算法是用来在已知(a,b)时，求解一组(p,q)，使得 $p*a+q*b=GCD(a,b)$。

首先，根据数论中的相关定理，解一定存在。

其次，因为 $GCD(a,b)=GCD(b,a\%b)$，所以 $p*a+q*b=GCD(a,b)=GCD(b,a\%b)=p*b+q*a\%b=p*b+q*(a-a/b*b)=q*a+(p-a/b*q)*b$，这样它就将 a 与 b 的线性组合化简为 b 与 $a\%b$ 的线性组合。

根据前边的结论：a 和 b 都在减小，当 b 减小到 0 时，就可以得出 $p=1,q=0$。然后递归回去就可以求出最终的 p 和 q 了。

代码实现如下：