**素数**：

* **一个大于1，且只能被1和它本身整除的正整数, 称为素数; 否则称为合数.**
* **由此可知，正整数集合可分为三部分: 素数、合数 和 1**
* **一些性质**
  1. **素数的个数是无穷的**
  2. **除2以外的素数一定是奇数，也称为奇素数**
  3. **任意两个相邻的正整数n和n＋l(n＞3)中必有一个不是素数**
  4. **相邻整数均为素数的只有2和3**

**素因子：**

**若b|a，且b是素数, 称b是a的素因子**

**例：**

**12=3×4, 3是12的素因子, 2也是12的素因子，而4不是（3|12，2|12）**

## 整数分解的唯一性定理

## 定理：

**任意一个正整数 a>1 总可以分解为一系列素数乘积的形式，而且分解形式是唯一的**

**a＝p1a1 × p2a2 × p3a3 × …… × pnan**

**例：**

**36＝22×32**

**3600＝24×32×52**

**Q如何生成素数?**

**现代密码学，特别是公钥密码学，常用随机的大素数,习惯上，常用符号 p或 q 表示素数**

**Q: 如何生成一个随机的大素数?**

**先生成一个随机数，然后将之分解得到其素因子，从而得到素数，这种方法是否可行？**

**不行！**

**就目前而言，对某些大整数进行素因子分解是计算上不可行的**

**生成素数的正确方法 —— 素性检测：随机产生一个大奇数，然后检测它是否是素数**

**在诸多素性检测算法中，Miller-Rabin算法 是容易实现且广泛使用的算法**

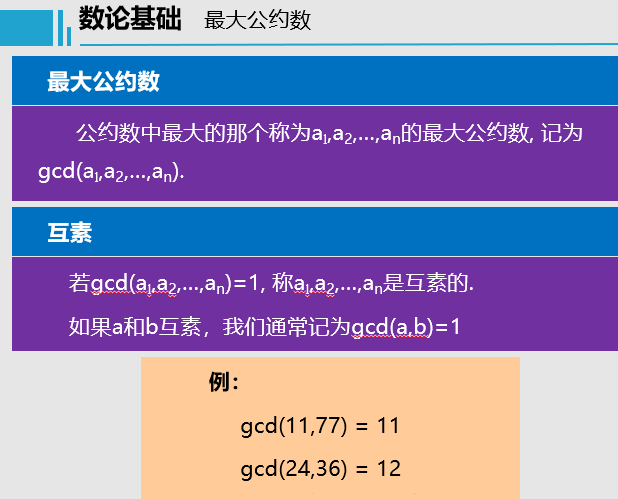
**随机素数的生成过程**

1. 随机产生一个n比特的随机数p (如通过伪随机序列发生器)。
2. 将p的最高位和最低位设置为1 (最高位设置为1的目的是确保素数达到最大有效长度，最低位设置为1的目的是确保该数是奇数)。
3. 检查p不能被所有小于2000的素数整除 (有方法可使这一步做得很快)。
4. 随机选择a, 且a<p。
5. 用a对p进行素性测试(如用Miller-Rabin算法)。若p没有通过测试,抛弃p,转到 ① (或将p加2作为新的p,然后转到③)。
6. 如果通过测试,转到④。如果p通过足够次数的测试(如5次)，认为p是素数。

**素性检测算法是概率算法，会有一定的错误概率**

* + 它永远不会把素数误判为合数，却可能把合数误判为素数，但概率很低
  + 对一个随机数重复进行多次素性测试，误判的概率会比你中六合彩还低
  + 因此，误判这个问题我们根本不必担心

**公约数(公因子)：设al,a2,…,an和d都是正整数(n≥2). 若d|ai (1≤i≤n), 则称d是al,a2,…,an的 公约数(公因子).**



公约数中最大的那个称为al,a2,…,an的最大公约数, 记为gcd(al,a2,…,an).

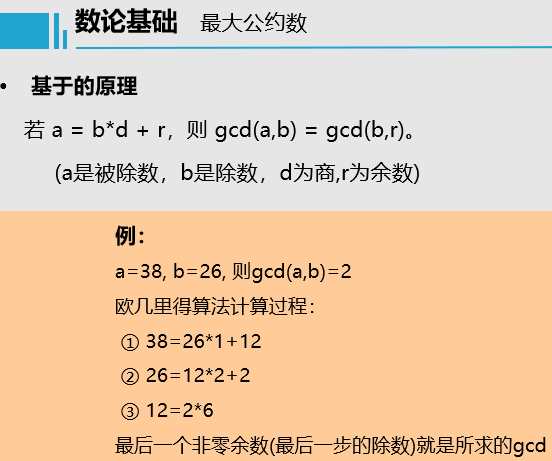
**最大公约数的性质**

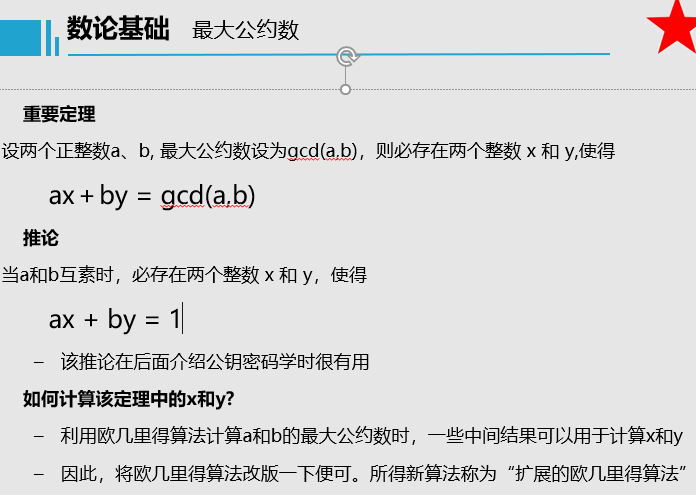
* 在互素的正整数中, 不一定有素数.
  1. 例：gcd(25,36)=1, 但25和36都不是素数.
* 在个数不少于3个的互素正整数中, 不一定两两互素.
  1. 例: gcd(6,10,15)= 1,

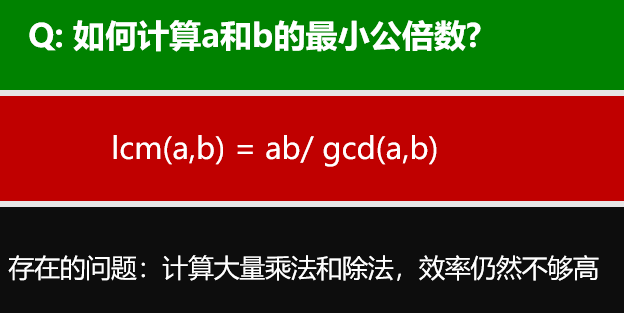
但 gcd(6,10)=2,

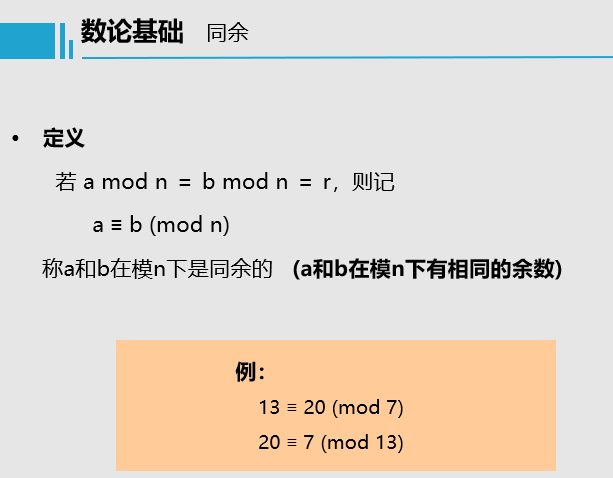
gcd(6,15)=3,

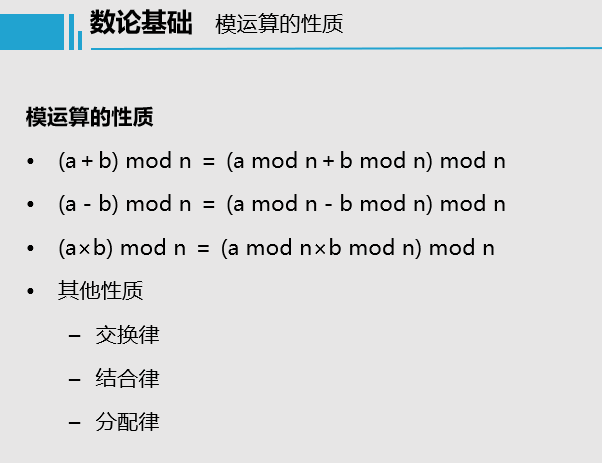
gcd(10,15)=5.







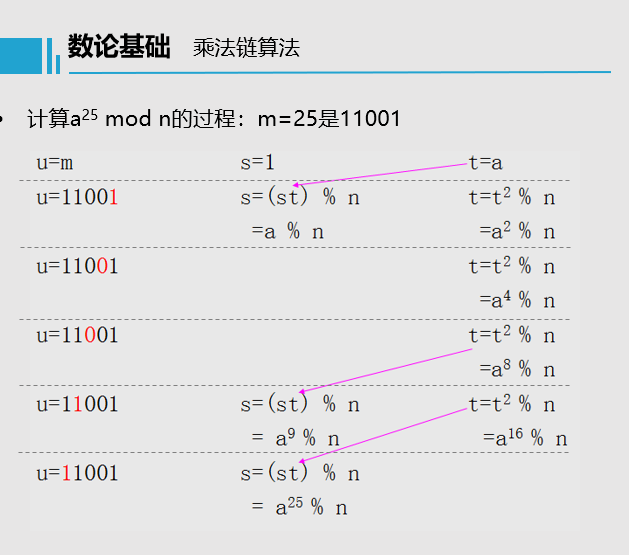


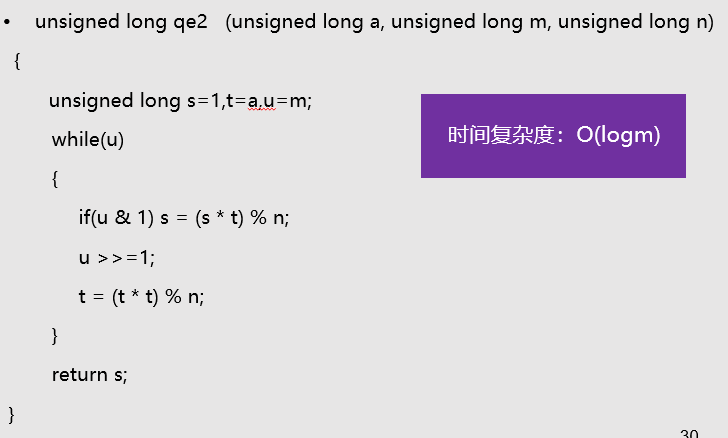


**乘法链运算：**

**Q如何计算am mod n？**

**A 将m化为二进制数**





**欧拉函数的计算结果：小于等于n且与n互素的正整数的个数。**

**欧拉函数的性质**

* **性质：**若p、q都是素数，则 *φ*(p) = p – 1 ，*φ*(p×q) = *φ*(p)*φ*(q) = (p - 1)(q - 1)
* **定理：**若p为素数，k为正整数，则 *φ*(pk)＝pk-1(p－1)
* **推论：**

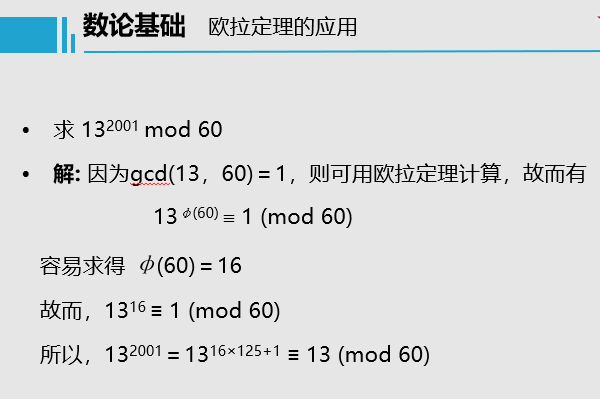
若gcd(a,b)＝1，则 *φ*(ab)＝*φ*(a)*φ*(b)

* **欧拉定理** 设n≥2，如果gcd(a,n)=1，则 a*φ*(n) ≡ 1 (mod n)，

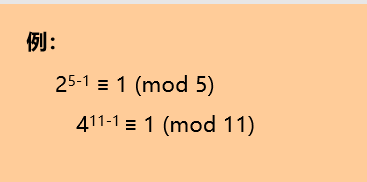
同理有 a*φ*(n)＋1 ≡ a (mod n)

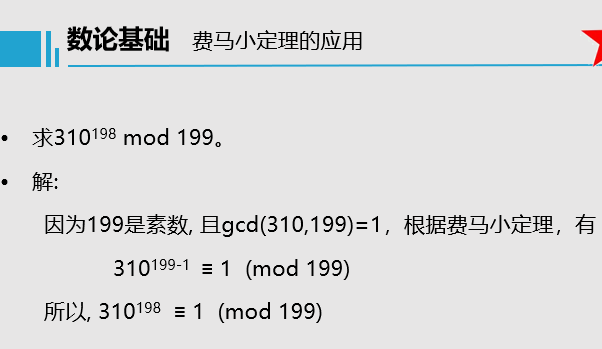
容易求得 *φ*(60)＝16

**Q(60)=Q(4)xQ(15)=2xQ(3)\*Q(5)=2x2x4;**



* **费马小定理：** 若p是素数, gcd(a,p)=1,则ap-1 ≡ 1 (mod p)同理有ap ≡ a (mod p)





二次剩余：

* **定义：**

设a是小于n(n>1)的正整数，且gcd(a,n)=1。如果存在x，使

x2 ≡ a (mod n)

成立，那么称a是模n下的二次剩余。



二次剩余的特性

* **性质1：**二次剩余的判定标准

如果 a(p-1)/2 ≡ 1 (mod p)，其中p>2是素数，1≤a<p

则 a∈ QRp

* **性质2：**当模数是素数时，设为p>2
  + 模p下的二次剩余恰有(p-1)/2个，与其二次非剩余的数目相同(各占一半)
  + 如果a=x2(mod p)是二次剩余，那么a恰好有两个平方根。一个是x,另一个是p-x。

## 群，循环群的几个性质

* 任何循环群必是交换群
* 设有限循环群的阶为n，则
  1. 生成元的阶也是n，其他元素的阶必是n的因子
  2. 生成元的个数为*φ*(n)
     + 例循环群中有15个元素，那么生成元的个数为*φ*(15)=*φ*(3)*φ*(5)=2\*4=8
  3. 阶为m的元素的个数共有*φ*(m)个

密码学中的常用群

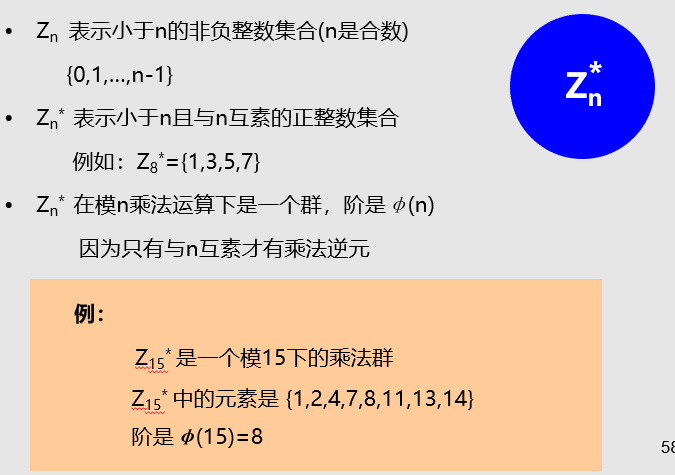
* Zp 表示小于素数p的非负整数集合

{0,1,2,…,p-1}

* Zp\* 表示小于素数p且与p互素的正整数集合

{1,2,…,p-1}

* Zp\* 在模p乘法运算下是一个循环群
* **相关性质：**
  + 单位元是1
  + 群的阶是p-1，生成元的个数是*φ*(p-1)
    - 例：Z7\*是一个循环群，生成元共有*φ*(6)=2个，分别是3和5。



## 域 有限域

* 若域F是有限的，称F为有限域，又称伽罗瓦域
* **定理1**
  + 有限域的元素个数必为pn 的形式(p为素数)，记为
* **定理2**
  + 有限域的乘法群是循环群
* **定理3**
  + 同阶(元素数目相同)的有限域都是同构的
* 密码学中，我们关注Fp(又称素数域)和
* **信息论告诉我们**
  + 除一次一密以外，任何密码都是可以破解的。
* **计算复杂性理论告诉我们**
  + 在宇宙爆炸前，它们是否可以被破解。而破解它们所花费的时间和空间是否已超出你所能承受的底线。

**现代密码学将安全性构建在计算复杂性理论之上**

* 公钥密码学以目前的计算方法不能有效解决的问题为构造基础，这些问题被称为“困难问题”。
* 密码体制的安全性就在于困难问题的困难性

**计算复杂性** 问题复杂性分类

* **问题本身有着内在固有的复杂性**，这与解决问题的算法的复杂度是不同的。
* 计算复杂性理论研究的主要内容：对问题的复杂性进行分类。
* 所用的工具便是图灵机

计算复杂性P问题

* **P问题**
  + 存在一个确定性图灵机，可以在多项式时间内解决的问题
* 通常认为，多项式时间算法是有效算法，多项式时间内可解决的问题是容易问题 (**P问题是容易问题**)
  + 但这种想法是不精确的。当k很大时，例如 k=200，即使n很小，例如n=2,3，O(n200)也非常大
  + 但多项式时间算法毕竟比指数和超多项式时间算法快得多,因此这种说法是可以接受的

计算复杂性NP问题

* **NP问题**
  + 存在一个非确定性图灵机，可以在多项式时间内解决的问题。
* 也就是说，NP问题用非确定性图灵机很容易解决。
* 因为在确定性图灵机上未发现多项式时间的算法，通常认为NP问题是难以解决的，俗称 **困难问题**
* **P问题也属于NP问题的范畴**
  + 因为确定性图灵机上在多项式时间内可以解决的问题，在非确定性图灵机上用多项式时间也可以解决。（把猜测阶段省略掉即可）
* **但我们一般所说的困难问题或NP问题，通常不包括P问题**。

计算复杂性

计算复杂性NPC问题

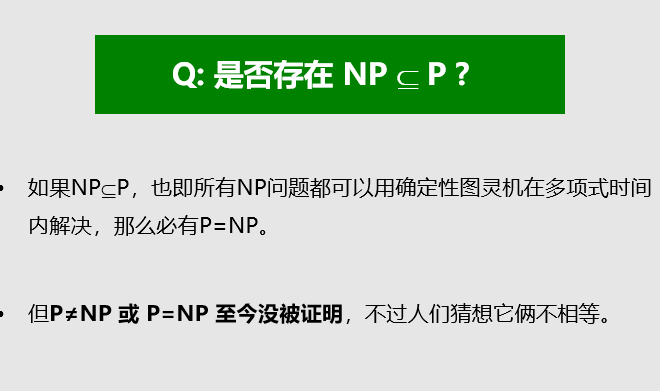
* NP中有一些特殊问题，如果找到一个确定的多项式时间算法可以解决这些问题中的一个，解NP中的任何问题都能找到多项式时间算法。
* 这类问题被称作NP完全问题，记为**NPC**.

计算复杂性：P与NP的关系

* + 由前述可知

P ⊆ NP

* + 任何在确定性图灵机上在多项式时间内可以解决的问题，在非确定性图灵机上在多项式时间内同样可以解决



**“P是否等于NP”是计算复杂性理论未解决的中心问题**

