



现代密码学

林喜军

linxj77@163.com



第5章

现代密码学的 理论基础

现代密码学用到的主要理论知识

概率论

信息论

} 分析安全性的基本工具 (略)
(发生不安全事件的可能性有多大)

数论

抽象代数

} 基本构造工具

计算复杂性
理论

—— 理论基础

5.1 数论基础

5.2 抽象代数

5.3 计算复杂性理论

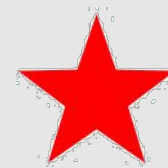


5.1 数论基础

数论

研究整数的性质

现代密码学中主要利用整数的性质



- 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
 - 素数的个数是无穷的
 - 除2以外的素数一定是奇数，也称为奇素数
 - 任意两个相邻的正整数 n 和 $n + 1$ ($n > 3$)中必有一个不是素数
 - 相邻整数均为素数的只有2和3



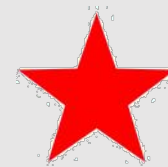
素因子：

若 $b|a$ ，且 b 是素数，称 b 是 a 的素因子

例：

$12=3\times 4$ ，3是12的素因子，2也是12的素因子，而4不是

整数分解的唯一性定理



- **定理：**

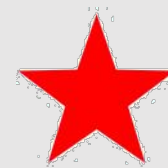
任意一个正整数 $a > 1$ 总可以分解为一系列素数乘积的形式，而且分解形式是唯一的

$$a = p_1^{a_1} \times p_2^{a_2} \times p_3^{a_3} \times \dots \times p_n^{a_n}$$

例：

$$36 = 2^2 \times 3^2$$

$$3600 = 2^4 \times 3^2 \times 5^2$$



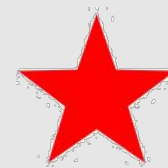
现代密码学，特别是公钥密码学，常用随机的大素数
习惯上，常用符号 p 或 q 表示素数

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！

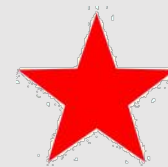
就目前而言，对某些大整数进行素因子分解是计算上不可行的



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

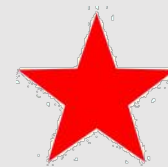
在诸多素性检测算法中，Miller-Rabin算法 是容易实现且广泛使用的算法



- 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。

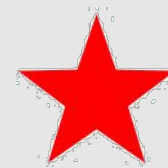
- **基于的原理**

- 在整数 n 附近，约每 $\ln(n)$ 个整数中就有一个素数
 - 事实上，在长度为512比特的整数中，约有 10^{151} 个素数
 - 素数的密度相当可观，因此这种概率检测法是很实用的
- 在实际执行算法时，生成素数是很快的
(上述过程通常可在几分钟,甚至几秒钟内生成一个素数)



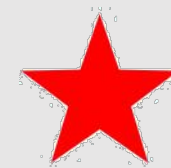
素性检测算法是概率算法，会有一定的错误概率

- 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 a_1, a_2, \dots, a_n 和 d 都是正整数($n \geq 2$). 若 $d|a_i$ ($1 \leq i \leq n$), 则称 d 是 a_1, a_2, \dots, a_n 的公约数(公因子).



最大公约数

公约数中最大的那个称为 a_1, a_2, \dots, a_n 的最大公约数, 记为 $\gcd(a_1, a_2, \dots, a_n)$.

互素

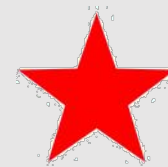
若 $\gcd(a_1, a_2, \dots, a_n) = 1$, 称 a_1, a_2, \dots, a_n 是互素的.

如果 a 和 b 互素, 我们通常记为 $\gcd(a, b) = 1$

例 :

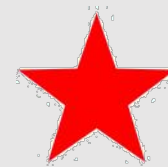
$$\gcd(11, 77) = 11$$

$$\gcd(24, 36) = 12$$



最大公约数的性质

- 在互素的正整数中, 不一定有素数.
 - 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.
 - 例: $\gcd(6, 10, 15) = 1$,
但 $\gcd(6, 10) = 2$,
 $\gcd(6, 15) = 3$,
 $\gcd(10, 15) = 5$.

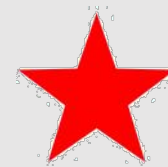


计算GCD的算法——欧几里得算法

- 又称辗转相除法
- 记于公元前300年欧几里得所著名为Elements的书中
- 历史学家们相信，该算法并非欧几里得发明，早在此前200年该算法就已经出现。
- 它是幸存到现在的最古老的非凡算法，至今它仍是完好的。



Euclid



- 基于的原理

若 $a = b*d + r$, 则 $\gcd(a,b) = \gcd(b,r)$ 。

(a是被除数, b是除数, d为商,r为余数)

例：

$a=38, b=26$, 则 $\gcd(a,b)=2$

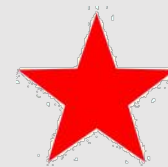
欧几里得算法计算过程：

① $38=26*1+12$

② $26=12*2+2$

③ $12=2*6$

最后一个非零余数(最后一步的除数)就是所求的gcd



- **重要定理**

设两个正整数 a 、 b , 最大公约数设为 $\gcd(a,b)$, 则必存在两个整数 x 和 y ,使得

$$ax + by = \gcd(a,b)$$

- **推论**

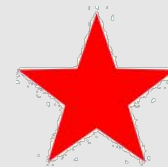
当 a 和 b 互素时, 必存在两个整数 x 和 y , 使得

$$ax + by = 1$$

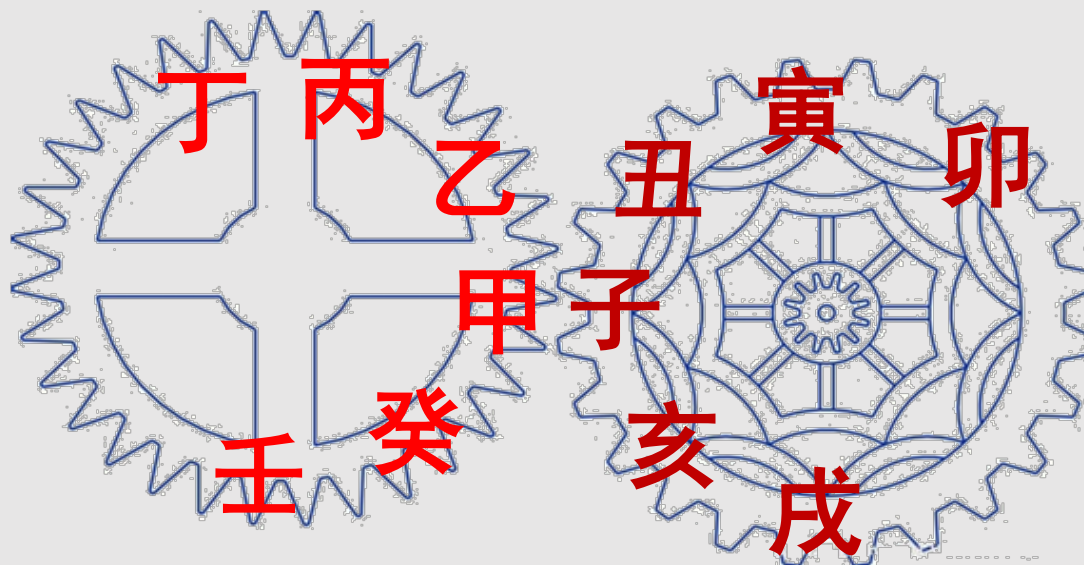
- 该推论在后面介绍公钥密码学时很有用

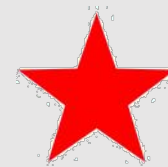
- **如何计算该定理中的 x 和 y ?**

- 利用欧几里得算法计算 a 和 b 的最大公约数时, 一些中间结果可以用于计算 x 和 y
- 因此, 将欧几里得算法改版一下便可。所得新算法称为“扩展的欧几里得算法”



- 六十年一甲子

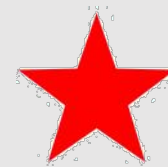




Q: 如何计算a和b的最小公倍数?

$$\text{lcm}(a,b) = ab / \text{gcd}(a,b)$$

存在的问题：计算大量乘法和除法，效率仍然不够高



Q: 如何只用加法计算最小公倍数?

$$\text{lcm}(6,15)=30$$

30

24

18

12

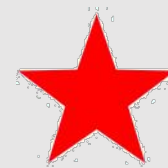
6

30

15

方法：

两者取最小
反复加自己
相等时停止



- 模运算，即为求余数的运算
- 模运算的运算符记为 mod

$$a \bmod n = r$$

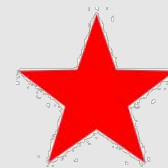
表示 a 除以 n 所得的余数为 r 。

- 上式中的 n 通常称为 **模数**

例：

$$17 \bmod 11 = 6$$

$$8 \bmod 7 = 1$$



- 定义

若 $a \bmod n = b \bmod n = r$, 则记

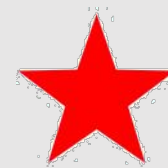
$$a \equiv b \pmod{n}$$

称 a 和 b 在模 n 下是同余的 (**a 和 b 在模 n 下有相同的余数**)

例：

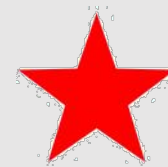
$$13 \equiv 20 \pmod{7}$$

$$20 \equiv 7 \pmod{13}$$



模运算的性质

- $(a + b) \bmod n = (a \bmod n + b \bmod n) \bmod n$
- $(a - b) \bmod n = (a \bmod n - b \bmod n) \bmod n$
- $(a \times b) \bmod n = (a \bmod n \times b \bmod n) \bmod n$
- 其他性质
 - 交换律
 - 结合律
 - 分配律



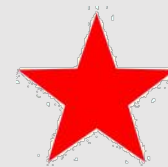
如何计算 $a^m \bmod n$?

不要直接计算 $(a \times a \times \dots \times a) \bmod n$
会导致因中间结果巨大而计算溢出

解决思路

利用模运算的性质简化中间结果
但即使这样，仍然有技巧

如果直接计算 $a \times (a \times \dots \times (a \bmod n) \bmod n) \bmod n$
需要计算 m 次模运算，仍然不够优化

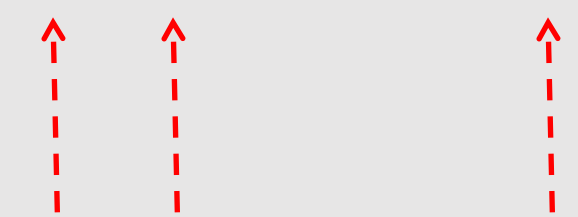


正确思路

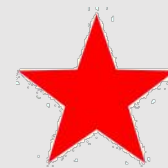
将m看成2的幂次方之和,再利用模运算的性质

计算 $a^{25} \bmod n$ ($m=25$ 是11001)

$$a^{25} = 1 \quad 1 \quad 0 \quad 0 \quad 1$$

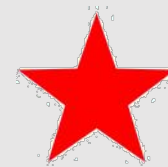


$a^{16} \quad a^8 \quad a^1$



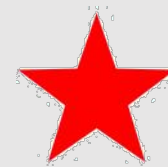
- 计算 $a^{25} \bmod n$ 的过程：m=25是11001

$u=m$	$s=1$	$t=a$
$u=11001$	$s=(st) \% n$ $=a \% n$	$t=t^2 \% n$ $=a^2 \% n$
$u=11001$		$t=t^2 \% n$ $=a^4 \% n$
$u=11001$		$t=t^2 \% n$ $=a^8 \% n$
$u=11001$	$s=(st) \% n$ $=a^9 \% n$	$t=t^2 \% n$ $=a^{16} \% n$
$u=11001$	$s=(st) \% n$ $=a^{25} \% n$	



```
• unsigned long qe2 (unsigned long a, unsigned long m, unsigned long n)
{
    unsigned long s=1,t=a,u=m;
    while(u)
    {
        if(u & 1) s = (s * t) % n;
        u >>=1;
        t = (t * t) % n;
    }
    return s;
}
```

时间复杂度 : $O(\log m)$

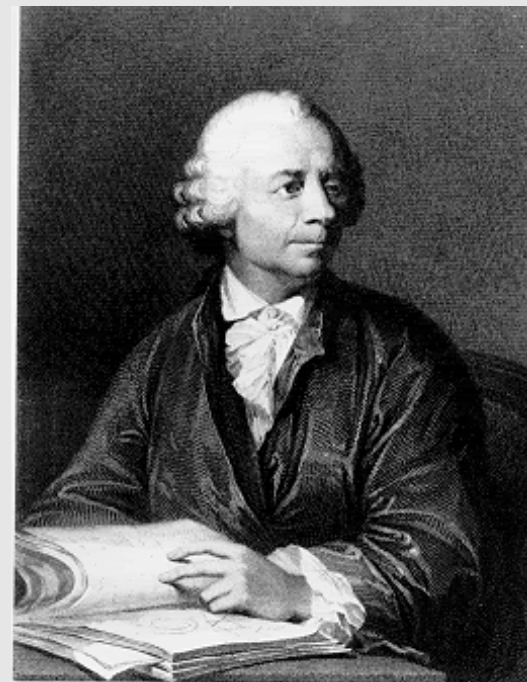


- **其计算结果是**
小于等于 n 且与 n 互素的正整数的个数。

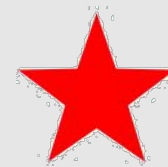
例：

小于等于8且与8互素的正整数有1,3,5,7.

所以 $\phi(8)=4$



Euler
1707—1783



欧拉函数的性质

•性质：若 p 、 q 都是素数，则

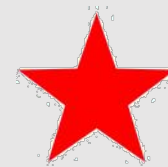
- $\phi(p) = p - 1$
- $\phi(p \times q) = \phi(p) \phi(q) = (p - 1)(q - 1)$

例：

$$n = 15 = 3 \times 5 = p \times q$$

$$\text{则 } \phi(15) = (3 - 1) \times (5 - 1) = 8$$

即 1、2、4、7、8、11、13、14

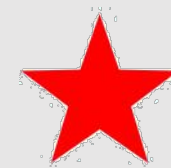


- **定理：**若 p 为素数， k 为正整数，则

$$\phi(p^k) = p^{k-1}(p - 1)$$

例：

$$\phi(8) = \phi(2^3) = 2^2 \phi(2) = 4$$



- 推论：

若 $\gcd(a, b) = 1$, 则

$$\phi(ab) = \phi(a) \phi(b)$$

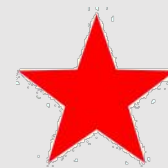
例：

$$\phi(12) = \phi(3) \phi(4)$$

$$\text{而 } \phi(3) = 2, \phi(4) = 2$$

$$\text{所以 } \phi(12) = 4$$

(小于等于12且与12互素的正整数有1、5、7、11总共4个)



- 欧拉定理

设 $n \geq 2$ ，如果 $\gcd(a, n) = 1$ ，则

$$a^{\phi(n)} \equiv 1 \pmod{n},$$

同理有

$$a^{\phi(n)+1} \equiv a \pmod{n}$$

例：

$$n = 15 = 3 \times 5, \text{ 有 } \phi(n) = (3-1)(5-1) = 8$$

$$\text{则 } 4^8, 7^8, 11^8 \equiv 1 \pmod{15}$$



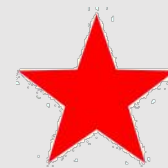
- 求 $13^{2001} \bmod 60$
- **解:** 因为 $\gcd(13, 60) = 1$, 则可用欧拉定理计算, 故而有

$$13^{\phi(60)} \equiv 1 \pmod{60}$$

容易求得 $\phi(60) = 16$

故而, $13^{16} \equiv 1 \pmod{60}$

所以, $13^{2001} = 13^{16 \times 125 + 1} \equiv 13 \pmod{60}$



- 费马小定理：

若 p 是素数, 且 $\gcd(a,p)=1$,则

$$a^{p-1} \equiv 1 \pmod{p}$$

同理有

$$a^p \equiv a \pmod{p}$$

例：

$$2^{5-1} \equiv 1 \pmod{5}$$

$$4^{11-1} \equiv 1 \pmod{11}$$



Fermat
1601—1665



- 求 $310^{198} \bmod 199$ 。

- 解:

因为199是素数, 且 $\gcd(310, 199)=1$, 根据费马小定理, 有

$$310^{199-1} \equiv 1 \pmod{199}$$

所以, $310^{198} \equiv 1 \pmod{199}$

- 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

$$x^2 \equiv a \pmod{n}$$

成立，那么称 a 是模 n 下的二次剩余。

不是所有的 a 都满足这一特性

QR_n	模 n 下二次剩余的集合
QNR_n	模 n 下二次非剩余的集合

- 例：

$n=7$, 二次剩余是1,2,4

- 因为： $1^2 \equiv 1 \pmod{7}$

$$2^2 \equiv 4 \pmod{7}$$

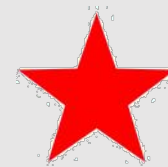
$$3^2 \equiv 2 \pmod{7}$$

$$4^2 \equiv 2 \pmod{7}$$

$$5^2 \equiv 4 \pmod{7}$$

$$6^2 \equiv 1 \pmod{7}$$

- 所以， $QR_7=\{1,2,4\}$; $QNR_7=\{3,5,6\}$



- **性质1**：二次剩余的判定标准

如果 $a^{(p-1)/2} \equiv 1 \pmod{p}$ ，其中 $p > 2$ 是素数， $1 \leq a < p$

则 $a \in QR_p$

- **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 $a = x^2 \pmod{p}$ 是二次剩余，那么 a 恰好有两个平方根。一个是 x ，另一个是 $p-x$ 。

韩信点兵问题：

3人一排余2人

5人一排余3人

7人一排余2人

问共多少人？

孙子歌诀

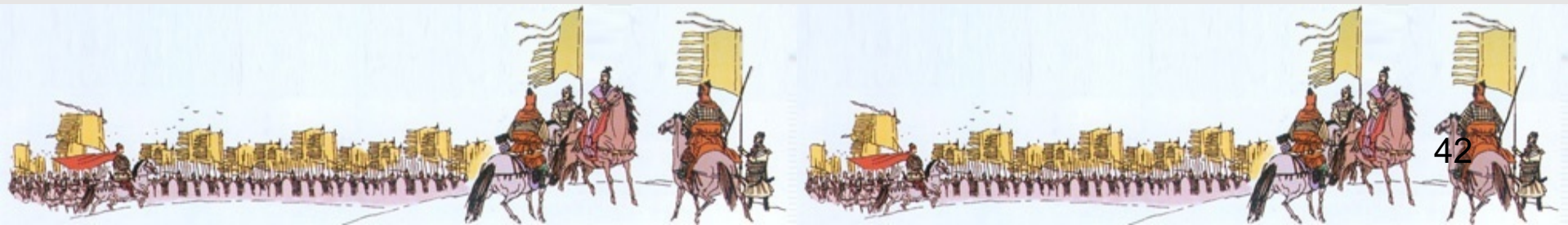
三人同行七十稀，

五树梅花廿一枝。

七子团圆正半月，

除百零五便得知。

$$70 \times 2 + 21 \times 3 + 15 \times 2 = 233 = 2 \times 105 + 23$$



- 中国剩余定理(孙子定理)

设 n_1, n_2, \dots, n_k 是两两互素的正整数, 对于任意整数 b_1, b_2, \dots, b_k , 下列同余方程组

$$\begin{cases} x \equiv b_1 \pmod{n_1} \\ x \equiv b_2 \pmod{n_2} \\ \dots \\ x \equiv b_k \pmod{n_k} \end{cases}$$

$x < n_1 n_2 \dots n_k$ 有唯一解

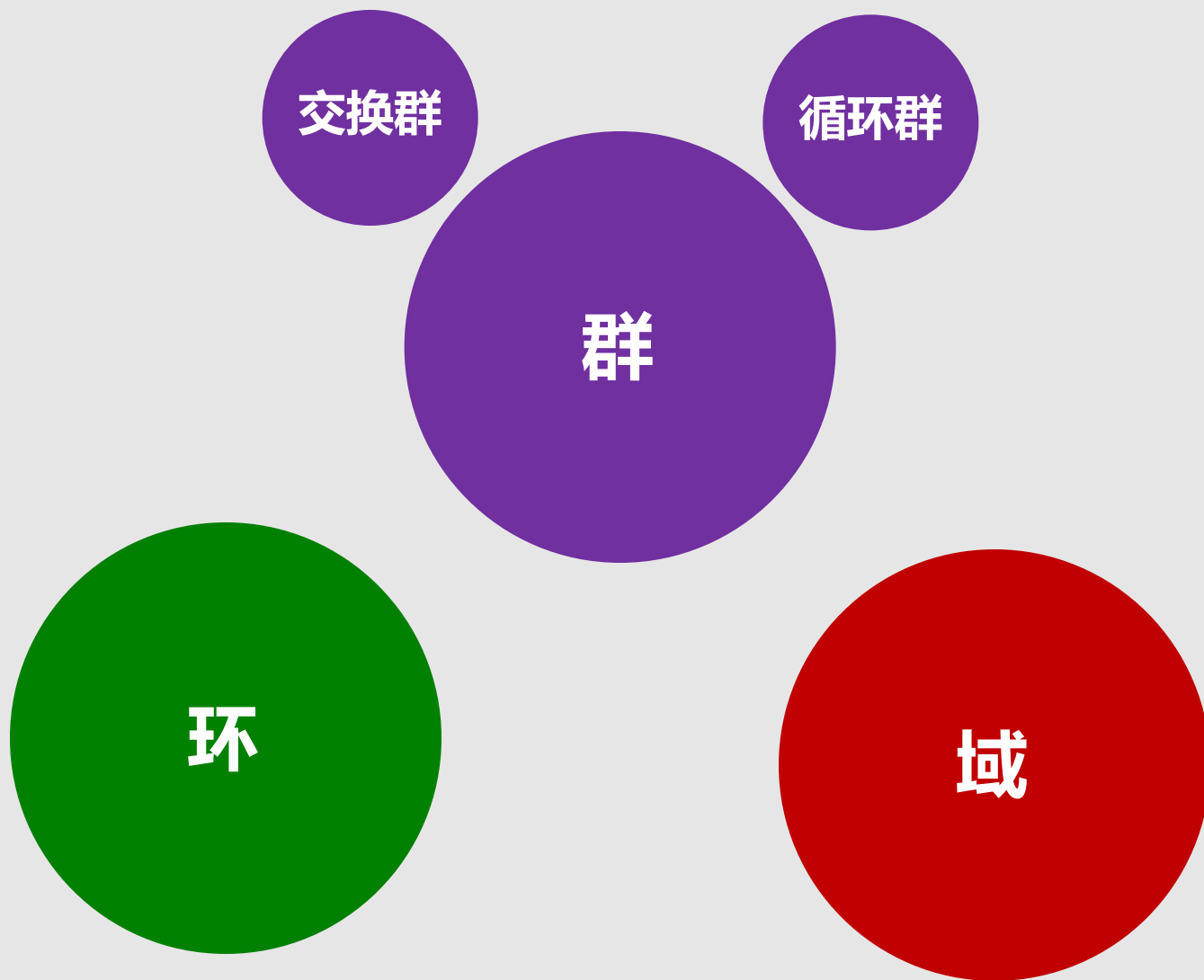
勒让德
符号

雅可比
符号

Blum
整数



5.2 抽象代数



- **代数系统由两部分组成**：非空集合、作用在集合上的运算。
- 一般写为 $(G, *)$ ，其中 G 是一个非空集合， $*$ 是作用在 G 上的运算。
- 也有具有多个运算的代数系统，最常用的是具有两个运算的。



Niels Henrik Abel (阿贝尔)

(1802 — 1829)

19世纪挪威最伟大的数学家
(埃尔米特曾说：阿贝尔留下的思想可供数学家们工作150年)



Evariste Galois (伽罗瓦)

(1811 – 1832)

19世纪法国天才少年数学家
(伽罗瓦理论创始人)

- 群的定义

设代数系统 $(G, *)$ ，对于任意 G 中的元素 a 、 b 、 c ，满足以下条件，则它是一个群：

(1) 封闭性: $a * b \in G$

(2) 结合性: $a * (b * c) = (a * b) * c$

(3) 单位元: 存在 $e \in G$ 使得 $a * e = e * a = a$

(4) 逆元: 存在 $d \in G$ 使得 $a * d = d * a = e$,

通常把 a 的逆元记为 a^{-1} 。

- 为简便起见，常直接称 G 为群

群

- 判定一个非空集合是不是群，一般用群的定义去衡量
- 例1： $G=(\mathbb{Z}, +)$ 是一个群。

\mathbb{Z} 是所有整数的集合， $+$ 是普通的整数加法。

- 原因：

(1)封闭性：对任意两个整数，相加后仍为整数

(2)结合性：整数加法满足结合律

(3)单位元：对任意整数 a ，都有 $a + 0 = 0 + a = a$

所以， 0 是 G 的单位元

(4)逆元：对任意整数 a ，都有 $a + (-a) = (-a) + a = 0$

所以， $a^{-1} = -a$

故而， G 是一个群

- **例2：**

$G=(\mathbb{N}, \times)$ 不是群。

\mathbb{N} 是自然数集合， \times 是普通的整数乘法。

- **原因：**

$\mathbb{N}=\{1,2,\dots\}$ ，在乘法运算下满足封闭性和结合性，且有单位元1。

但除1以外，所有元素都没有逆元。

- **群的阶**

群中元素的个数，通常记为 $|G|$ 。

- **元素的阶**

设群 $(G, *)$ ，若 $a \in G$ ，有

$$a * a * \dots * a = a^n = e$$

且 n 是满足这一公式的最小正整数，称 n 为 a 的阶。

- **有限(无限)群**

群的阶是有限(无限)的

1. 单位元是唯一的

2. 元素的逆元也是唯一的

3. 群中没有零元

- **交换群的定义**

群 $(G, *)$, 对于任意 $a, b \in G$, 若有

$$a * b = b * a \quad (\text{运算满足交换律})$$

则称该群为交换群(阿贝尔群)。

- **循环群定义**

设群 $(G, *)$, 若存在 $a \in G$, 使得对于任意元素 $b \in G$, 都存在一个整数 m , 有

$$a^m = b$$

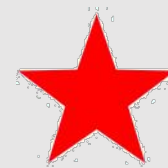
则称该群为循环群。称 a 为生成元。

- **含义**

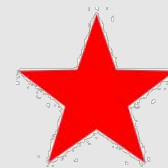
群中存在一个元素(生成元), 其他元素都可以由该元素生成。

- 循环群中的元素都可以表示为 a^m 的形式, 其中 m 是整数。

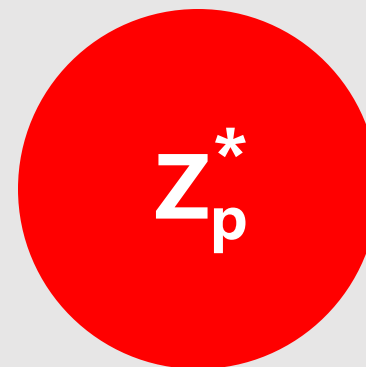
因此, 循环群中的元素为 $a^0 = e, a^1, a^2, \dots, a^m, \dots$

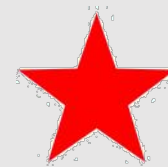


- 任何循环群必是交换群
- 设有限循环群的阶为 n ，则
 - ① 生成元的阶也是 n ，其他元素的阶必是 n 的因子
 - ② 生成元的个数为 $\phi(n)$
 - 例：循环群中有15个元素，那么生成元的个数为 $\phi(15) = \phi(3) \phi(5) = 2 \times 4 = 8$
 - ③ 阶为 m 的元素的个数共有 $\phi(m)$ 个



- Z_p 表示小于素数 p 的非负整数集合
 $\{0, 1, 2, \dots, p-1\}$
- Z_p^* 表示小于素数 p 且与 p 互素的正整数集合
 $\{1, 2, \dots, p-1\}$
- Z_p^* 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 $\phi(p-1)$
 - 例： Z_7^* 是一个循环群，生成元共有 $\phi(6)=2$ 个，分别是3和5。





- Z_n 表示小于 n 的非负整数集合(n 是合数)
 $\{0, 1, \dots, n-1\}$
- Z_n^* 表示小于 n 且与 n 互素的正整数集合
例如： $Z_8^* = \{1, 3, 5, 7\}$
- Z_n^* 在模 n 乘法运算下是一个群，阶是 $\phi(n)$
因为只有与 n 互素才有乘法逆元



例：

Z_{15}^* 是一个模15下的乘法群

Z_{15}^* 中的元素是 $\{1, 2, 4, 7, 8, 11, 13, 14\}$

阶是 $\phi(15)=8$

- 域的概念最初被阿贝尔和伽罗瓦用于对方程可解性的工作上。
- 现如今，域是密码学，特别是公钥秘密学中最重要数学基础。

- 定义：

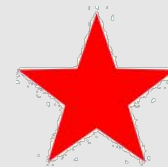
域 $(F, +, *)$ 必须满足以下条件

- F 关于 $+$ 构成交换群。单位元记为0。称为加法群
- F 中非0元素对 $*$ 构成交换群。单位元记为1。称为乘法群
- $*$ 对 $+$ 满足分配律：

$$a*(b+c) = a*b + a*c$$

$$(b+c)*a = b*a + c*a$$

则称 F 为一个域



- 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 p^n 的形式(p 为素数)，记为 F_{p^n}
- **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 F_p (又称素数域)和 F_{2^n}

不可约
多项式

多项式
环

域的
扩张

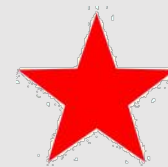


5.3 计算复杂性理论

- 计算复杂性理论在现代密码学中占有重要地位
- 它为现代密码学提供理论基础，并可以对密码算法的安全性进行评估和比较



- **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



- 目前出现了一种更强大的计算模型
 - 量子计算
- 该模型下，指数级计算可以并行的完成，致使很多有用的困难问题不再困难。
- 不过，量子计算离实际应用还有很长一段路要走，我们不必为此过分担忧。
- 因此，下面我们将介绍“不够强”的传统计算模型和现代密码学的计算复杂性理论基础。

- 一个算法的复杂度即运行所需的计算能力，常用两个量来度量
 - 时间复杂度
 - 空间复杂度
- 推导算法的精确运行时间是非常困难的。
- 因此，描述算法的运行时间时，就利用渐进表示方法，以研究算法运行时间如何随着输入长度变化的。
- 这样，描述运行时间就不依赖于具体的计算机系统。

- **多项式**

函数 $p(n)$ 是整数上关于 n 的多项式，则它有如下形式：

$$p(n) = c_k n^k + c_{k-1} n^{k-1} + \dots + c_1 n + c_0$$

其中 k 和 c_i 是常整数。

如果算法复杂度不依赖于 n ，那么它是常数的，记为 $O(1)$ 。

如果是 $O(n)$ ，那么是线性的

如果多项式的次数是常数 $k > 1$ ，复杂度常表示为 $O(n^k)$

- 最坏运行时间复杂度是 $O(n^k)$ 的算法，称为**多项式时间**(polynomial-time)算法。
- 有些算法的计算时间慢于多项式时间，但快于指数时间，故称为超多项式时间的，或亚指数时间的。
- **在计算复杂性理论中，我们主要关注多项式时间算法**

计算复杂性

不同算法族的运行时间差异巨大

族	复杂度	操作次数	时间
常数的	$O(1)$	1	1微秒
线性的	$O(n)$	10^6	1秒
二次方的	$O(n^2)$	10^{12}	11.6天
三次方的	$O(n^3)$	10^{18}	32,000年
指数的	$O(2^n)$	10^{301030}	宇宙年龄的 10^{301006} 倍

假设 $n=10^6$, 计算机的时间单位是微秒

- 1912年6月23日生于英国伦敦
- 24岁提出图灵机理论
- 31岁参与破译Enigma
- 33岁设想仿真系统
- 35岁提出自动程序设计概念



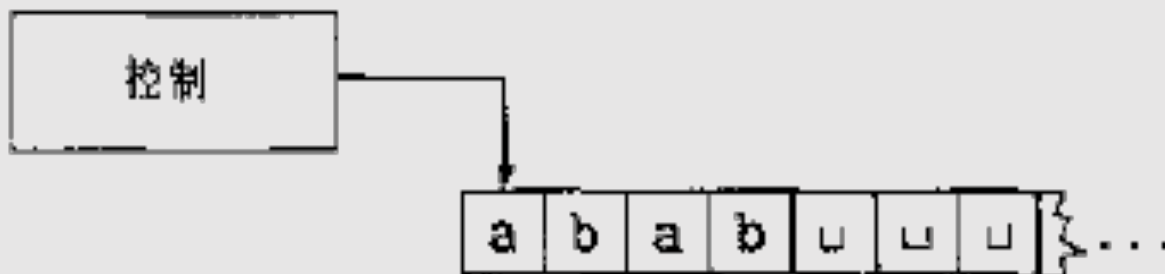
Alan Turing
1912 –1954

- **图灵机**
 - 刻画解决某问题所花费的时间和空间的工具
 - 是一个精确的通用计算模型
 - 用于精确定义算法这一概念的
- 它能做实际计算机能做的所有事情
 - 当然也有它不能解决的问题，但这已超出计算的理论极限。
- 在计算复杂性理论中，认为一个问题已经解决，是指**该问题的所有实例都可以用同一个图灵机(同一个算法)求解**，只有如此才认为该算法是充分通用的

图灵机的组成

- ①一条(多条)纸带。具有无限个单元(小格子)，纸带的右端可以无限伸展。
- ②一个读写头。可以在纸带上读、写或移动。
- ③控制器。根据内部状态和读写头所指单元上字符，确定读写头下一步动作。

状态的数目是有限的，且有两个特殊状态：接受状态和拒绝状态。



图灵机只是一个理想的设备(计算模型)

- 虽然它的每一部分都是有限的，但有一个无限长的纸带。

- ① 开始时，图灵机处于初始状态，纸带上从最左端向右依次填写输入字符串，后面的单元格都是“空白”，一望无际。
- ② 图灵机可以左右移动读写头。
- ③ 也可在读写头所指单元上的读取字符
- ④ 或在读写头所指的单元上输出字符
- ⑤ 图灵机不停地计算，可根据需要改变状态，直到产生输出为止(进入接受或拒绝状态)
- ⑥ 如果不进入接受或拒绝状态，图灵机将一直执行下去，永不停止。

- 例：

给定一个比特串，设计一个方法，让图灵机动起来，并让它识别该串表示的整数是否是偶数。

- 是的话，就进入接受状态。
 - 不是的话，就将之改成偶数，并进入拒绝状态。
- 面对一个长长的比特串，你所能做的只是在串上来回移动，但允许你做标记。你该怎么做？

- **思路**
 - 将读写头移动到最后一个比特上，判断它是1还是0
- **需解决的问题**
 - ① **如何判断哪个比特是最后一个比特？**
 - 最后一个比特的右边位置必然存放的是“空白”
 - ② **图灵机并没有判断功能，如何让它有判断能力？**
 - 利用内部状态和当前单元格上的字符，使图灵机进入新状态，从而实现判断的功能

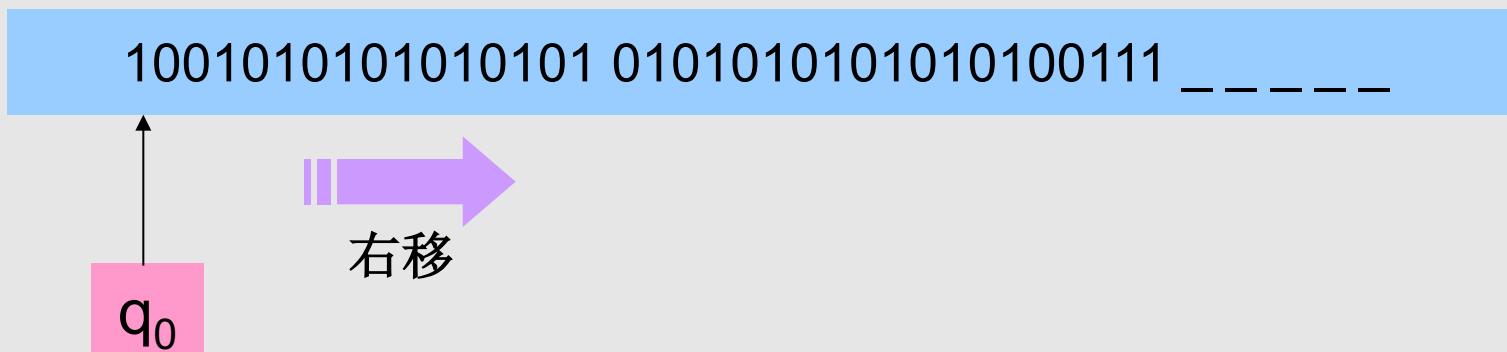
状态转换图

	0	1	空白
q_0	$(q_0, 0, R)$	$(q_0, 1, R)$	$(q_1, \text{空白}, L)$
q_1	$(q_T, 0, -)$	$(q_R, 0, -)$	-

- L、R：分别表示读写头向左移动一格，或向右移动一格
- q_T 和 q_R 分别表示接受和拒绝状态

实例

	0	1	空白
q_0	$(q_0, 0, R)$	$(q_0, 1, R)$	$(q_1, \text{空白}, L)$
q_1	$(q_T, 0, -)$	$(q_R, 0, -)$	-



- ① 初始时，读写头指向比特串开头, 内部状态为 q_0
- ② 向右移动读写头，直到遇到第一个“空白”为止

计算复杂性 图灵机举例

	0	1	空白
q_0	$(q_0, 0, R)$	$(q_0, 1, R)$	$(q_1, \text{空白}, L)$
q_1	$(q_T, 0, -)$	$(q_R, 0, -)$	-

1001010101010101 0101010101010100111 _ _ _ _



左移一格

q_0

- ③ 左移一格，将读写头指向串的最后一个比特，并改变内部状态为 q_1

计算复杂性 图灵机举例

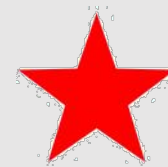
	0	1	空白
q_0	$(q_0, 0, R)$	$(q_0, 1, R)$	$(q_1, \text{空白}, L)$
q_1	$(q_T, 0, -)$	$(q_R, 0, -)$	-

1001010101010101 0101010101010100111 _ _ _ _ _

q_1

④ 读写头已指向串的最后比特

此时，遇到0便进入接受状态 q_T 遇到1便改写为0，并进入拒绝状态 q_R



- **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

- **确定性图灵机**

- 其输出结果完全取决于输入和初始状态。
- 在每个状态，遇到某一字符时，下一步的动作是确定的(固定的)。
- 也就是说，不论图灵机运行多少次，对于同样的输入和初始状态，其输出是相同的。

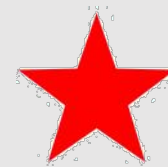
	0	1	空白
q_0	$(q_0, 0, R)$	$(q_0, 1, R)$	$(q_1, \text{空白}, L)$
q_1	$(q_T, 0, -)$	$(q_R, 0, -)$	-

- **非确定性图灵机**

- 进行下一步动作时，可以有有限个选择。
- 它任意选择一个继续执行，直到最后停机为止。
- 可以这样想象：非确定性图灵机解决问题时要进行一系列猜测。

	0	1	空白
q_0	$(q_0, 0, R)$ $(q_1, \#, L)$	$(q_0, 1, R)$ $(q_0, 0, R)$ $(q_1, 1, R)$	$(q_1, \text{空白}, L)$ $(q_T, 0, -)$
q_1	$(q_T, 0, -)$	$(q_R, 0, -)$ $(q_T, 0, -)$	-

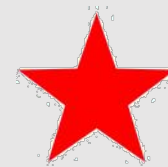
- 非确定性图灵机的工作分为两个阶段
 - ① 猜测阶段
 - 通过一系列猜测步骤，输出一个结果
 - ② 验证阶段
 - 检查猜测的结果是否是问题所求的答案



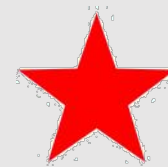
- **P问题**

- 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 $n=2,3$ ， $O(n^{200})$ 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

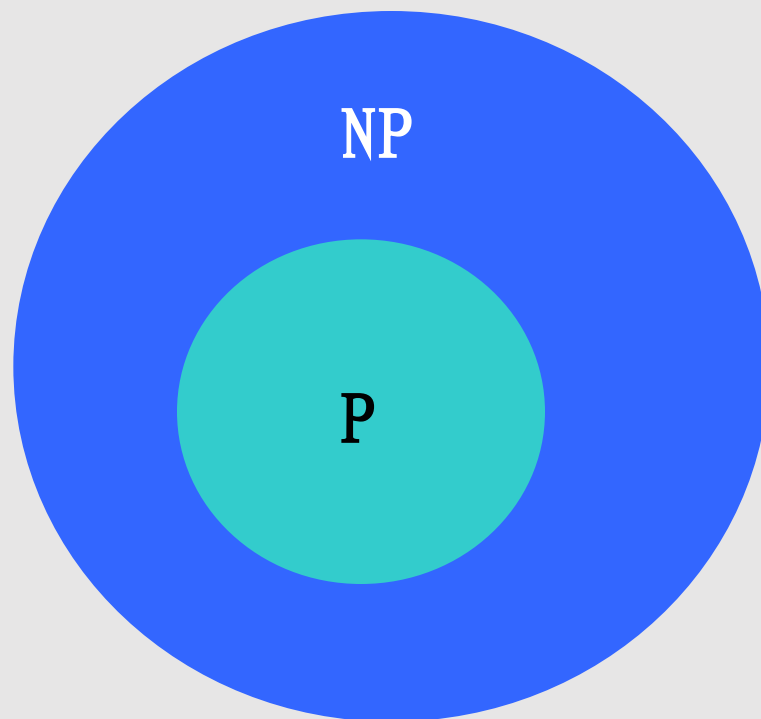
- 有些问题很难，到目前为止，也没发现解决这些问题的多项式时间算法。
- 也即，确定性图灵机上，还没发现解决这些问题的多项式时间算法。
- 当然，解决的算法是有的，只是效率很低。（因为不是多项式时间的）
- 如果我们改用 非确定性图灵机，则可以找到解决这些问题的多项式时间算法。

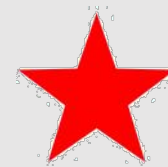


- **NP问题**
 - 存在一个非确定性图灵机，可以在多项式时间内解决的问题。
- 也就是说，NP问题用非确定性图灵机很容易解决。
- 因为在确定性图灵机上未发现多项式时间的算法，通常认为NP问题是难以解决的，俗称 **困难问题**

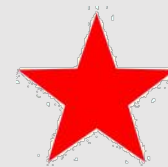


- **P问题也属于NP问题的范畴**
 - 因为确定性图灵机上在多项式时间内可以解决的问题，在非确定性图灵机上用多项式时间也可以解决。（把猜测阶段省略掉即可）
- **但我们一般所说的困难问题或NP问题，通常不包括P问题。**





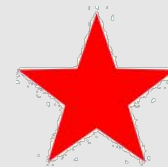
- NP中有一些特殊问题，如果找到一个确定的多项式时间算法可以解决这些问题中的一个，解NP中的任何问题都能找到多项式时间算法。
- 这类问题被称作NP完全问题，记为**NPC**.



- 由前述可知

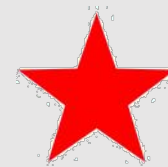
$$P \subseteq NP$$

- 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

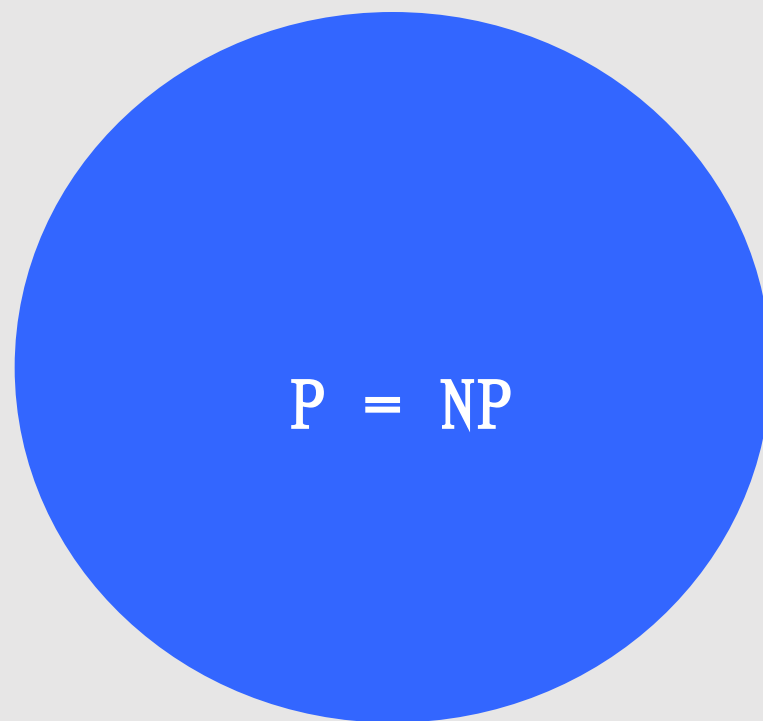
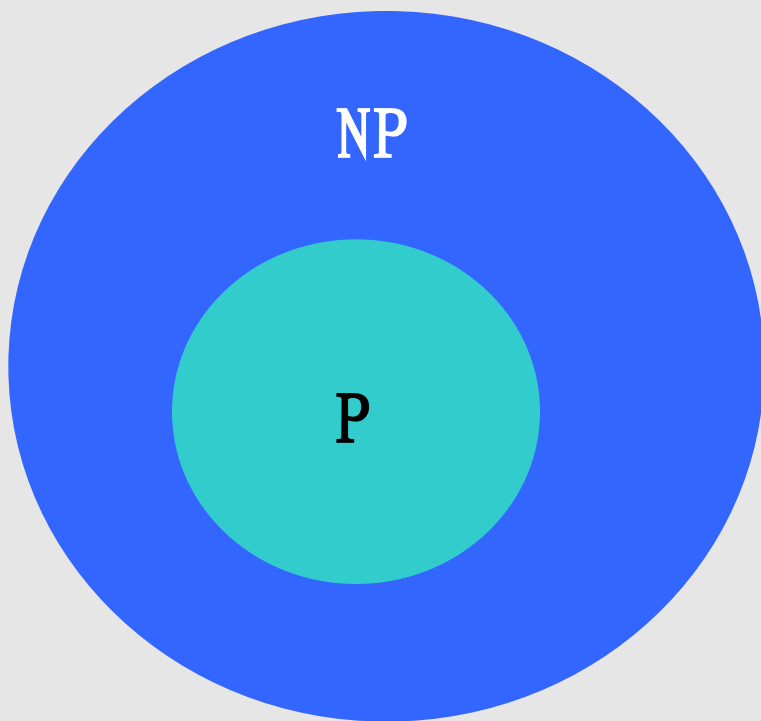
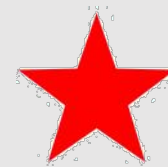


Q: 是否存在 $NP \subseteq P$?

- 如果 $NP \subseteq P$ ，也即所有NP问题都可以用确定性图灵机在多项式时间内解决，那么必有 $P=NP$ 。
- 但 **$P \neq NP$ 或 $P=NP$ 至今没被证明**，不过人们猜想它俩不相等。



“P是否等于NP” 是计算复杂性理论未解决的中心问题



两种关系，二者必有其一

本章小结

1. 掌握素数、最大公约数和模运算的基本性质
2. 掌握欧拉函数、欧拉定理、费马小定理的运算
3. 掌握群、循环群、域的概念和相关性质定理
4. 掌握 Z_p^* 和 Z_n^*
5. 理解P问题、NP问题、NPC问题的含义
6. 了解二次剩余的含义
7. 了解图灵机的工作方式

练习题

1. 求解欧拉函数 $\phi(2000)$

$$\begin{aligned}\text{解： } \phi(2000) &= \phi(2^4 5^3) \\ &= 2^3 \phi(2) 5^2 \phi(5) \\ &= 8 \times 25 \times 4 \\ &= 800\end{aligned}$$

练习题

2. 利用费马小定理计算 $3^{101} \bmod 13$

解：因为 $\gcd(3,13)=1$ ，所以可以利用费马小定理计算

又因为 $101=8\times 12+5$

所以 $3^{101} \equiv 3^{8\times 12+5} \equiv 3^5 \equiv 243 \equiv 9 \pmod{13}$

练习题

3. 如果在干支纪年法中，天干有36个，地支有15个，请问“六十年一甲子”的说法应该改成多少年一甲子，请列出公式并计算。

解：利用最小公倍数计算得：

$$\text{lcm}(36, 15) = 180$$

所以，应该改成“一百八十年一甲子”