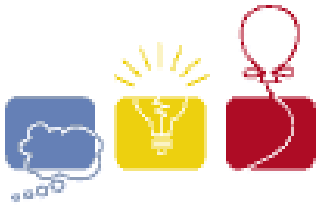


acm International Collegiate Programming Contest

数论在ACM竞赛中的运用

福州大学　陈鸿

数论

- 数论是最原始的两个数学分支，即算术与几何，保留下来的问题。传统的几何学已经枯萎，所有的问题都得到解决。而传统的算术却积累了越来越多的问题，成为难以穿越的密林。过去被认为是纯粹数学的，是专门研究整数的性质，整数按乘法性质划分，可以分成“素数”，“合数”，“1”，素数产生了很多一般人也能理解而又悬而未解的问题，如哥德巴赫猜想。两千多年来，数论学一个最重要的任务就是寻找一个**素数普遍公式**，人们花费了巨大的心血，始终未获成功。很多诸如此类的问题虽然形式上十分初等，但事实上却要用到许多艰深的数学知识。这一领域的研究从某种意义上推动了数学的发展，催生了大量的新思想和新方法。

- 卡尔·弗里德里希·高斯曾说：“数学是科学的皇后，数论是数学的皇后。”

一些问题

- 1.求出数16000001的一切除数。
- 2.直角三角边的某一边之长为48，试写出十对正整数，其中每一对是该三角形中另外两边之长。（丢番图分析）
- 3.小于5929且与之互质的正整数一共有多少？
- 4.试求出一个最小的，完全由数码3与7构成的正整数，此数以及其数字和均能被3与7整除。
- 5.求出能形成算术级数的三个平方数，四个行不行？（丢番图分析）
- 6.求出一个恰好拥有100个除数的最小数。
- 7.证明当n为素数时，1*2*3*4*...*(n-1)+1必能被n整除，但当n为合数时，决不能整除。
- 8.证明形如4x+1的任一素数必能唯一地表示为两个平方数之和。
- 9.找出一些数，其中的每一个都等于其不同除数之和（所谓除数，其中应包括1，但不包括此数本身）。（完全数）
- 10.找出x值的一般公式，使2^x-1得出素数。（梅森数）
- 11.证明任一偶数都可表为两个素数之和。（哥德巴赫猜想）
- 12.证明n>2时，x^n+y^n=z^n不可能有整数解。（费马大定理）

新浪微博
t.sina.com.cn

Fermat费马

http://t.sina.com.cn/1746521221

海外 法国

图卢兹议会荣誉议长；解析几何的发明者；亲和数之父；

+ 加关注　　推荐给朋友

微博筛选：默认

将一个立方数分成两个立方数，一个四次幂分为两个四次幂，或者一般地将一个高于二次的幂分为两个同次的幂，这是不可能的。关于此，我已发现一种美妙的证法，可惜新浪的微博只能写140个字，限制字数太少，写不下。

5月25日 10:15 来自新浪微博　　转发(3) | 收藏 | 评论(3)

进制问题

- 进制也就是进位制，是人们规定的一种进位方法。 对于任何一种进制---X进制，就表示某一位置上的数运算时是逢X进一位。 十进制是逢十进一，十六进制是逢十六进一，二进制就是逢二进一

- 假设一个数字N可以用P进制表示，那么有
- N = a0 * P^0 + a1 * P^1 + a2 * P^2 + … + ak * P^k
- 其中 (0 <= ai < P)

- 例子:
- (1)15分解为2进制
- 15 = 1 * 1 + 1 * 2 + 1 * 4 + 1 * 8(1111)
- (2)15分解为7进制
- 15 = 1 * 1 + 2 * 7
- ……

- 观察容易发现，N % P之后得到的是a0的值
- 同时 N 整除P以后 继续 %P 得到的是a1…
- 如此继续
- 于是可以得到P进制下的{a0,a1…ak}

ACM中的数论

- 数论可以称为'素论',大部分的题目都围绕着素数展开
- 数论在比赛中一般最多出现一题,而且难度一般不低,足够让基础不扎实的人冥思苦想一阵子.

什么是素数？

- 素数，亦称质数，指在一个大于1的自然数中，除了1和此整数自身外，无法被其他自然数整除的数。换句话说，只有两个正因数（1和自己）的自然数即为素数。

- 素数序列的开头是这样：

- 2，3，5，7，11，13，17，19，23，29，31，37，41，43，47，53，59，61，67，71，73，79，83，89，97，101，103，107，109，113

素数公式

- 在数学领域中，表示一种能够仅产生素数的公式。即是说，这个公式能够一个不漏地产生所有的素数，并且对每个输入的值，此公式产生的结果都是素数。迄今为止，人们尚未掌握任何一个易于计算的素数公式，但人们对于素数公式应该具备的性质已经有了大量的了解。

丢番图方程组

- 一个很著名的素数公式是以下的有26个未知数的由14个方程组成的丢番图方程组 Jones et al. (1976)：

$$0 = wz + h + j - q$$
$$0 = (gk + 2g + k + 1)(h + j) + h - z$$
$$0 = 16(k + 1)^3(k + 2)(n + 1)^2 + 1 - f^2$$
$$0 = 2n + p + q + z - e$$
$$0 = e^3(e + 2)(a + 1)^2 + 1 - o^2$$
$$0 = (a^2 - 1)y^2 + 1 - x^2$$
$$0 = 16r^2y^4(a^2 - 1) + 1 - u^2$$
$$0 = n + l + v - y$$
$$0 = (a^2 - 1)l^2 + 1 - m^2$$
$$0 = ai + k + 1 - l - i$$
$$0 = ((a + u^2(u^2 - a))^2 - 1)(n + 4dy)^2 + 1 - (x + cu)^2$$
$$0 = p + l(a - n - 1) + b(2an + 2a - n^2 - 2n - 2) - m$$
$$0 = q + y(a - p - 1) + s(2ap + 2a - p^2 - 2p - 2) - x$$
$$0 = z + pl(a - p) + t(2ap - p^2 - 1) - pm$$

- 对于这个方程组的所有正整数解：（a,b,...,z），$k+2$都是素数。

素数的判断

- 1.可以暴力枚举1..n的数字,如果存在某个数i,有n%i==0,那么这个数肯定不是素数,复杂度显然是O(n)
- 2.稍微优化一下,可以知道实际上n的因子是成对出现的(除非N是完全平方数).所以只需要枚举1..sqrt(N)判断既可,复杂度O(sqrt(N))

- 可是如果N很大,那如何判断?于是基于费马定理有一种方案可以解决,既随机化算法.
- 我们知道对于某个素数P,由费马定理得:
- a^p mod p=a
- 于是就有了希望使用这个来判断某个p是否是素数

- 但a^p mod p = a不是p是素数充分条件，存在有名但是极端稀少的Carmichael数，它们不是素数但是满足费马小定理，比如561, 1105, 1729, 2465, 2821, 6601, 8911, 10585, 15841, 29341...。所以如果我们只是随机的从1和p-1之间（这里p是一个待判断的正整数）取一个数a计算a^p mod p的值，如果它不是a，那么我们100%肯定p不是素数。但如果a^p mod p的值是a,p可能是素数也可能不是。

米勒-拉宾素数测试

- 我们看费马小定理的一个特殊情况，如果n是一个素数，那么phi(n)=n-1，而任意数字不大于n的数a都与n互质，于是费马小定理的推论为：
  当n为素数，而a为任意整数时，
  a^(n-1) =1 （mod n)

- 即一个数字n为素数的必要条件是对于任意a<n，都有 (a^(n-1)) mod n = 1.
  但是这是必要条件，根据计算，如果n为任意数字，有如果单独任取一个a<n，当(a^(n-1)) mod n = 1成立时，有25%可能性n为合数。
  所以我们可以通过多次随机选择不同的a，如果上面测试都被通过，我们就可以说n有很大的可能是素数。这种算法就是米勒-拉宾素数测试。

一个例题

- http://acm.fzu.edu.cn/problem.php?pid=1649

素数表的获得

- 这里只介绍**百万级别**的素数表的获得.
- 很自然想到的就是把1...N的数字全判断一次,然后把是素数的加到素数表里面.

- 实际上这么做是做了很多无谓的计算.

埃拉托斯特尼筛法

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

去掉含有因子2的数,这些数肯定不是素数

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

去掉含有因子3的数,这些数肯定不是素数

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

.....

最后幸免于难的数只有:2 3 5 7 11 13 17 19
全是素数

实现方案

- 如果某个数可以被小于它的某个数整除,那肯定不是素数,换句话就是说如果某数有不为1的因子,那肯定不是素数.
- 于是我们开一个数组bool prime[1000001]
- Prime[i]表示i是否是素数
- Prime[0]=prime[1]=1;//这里的1表示非素数

```
For(i=2;i*i<=1000000;++i)
 if(!prime[i])
   for(j=i;j*i<=1000000;++j)
     prime[i*j]=true;//这些数都包含i这个因子
```

- 得到以后就可以得到范围内的素数表了.
- 得到素数表一般只有2个目的,就是判素数和**分解素因子**.判素数只需要看prime[i]是否为0,如果为0就是素数.

分解素因子

- 数论中最经常使用的方法
- 通常使用试除法
- 得到素数表以后扫描素数表,然后把N的素因子加入解集.

- For(I = 0; I < plen; ++ i)
  - If(n % p[i] == 0) {
  - ……
  - ……
  - }
  - P[] 存放的是素数,而plen则表示素数表总长度
  - 对于一个数字N，若想分解它，那么素数表必须筛选到Sqrt(N) 的级别

N!中素因子P的个数

- 写成 : 1 * 2 * 3 * …. * N
- 我们先提取所有P的倍数 ：显然有 [N / P]个
- 之后我们提取的数字显然是： P * 2P * 3P…
- 提取P,之后我们又得到了 1* 2 * 3 * … * [N / P]
- So:
- cnt=0;
  while (N)
  {
  cnt+=N/p;
  N/=p;
  }

因子和

- (1+p1+p1^2+…+p1^c1)*(1+p2+p2^2+…+p2^c2)*…*(1+pk+pk^2+…+pk^ck)
- Pi表示N的第i种素因子
- Ci表示N的第i种素因子的个数

- 如果N比较大,那么结果很可能比较大,注意合适选择64 位…

因子个数

- 根据组合数学可以很容易的知道
- Ans=(1+c1)*(1+c2)*..*(1+ck)
- 为什么?
- 因为每个素因子选的个数0..c1,有(1+c1)种组合,根据组合数学的乘法原则,直接乘完就是解.

例题

- 1/n=1/a+1/b(a<b)的对数
- 我们要求求a<b的时候的解的个数

(因为a==b必然有一个解，a>b的解在顺序上和a<b的只是互相交换了一下而已)

- 化啊化

$$\frac{1}{n}=\frac{1}{n+k}+\frac{k}{n(n+k)}$$

$$\frac{1}{n}=\frac{1}{n+k}+\frac{1}{\frac{n^2+nk}{k}}$$

$$\frac{1}{n}=\frac{1}{n+k}+\frac{1}{\frac{n^2}{k}+n}$$

- 于是问题就很简单了
  由于n+k<2*n(如果==2n那么就是a==b了，不在我们讨论范围内)
  so:k<n
  而显然k是n^2的因子

欧拉函数

- 在数论，对正整数n，欧拉函数是少于或等于n的数中与n互质的数的数目。此函数以其首名研究者欧拉命名，它又称为Euler's totient function、Φ函数、欧拉商数等。 例如Φ(8)=4，因为1,3,5,7均和8互质。 从欧拉函数引伸出来在环论方面的事实和拉格朗日定理构成了欧拉定理的证明。

- Euler() ,Phi()
- 求完素因子以后
- Phi(n)=n*(1-1/p1)*(1-1/p2)*…*(1-1/pk)
- 如何优化?

```
typedef long long llong;
llong Eular(llong n)
{
    int i;
    llong ret=n;
    for(i=2;i*i<=n;++i)
    {
        if(n%i==0)
        {
            ret-=ret/i;
            while(n%i==0)n/=i;
            if(n==1)break;
        }
    }
    if(n!=1)ret-=ret/n;
    return ret;
}
```

例题

- 定义法雷数列(Farey Sequence)

- F_2 = {1/2}
  F_3 = {1/3, 1/2, 2/3}
  F_4 = {1/4, 1/3, 1/2, 2/3, 3/4}
  F_5 = {1/5, 1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5}

- 既对于F_i来说，实际上是F_(i-1) 添加 若干个以i为分母，和i互质的数为分子的分数

- 求F_n的长度

- 可是面对 n<=10^6的规模，暴力计算欧拉函数显然力不从心

- 是否能高效的产生[1..n]内的所有欧拉函数呢？答案是肯定的

```
//求区间内Eular,1..N
//顺便还产生了1..N内素数-_-if(phi[i]==i-1) so i is a prime...
#define N 1500001
int phi[N];
void mkphilist()
{
    int i,j;
    phi[1]=1;
    for(i=2;i<N;++i)
     if(!phi[i])
      for(j=i;j<N;j+=i)
      {
            if(!phi[j])
                phi[j]=j;
            phi[j]-=phi[j]/i;
      }
}
```

最大公约数

- 最大公因子（Greatest Common Divisor，简写为G.C.D.；或Highest Common Factor，简写为H.C.F.），指某几个整数共有因子中最大的一个。

- GCD(12,4) = 4
- GCD(12,8) = 4
- GCD(12,7) = 1
- ……

- 同时,a和b的最大公约数经常写为 (a,b)

最小公倍数

- 两个整数公有的倍数称为它们的公倍数，其中最小的一个正整数称为它们两个的最小公倍数（*least common multiple*）。

- LCM(12,8) = 24
- LCM(12,4) = 12
- LCM(12,7) = 84
- ……

欧几里得算法

- 定理：gcd(a,b) = gcd(b,a mod b)
- 证明：a可以表示成a = kb + r，则r = a mod b
- 假设d是a,b的一个公约数，则有
- d|a, d|b，而r = a - kb，因此d|r
- 因此d是(b,a mod b)的公约数
- 假设d 是(b,a mod b)的公约数，则
- d | b ， d |r ，但是a = kb +r
- 因此d也是(a,b)的公约数
- 因此(a,b)和(b,a mod b)的公约数是一样的，其最大公约数也必然相等，得证

扩展欧几里德算法

- **扩展欧几里德定理**
- 对于不完全为 0 的非负整数 a，b,必然存在整数对 x，y，使得 (a,b) =ax+by。

下面是一个使用C++的实现：

```
int exGcd(int a, int b, int &x, int &y)
{
    if(b == 0)
    {
        x = 1;
        y = 0;
        return a;
    }
    int r = exGcd(b, a % b, x, y);
    int t = x;
    x = y;
    y = t - a / b * y;
    return r;
}
```

把这个实现和Gcd的递归实现相比，发现多了下面的x,y赋值过程，这就是扩展欧几里德算法的精髓。

- 对于a' = b, b' = a % b 而言，我们求得 x, y使得 a'x + b'y = Gcd(a', b')
  由于b' = a % b = a - a / b * b (注：这里的/是程序设计语言中的除法)
  那么可以得到:
  a'x + b'y = Gcd(a', b') ===>
  bx + (a - a / b * b)y = Gcd(a', b') = Gcd(a, b) ===>
  ay +b(x - a / b*y) = Gcd(a, b)
  因此对于a和b而言，他们的相对应的x’，y’分别是y和(x-a/b*y)

求解模线性方程 $ax = b(\mod n)$

- $ax = b + kn \rightarrow ax - kn = b \rightarrow ax + (-k)n = b$
- 由**扩展欧几里德定理**，如果 不存在$(a,n) \mid b$,那么方程无解
- **如 $2x = 1 \pmod 4$ 必然无解**
- **否则可以转化为 $b = c * (a,n)$,之后套用扩展欧几里德算法求解**

一个例子

- $3x = 9 (\mathrm{mod}\ 10)$
- 转化为 $3x + 10y = 9$
- 我们知道 $3x + 10y = 1$ 必然有解，于是通过扩展欧几里德算法求解，例如求得 $x = -3 , y = 1$
- 那么$3x + 10y = 9$ 的一个解为:$x = -3 * 9 = -27, y = 9$
- 那么可以得到$x = -27$为一个解,于是取模以后得到结果
- $x = 3 (\mathrm{mod}\ 10)$

- http://acm.pku.edu.cn/JudgeOnline/problem?id=2115

ax = b(mod n)解的个数

- 在[0..n- 1]内,存在 (a,n)个解
- ax + kn = b
- 假设得到了一个可行解X1, 满足 aX1 = b (mod n)
- (X1 + n / (a,n)) mod n ,(X1 + 2*n/(a,n))mod n, … (X1 + ((a,n) - 1) * n / (a,n)) mod n必然也为解
- aX1 = b(mod n) ,a(X1 + k * n / (a,n)) = b + 0 (mod n)

- a * k * n / (a,n) mod n == 0

- n / (a,n)　　--n中消去了a,n同时含有的因子
- a=a’ * (a,n)
- So: a * k * n / (a,n) mod n = a’ * k * n (mod n) = 0

模方程组

- 解决求得到最小的x,满足
- A1 * x = b1 (mod c1)
- A2 * x = b2 (mod c2)
- …..
- …..
- Am * x = bm (mod cm)

- 公元前后的《孙子算经》中有“物不知数”问题：“今有物不知其数，三三数之余二，五五数之余三，七七数之余二，问物几何？”答为“23”。也就是求同余式组x≡2（mod3），x≡3（mod5），x≡2（mod7）（式中a≡b（modm）表示m整除a－b）的正整数解。

中国剩余定理(CRT)

- 要求 任意2个 I,j 满足 (Ci,Cj) = 1

模方程组

- 给定方程

$x = c1 \pmod{b1}$ ……(1)

$x = c2 \pmod{b2}$ ……(2)

(b1,b2)可以不为1

于是通过取mod 定义，我们得到

$x = k1 * b1 + c1$……(3)

(3) 带入(2)

k1 * b1 + c1 = c2 (mod b2)……(4)

化简

k1 * b1 = c2 - c1 (mod b2)……(5)

于是可以解得到

令G = gcd(b1,b2),C = c2 - c1 (mod b2)

那么由(5)得到

k1 * b1 = W * b2 + C

---->>>>>

k1 * b1 / G = W * b2 / G + C / G

令C' = C/G

k1 * b1 / G = W * b2 / G + C '

k1 * b1 / G = C' (mod b2 / G)

--->

k1 = K (mod b2/G)……(6)

那么有

k1 = k' * b2/G + K……(7)

(7)带入(3)

x = k' * b2 * b1/G + K * b1 + c1……(8)

x = K*b1 + c1 (mod b1 * b2/G)

一些题目

- http://acm.fzu.edu.cn/problem.php?pid=1402

乘法逆元

- 若对于数字A,C 存在X
- A * X = 1 (mod C) ,那么称X为 A 对C的乘法逆元

- 逆元的作用？ 让我们来看下面的例子:

- 12 / 4 mod 7 = ? , 很显然结果是3
- 我们现在对于数对 (4,7), 可以知道 X = 2是 4 对7的乘法逆元
- 那么我们有
- (12 / **4**) * (**4** * 2) = (?) * (1) (mod 7)
- 除法被完美地转化为了乘法

- 理论依据:
- F / A mod C = ?
- 如果存在 A*X = 1 (mod C)
- 那么2边同时乘起来,得到
- F * X = ? (mod C)

成立条件

- (1) 模方程 A * X = 1(mod C) 存在解
- (2) A | F (F % A == 0)

- http://61.187.179.132:8080/JudgeOnline/showproblem?problem_id=1856

a^b mod c

- (1) 当a,b,c 都比较小的时候，可以使用赤裸裸的暴力

伪代码:

```
v:=1;
for i := 1 to b begin
v:=v*a;
v:=v mod c;
end
```

这也是我们在第一次遇到这种问题的时候首先能想到的.

- 2.a,b,c都比较大的时候
  这里需要考虑c的大小了，假设c*c<2^64吧(再大就只能再做特殊处理了)
  b比较大的话使用1的方法显然时间上是承受不了的，所以可以利用所谓的二分法.
  b=b0+b1*2^1+b2*2^2+...+bn*2^n
  显然a^b可以变成若干项的乘积

Power_mod

- 伪代码:

```
v:=1;
while b<>0 do begin
if (b and 1= 1) do begin v:=v*a;v:=v mod c;end
a:=a*a;
a:=a mod c;
b:=b shr 1;
end
```

- 3.如果c很大，那么需要使用到模拟乘法来避免溢出,当然如果c已经大到10^100,那么只能用高精了..这里只能处理最多c=2^64-1的情况吧

伪代码:

```
int mul(int a,int b,int c)
{
int ret=0,tmp=a%c;
while(b)
{
   if(b&0x1)if((ret+=tmp)>=c)ret-=c;
   if((tmp<<=1)>=c)tmp-=c;
   b>>=1;
}
return ret;
}
```

指数循环

- 观察 2^i mod 7
- 可以得到数列
- 1,2,4,1,2,4,1,2,4……

- 循环永无止境

- 那么计算 2^ 1000000000 mod 7 ?

为什么会出现循环

- 可知对于mod p的结果必然在区间[0,p-1]内
- 那么假设乘了p次,则由鸽巢原理不难得到循环必然出现

- 第p次出现的结果必然和前面的某次结果相同，并在此后无限循环…

一些例子

- 5^i mod 7: 1,5,4,6,2,3,1,5,4,6,2,3,1,5,4,6,2….
- 7^i mod 7: 0,0,0,0,0,0,0,0,0,0,0,0,0,0….
- 4^i mod 10: 1,4,6,4,6,4,6……
- 可以发现循环的开始位置不一定是从头开始……

关于循环的一些定理

- 定理 1:
  对于一个数对(A,C) 必然存在一个最小的正整数 L, 满足

$$A^x = A^{(x+L)} \bmod C \ (x \geq SPOS)$$

其中SPOS 是一个大于等于0的整数(下面具体介绍)
我们称L 为(A,C) 的最小循环节长度

证明:
根据鸽巢原理,得到在x >= C 后必然出现循环,从而定理得证.

- 定理 2:
  对于数对 (A,C) 下面的公式必然成立

$$A^x = A^{(x+k*L)} \bmod C \ (x \geq SPOS)$$

其中 k >= 0
既L 的任意倍数均为一个新的循环节长度.
证明:
根据定理1,不难得证.

如何实现找循环？

- 开一个数组记录即可

- 假设寻找 a^I mod c 的循环
- int num = 1;
- For(i = 0; ; ++ i){
  - Num = num * a % c;
  - If(hash[num] == 1) break; // 找到循环
- }

- http://acm.fzu.edu.cn/problem.php?pid=1759

- 可是如果取模的数太大，暴力找循环必然导致TLE
- A^x = A^(x % Phi(C) + Phi(C)) (mod C)
- 证明比较复杂，可以参见这里

- 自此，对于部分
- A^10000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000
  00000000000000…..
- Mod P
- 的问题，可以解决！

若干其他循环节

- (1) 斐波那契数列 对于mod m存在循环节
- (2) 错排数列队伍mod m 存在循环节
- (3) n! mod m存在循环节
- ……
- ……

- (1)http://acm.hdu.edu.cn/showproblem.php?pid=2814
- (2) http://acm.fzu.edu.cn/problem.php?pid=1944

离散对数

- 离散对数是在整数中，一种基于同余运算和原根的一种对数运算。

原根

- 在数论，特别是整除理论中，原根是一个很重要的概念。对于两个正整数(a,m) = 1，由欧拉定理可知，存在正整数， 比如说欧拉函数d= φ(m)，即小于等于m的正整数中与m互质的正整数的个数，使得a^d = 1(mod m)。由此，在(a,m) = 1时，定义a对模m的指数Ordm(a)为使a^d = 1(mod m)成立的最小的正整数d。由前知Ordm(a)一定小于等于 φ(m)，若Ordm(a) = φ(m)，则称a是模m的原根。

- 设$m = 7$，则等 于6。
- 设$a = 2$，由于，2^3 = 1(mod 7) 而， 所以 2 不是模 7 的一个原根。
- 设$a = 3$，由于3^6 = 1 (mod m)， 因此有， 所以 3 是模 7 的一个原根。

- 可以证明，如果正整数$(a,m) = 1$和正整数 d 满足 a^d = 1(mod m)， 则 d 整除 $\phi(m)$。因此$Ord_m(a)$整除 $\phi(m)$。在例子中，当$a = 3$时，我们仅需要验证 3 的 1 、2、3 和 6 次方模 7 的余数即 可。

Baby-Step Giant-Step

- 【问题模型】

  求解

  A^x = B (mod C) 中 0 <= x < C 的解,C 为素数

【思路】

我们可以做一个等价

x = i * m + j (0 <= i < m, 0 <=j < m) m = Ceil (sqrt(C))

而这么分解的目的无非是为了转化为:

(A^i)^m * A^j = B (mod C)

之后做少许暴力的工作就可以解决问题:

(1) for i = 0 -> m, 插入Hash (i, A^i mod C)

(2) 枚举 i ,对于每一个枚举到的i,令 AA = (A^m)^i mod C

- 我们有

AA * A^j = B (mod C)

显然AA,B,C均已知，而由于C为素数,那么(AA,C)无条件为1

于是对于这个模方程解的个数唯一(可以利用扩展欧几里得或 欧拉定理来求解)

那么对于得到的唯一解X,在Hash表中寻找,如果找到，则返回 ***i * m + j***

注意:**由于i从小到大的枚举,而Hash表中存在的j必然是对于某个剩余系内的元素 X 是最小的(就是指标)**

所以显然此时就可以得到最小解

- 如果需要得到 $x > 0$的解，那么只需要在上面的步骤中判断 当 $i * m + j > 0$ 的时候才返回

到目前为止，以上的算法都不存在争议,大家实现的代码均相差不大。可见当C为素数的时候，此类离散对数的问题可以变得十分容易实现。

- 例子

- 求最小的 x,满足 7^x = 12 (mod 13)

(1) sqrt(13) = 3.60555127546398929311922126747 05

我们为了保险起见，选择 4

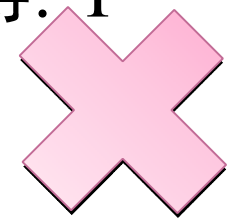
枚举

- 7^0 = 1(mod 13) **保存数对 (0,1)**
- 7^1 = 7 (mod 13) **保存数对 (1,7)**
- 7^2 = 10(mod 13) **保存数对 (2,10)**
- 7^3 = 5(mod 13) **保存数对 (3,5)**
- 7^4 = 9 (mod 13) **保存数对 (4,9)**
- 到这里我们做了 **[sqrt(13)] 次**的枚举

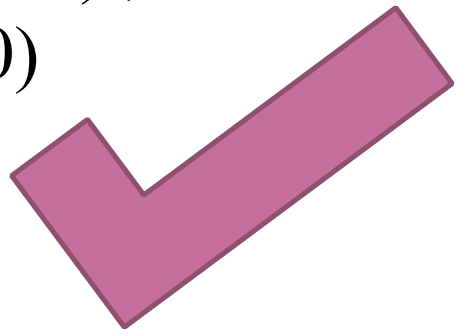
(3) 计算 7^4 = 9 (mod 13)

(4) 枚举 I

i = 0: 9^0 = 1 (mod 13) , 1 * Y = 12 (mod 13) ,解得: Y = 12(mod 13), 寻找是否存在 数对 (*, 12)



i = 1: 9^1 = 9 (mod 13) , 9 * Y = 12 (mod 13) ,解得: Y = 10(mod 13), 寻找是否存在 数对 (*, 10)



(2, 10)

So: x = 1 * m + 2 = 1 * 4 + 2 = **6**

- http://acm.fzu.edu.cn/problem.php?pid=1493

*扩展Baby Step Giant Step

- 此部分难度系数不小，有兴趣的可以看这里

- http://acm.hdu.edu.cn/showproblem.php?pid=2815

数论运用_组合数求模

- N比较小的时候
  暴力递推或暴力计算(1到M关于MOD有逆)

```
int C[N][N];
for (i=0;i<N;i++)
for (j=0;j<=i;j++)
if (j==0 || j==i) C[i][j]=1%MOD;
else C[i][j]=(C[i-1][j]+C[i-1][j-1])%MOD;
```

- N稍微大的时候

筛素数,分解成素数表示+power_mod

即C(N,M）＝N!/(N-M)!/M! 可以统计C(N,M)中小于等于N的素数的出现次数，具体就是加上N!，减去(N-M)!,M!中的素数。

算N!中某个素数p的出现次数，可以简单得写：

```
cnt=0;
while (N)
{
    cnt+=N/p;
    N/=p;
}
```

最后用power_mod会比较合适

- N很大,M很小(1到M关于MOD有逆)
  C(N,M)=∏(N-M+i)/i=∏(N-M+i)*inv(i)

- Inv(i) 表示 i对 MOD的乘法逆元

MOD 为 Square-Free-Number

- Square-Free-Number
- 不存在任意一个 平方数的因子 (除了1)
- 比如12 = 2 * 2 * 3， 不是 Square-Free-Number
- 6 = 2 * 3， 是Square-Free-Number

- 分解MOD = P1 * P2 * .. * PK
- 分别计算 C(n,k) mod P1, C(n,k) mod P2…
- 之后利用CRT合并得到结果 C(n,k) mod MOD

- 我们知道前面的做法是基于 对于任意的 I, (I,MOD) = 1的前提
- If (I,MOD) != 1?
- C(5,2) mod 4.?

- 对于MOD,我们分解素因子
- MOD = p1^a1 * p2^a2 * … * pk^ak

- 可以知道如果对于分母中的某个I,(I,MOD)!=1
- 那么必然存在j,使得 Pj | (I,mod)

- 于是如果完全“无视” 这些 MOD 的因子，那么乘法逆元必须存在

- 这使得我们容易想到： 对于 MOD的这些素因子，我们单独保存它们的个数 (特别服务)

- 之后其他部分使用逆元即可！

例子

- C(5,3) mod 4 = ?
- (1) 分解 4 ， 4 = 2^2
- (2) 求出 5!中2的个数: 5/2 + 2 / 2 = 3个
- 求出 2!中2的个数 ： 1个，求出3!中2的个数:1个
- 于是C(5,3)中存在1个因子2

- 那么对于消灭了因子2以后
- (1 * 3 * 5) / (3) mod 4
- 3对于4的逆元为3 so
- Ans0 = 1 * 3 * 5 * 3 mod 4 =1

- Ans0是剩余部分的结果
- 我们再根据2的个数为1
- 得到 Ans = Ans0 * pow_mod(2, 1 , 4) = 1 * 2 mod 4 = 2…

- So C(5,3) = 2 (mod 4)

- 更多细节，请参考 这里

推荐题目

- 1.素数的筛选

  http://acm.fzu.edu.cn/problem.php?pid=1607

- 下面这题也用到了素数筛选(素数筛选实在太常用了,这里只给比较经典的题目)
  http://acm.fzu.edu.cn/problem.php?pid=1753

- 2.模方程问题
  http://acm.pku.edu.cn/JudgeOnline/problem?id=1061

- 3.模方程组问题
- http://acm.fzu.edu.cn/problem.php?pid=1402

- 4.循环节
- fibonacci数列的性质
  f(a^b) %c 的循环节开始位置居然都是1(神奇..)
  http://acm.hdu.edu.cn/showproblem.php?pid=2814(**
)

- 5. 离散对数
- http://acm.fzu.edu.cn/problem.php?pid=1493
- http://acm.hdu.edu.cn/showproblem.php?pid=2815(**)
- http://acm.zju.edu.cn/onlinejudge/showProblem.do?problemCode=3254 （***）

- 6.组合数求模
- http://acm.fzu.edu.cn/problem.php?pid=1775 （*）

The end

Q & A