CINTA 重点

根据华南师范大学计算机学院斌头老师的课件整理

整理者：0.H.P

原文链接：https://www.jianshu.com/p/34107ca9b4ee

1 可除性和带余除法

1. 定理：设 a,b $\in Z$,$b > 0$, 那么 $\exists q, r \in Z$, 使得 $a = bq + r$ 其中 $0 \leq r \leq b-1$, 且这对整数 q,r 为上述 a,b 所唯一确定。

2. 证明：
   (1). 证 r 的范围：
   将实数轴分为左闭右开的长度均为 b 的无穷个小区间，即

$$[nb, (n+1)b), n \in Z$$

   可得 a 必定落在某一区间内，故 $\exists q \in Z$, 使得 $qb \leq a < (q+1)b$. 令 $r = a - bq$, 则 $0 \leq r < b$，又因为 $r \in Z$, 故得 $r \leq b-1$.
   (2). 证 q 与 r 的唯一性：
   首先证 a 所在区间的唯一性：运用反证法。
   假设 $a \in [nb, (n+1)b)$ 且 $a \in [mb, (m+1)b)$，其中 $n \neq m$, 那么假设 $n < m$. 得 $n+1 \leq m$ 那么假设 $\exists x \in [nb, (n+1)b), \exists y \in [mb, (m+1)b)$, 那么有：

$$\lim_{x \to max} x < \lim_{y \to min} y$$

   故得：a 所在区间唯一。又由于 $a \in [qb, (q+1)b)$, 得 q 唯一。又 a=qb+r, 得 r 唯一。
   证毕。

2 欧几里得算法 (gcd)

1. 目的：求两个数的最大公约数

2. **公式**：$gcd(a, b) = gcd(b, a \ mod \ b)$ 直到 $b = 0$

3. **方法**：对于 $ab \neq 0$, 且 $b > 0, b \nmid a$, 运用带余除法表达式，有

$$a = bq_0 + r_0, q_0 与 r_0 \in Z, 且有 r_0 \in [1, b-1]$$

$$b = r_0 q_1 + r_1, q_1 与 r_1 \in Z, 且有 r_1 \in [1, r_0 - 1]$$

$$r_0 = r_1 q_2 + r_2, q_2 与 r_2 \in Z, 且有 r_2 \in [1, r_1 - 1]$$

$$r_1 = r_2 q_3 + r_3, q_3 与 r_3 \in Z, 且有 r_3 \in [1, r_2 - 1]$$

$$\vdots$$

   可得除第一与第二式外，其余都是用后一个余数除前一个余数的余数，即 $r_0 = a \ mod \ b, r_1 = b \ mod \ r_0 \cdots$ 以此计算，直到余数为 0，即

$$r_{n-1} = r_n q_{n+1}, q_{n+1} \in Z$$

   r_{n-1} 即为结果。

4. **证明**：假设 $a, b, d, q \in Z$, 且不全为 0，且有 $a = bq + c$.
   若 a 与 b 的公因数为 d,$\Rightarrow d|a$ 且 $d|c \Rightarrow d|qb + c \Rightarrow d|c$
   即 d 也是 b 与 c 的公约数 $\Rightarrow gcd(a, b) = gcd(b, c)$
   以此计算，有 $gcd(a, b) = gcd(b, r_0) = \cdots gcd(r_n q_{n+1}, r_n)$
   证毕.

5. 时间复杂度：$log_2 n$

6. 递归算法：

```
def gcd(a,b):
    if b==0:
        return a
    else:
        return gcd(b,a%b)
```

7. 迭代算法：

```
def gcd(a,b):
    while b!=0:
        y=b
        b=a%b
        a=y
    return a
```

3 拓展欧几里得算法 (egcd)

1. 目的：求线性方程组 $a * r + b * s = gcd(a, b)$ 的一组解

2. 方法： $q = a//b$

$$\begin{bmatrix} r_0 & s_0 & a \\ r_1 & s_1 & b \end{bmatrix} = \begin{bmatrix} r_1 & s_1 & b \\ r_0 - q * r_1 & s_0 - q * s_1 & a\%b \end{bmatrix} = ... = \begin{bmatrix} r_n & s_n & a\%..\%b \\ r_{n-1} - q * r_n & s_{n-1} - q * s_n & 1 \end{bmatrix}$$

   以此循环

```
def egcd(a,b):
r0,r1,s0,s1=1,0,0,1
while b:
        q,a,b=a//b,b,a%b
        r0,r1=r1,r0-q*r1
        s0,s1=s1,s0-q*s1
print(a,r0,s0)
```

4 费马小定理

1. 公式：p 是素数时，$a \in [1,p)$, 有 $a^{p-1} \equiv 1(mod\ p)$

2. 证明：
   取 $i,j\ [1,p-1), i \neq j\ \ a*i \equiv a*j(mod\ p)$
   由消去律得 $i \equiv j(mod\ p),\ i=j,$
   故得 $a*i(mod\ p)$ 有 p-1 种结果
   所以
   $$\prod_{i=1}^{p-1} i = \prod_{i=1}^{p-1} a*i = a^{p-1}\prod_{i=1}^{p-1} i(mod\ p)$$
   即 $a^{p-1} \equiv 1(mod\ p)$

5 欧拉公式

1. 公式：$a^{(n)} \equiv 1(mod\ n)$

2. 证明：$(n) = |b : 1 <= b < n\ and\ gcd(b,n) = 1|$
   设集合 $S = \{b_1, b_2, b_3 ... b_i : 1 <= b_i < n\ and\ gcd(b_i, n) = 1\}$
   $S' = \{a*b_1, a*b_2 ... a*b_1 : 1 <= b_i < n\ and\ gcd(a,n) = 1\}$
   反证：设存在 $i \neq j; b_i, b_j \in S,\ a*b_i = a*b_j(mod\ n)$
   由消去律得 $b_i = b_j(mod\ n)$
   与条件矛盾，故得 $S = S'$
   所以
   $$\prod_{i=1}^{\varphi(n)} b_i = \prod_{i=1}^{\varphi(n)} a*b_i = a^{\varphi(n)}\prod_{i=1}^{\varphi}(n)b_i(mod\ n)$$

   由消去律得 $a^{\varphi(n)} \equiv 1\ (mod\ n)$

6 米勒拉宾素数测试

米勒拉宾算法：一种检测一个整数是否为素数的算法，准确率约大于 3/4，即伪素数通过检验的概率约小于 1/4。
介绍该算法前，先了解素数的两个性质。

(1) 若 p 是素数，那么存在 $1 < a < p$，当且仅当 $a\ mod\ p = 1$ 或 $a\ mod\ p = -1 = p-1$(逆元)时，有 $a^2\ mod\ p = 1$

(2) 若 p 是素数，且 $p > 2$ 那么有：$p - 1 = 2^k q, k > 0, 2 \nmid q$, 有 $a \in (1, p-1)$, 满足一下条件之一：

- $a^q \equiv 1(mod\ p)$
- 在 $a^q, a^{2q}, a^{4q} ... a^{2^{(k-1)}q}$ 中，必然存在一个数模 p 与 -1 同余。

理解这两个性质：
首先，从费马小定理出发，有 $a^{(p-1)} \equiv 1(mod\ p)$, 那么，由于 p 是素数，所以 $2 \mid p-1$, 那么 $p-1$ 可表示为 $p-1=\frac{p-1}{2}*2$ 那么得 $a^{(p-1)}=a^{\frac{p-1}{2}*2} \equiv 1(mod\ p)$, 那么必然有 $a\ mod\ p=1$ 或 $a\ mod\ p=-1=p-1$(逆元)(由此理解性质 1)
然后，将 $(p-1)$ 按以上思路进行分治，直到 $(p-1)=2^k q, 2 \nmid q$, 分治过程中，要么所有值都为 1，那么必然 $a^q \equiv 1$; 或者中间某个值为 -1.(由此理解性质 2)

算法步骤：

1. 将 $(p-1)$ 分解至 $(p-1)=2^k q, 2 \nmid q$, 得到 q
2. 选取随机数 $a, a \in (1, p-1)$
3. 若 $a^q\ modp=1$, 通过检验; 否则循环一下操作
4. 计算 $a^{2^j q}$, 其中 $j \in (0, k-1)$ 若出现有结果为 -1, 那么通过检验，否则不通过

应用：
一般用作检验时，循环 10 次以上米勒拉宾操作，以保证正确概率。
可用该算法，随机生成一个大整数，然后检验该数是否为素数的方法，生成随机素数。

代码：

```
#A quickly multiply algorithm with modding n
def Qmul(a,b,n):
    r=0
    while b:
        if b&1:
            r=(r+a)%n
        b=b>>1
        a=(a<<1)%n
    return r

#A quidckly exponential algorithm with modding n
def QPow(a,x,n):
    result=1
    while x:
        if x&1:
            result=(result*a)%n
        x=x>>1
        a=(a*a)%n
    return result

#Find the number = (n-1)/2^k
def Find_q(n):
```

```
23      while not n&1:
24          n=n>>1
25      return n
26
27  #Miller Rabin algorithm
28  def Miller_Rabin(a,n):
29      q=Find_q(n-1)
30      aq=QPow(a,q,n)
31      #final condition,q=n-1
32      while q<n:
33          if aq==1 or aq==-1:
34              return True
35          #make aq = aq(q*2^j)
36          aq=QPow(aq,2,n)
37          q=q<<1
38      return False
```

7 群

1. 性质：
   - 满足封闭性
   - 满足结合律
   - 存在单位元
   - 存在逆元
2. 单位元唯一性:
   设 e' 亦为 G 的单位元
   那么有：
   $ee' = e'(e$是单位元$)$
   $ee' = e(e'$是单位元$)$
3. 逆元唯一性:
   设 $g \in G$,g 的逆元为 g^{-1}, 即 $gg^{-1} = e$
   存在性：在群操作中，运算满足封闭性，h 恒存在
   唯一性：设有另一个逆元 h, 使得 $gh = e$
   那么：$gg^{-1} = gh$, 由消去律得 $h = g^{-1}$
   所以 $e = e'$
4. 子群：H 中元素的集合是 G 的子集，H 的群操作与 G 相同，称 H 是 G 的子群

7.1 循环群 (cyclic group)

1. 定义：$g \in G$,$g = \{g^k : k \in Z\}$ 为循环群，g 是生成元
2. 阶：满足 $g^n = e$, 取 n 的最小值, 阶是 n, 记 $|g| = n$
3. 例：
   群 Z_p^*，阶为 $\varphi(p)$, 生成元个数 $\varphi(\varphi(p))$, 生成元 a, 有 $gcd(a, \varphi(p)) = 1$
   群 Z_{11}^*, 阶为 $\varphi(p) = 10$, 生成元个数 $\varphi(\varphi(p)) = 4$, 生成元：$\{1, 3, 7, 9\}$
4. 定理一：若存在正整数 k，使得 $g^k = e$, 当且仅当 k 整除 n.
   证：
   充分性：令 $g^k = g^{ns} = e$
   必要性：令 $k = nq + r$
   $e = g^k = g^{nq+r} = g^{nq}g^r = eg^r (0 <= r < n)$
   由消去律得：$g^r = 0$，故 $r = 0$
5. 定理二：$h = g^k$,h 的阶为 $n/d, d = gcd(k, n)$
   证：
   设有正整数 m, 使得 $h^m = g^{mk} = e$
   $d = gcd(n, k)$
   故 $n \mid mk$, 那么：$(n/d) \mid (k/d)m$
   由于 (n/d) 与 (k/d) 互素，无法整除
   故

$$(n/d) \mid (k/d)m = (n/d) \mid m$$

   所以 $m_{min} = n/d$

7.2 陪集 (coset)

1. 定义：
   H 是 G 的子群，对于 $g \in G$
   左陪集：$gH = \{gh : h \in H\}$
   右陪集：$Hg = \{hg : h \in H\}$
2. 例：
   设 $H = \{[0], [2]\}$ 是 Z_4^+ 的 subgroup，取所有 $g \in Z_4^+$
   遍历:

$$[0] + H = \{[0], [2]\} \text{ ①}$$

$$[1] + H = \{[1], [3]\} \text{ ②}$$

$$[2] + H = \{[0], [2]\} \text{ ③}$$

$$[3] + H = \{[1].[3]\} \text{ ④}$$

得 ① ③， ② ④ 分别相等，所以左陪集有俩：{[0][2]} 与 {[1][3]}
同理可求右陪集

3. 性质 1：H 是 G 的 subgroup，则有 $g_1, g_2 \in G, g_1H = g_2H$或者$g_1 \cap g_2 = \emptyset$

4. 性质二：划分。
所有左陪集构成一个划分，所有右陪集构成一个划分

7.3 拉格朗日定理

1. 定义：G 是有限群，H 是 G 的子群，则 $|G| = |H| * [G : H], [G : H]$ 为 H 在 G 中右陪集的个数。
证明：
设 $[G : H] = r; a_1, a_2...a_r$ 分别为 H 的 r 个右陪集的代表元素
由 $G = Ha_1 \cup ...Ha_r$
由于每个陪集不同，得：$|G| = |Ha_1| + ...|Ha_r|$
$|Ha_i| = |H|$
$|G| = |H| * r = |H|[G : H]$

2. 推论一：群 G 的阶为 n, 所有 $a \in G, |a|$是 n 的因子，且有$a^n = e$

3. 推论二：G 是素数阶群, 则存在 $a \in G, G =< a >$

7.4 同构 (Isomorphisms)

1. 定义：
群 $(G, \cdot)$ $and(H, *)$, 满足 one-to-one(单射) and onto(满射) 映射, 称 φ 为一个同构 (isomorphic)

2. 证明方法：

   (a) 构造映射

   (b) 证明阶相同

   (c) 满足群操作

   (d) 双射：
   (1) 单射 (one-to-one): 利用反证法,$a, b \in G$, 且 $a \neq b$, 有 $f(a) \neq f(b)$
   (2) 满射 (onto)：$a \in G, b \in H$, 有 f(a)=b 恒成立。

   (e) 群操作保持：即先映射后计算等于先计算后映射 (证同态)

7.5 同态 (Homomorphisms)

定义：群 $(G, \cdot)$ $and(H, *)$，φ 是两者间的一个映射，对于所有 $a, b \in G$

$$\varphi(a \cdot b) = \varphi(a) * \varphi(b)$$

7.5.1 正规子群 (normal subgroup)

1. 定义：H 是 G 的子群，对于所有 $a \in G$, 有 aH=Ha, 称 H 为 G 的正规子群 (即左右陪集相等)

2. 性质：
   若群 G_1 与 G_2 同态，则

   (a) e 是 G_1 的单位元，那么，e 也是 G_2 的单位元

   (b) 对任何 $g \in G_1, \varphi(g^{-1}) = [\varphi(g)]^{-1}$

   (c) H_1 是 G_1 的子群，那么，$\varphi(H_1)$ 是 G_2 的子群

   (d) H_2 是 G_2 的子群，那么，$\varphi^{-1}(H_2)$ 是 G_1 的子群
   H_2 是 G_2 的正规子群，那么，$\varphi^{-1}(H_2)$ 是 G_1 的正规子群

7.5.2 商群 (quotient group)

1. 定义：
   N 是 G 的正规子群，对于所有 $a, b \in G$, 有 (aN)(bN)=abN
   则 N 的陪集 a,b 构成商群，记为 G/N

2. 性质：商群的阶等于陪集元素的个数，即 $|G/N| = [G : N]$

3. 例：
   $3Z^+$ 是 Z^+ 的正规子群得

$$0 + 3Z^+ = \{..., 0, 3, 6...\}$$

$$1 + 3Z^+ = \{..., 1, 4, 7...\}$$

$$2 + 3Z^+ = \{..., 2, 5, 8...\}$$

   满足 $(aN)(bN) = (abN)$

8 中国剩余定理

1. 概念：一种求一次同余式组的算法

2. 算法：
   有一次同余式组：

$$x \equiv a_1 (mod\ m_1)$$

$$x \equiv a_2 (mod\ m_2)$$

$$.....$$

$$x \equiv a_n (mod\ m_n)$$

   解：
   令：$M = \prod_{i=1}^{n} m_i$
   $b_1 = M/m_i$

$b'_i = b_i^{-1}(mod\ m_i)$(乘法逆元)

$$x = \sum_{i=1}^{n} a_i b_i b'_i (mod\ M)$$

3. 代码：

```
def CRT(a,m):
        M,x=1,0
        b=list()
        b_=list()
        for n in range(len(m)):
                M=M*m[n];
        for i in range(len(m)):
                b.append(M//m[i])
                '''
                compute b_
                because b_[i]*b[i]=1(mod m[i])
                so b[i]*b_[i]-1=c1*m[i]
                than b[i]*b_[i]+c2*m[i]=1
                use egcd to compute the result
                '''
                b_.append(egcd(b[i],m[i]))
                '''
                when the result is a negative number
                let it become a positive number which is a
                   Equivalence class about b_[i]
                '''
                if b_[i]<0:
                        b_[i]=b_[i]%m[i]
                x=(a[i]*b[i]*b_[i]+x)%M
        return x
```

4. 证明：

假设 x0 为解，且唯一。

由 $b'_i(mod\ m_i)$ 的定义可知，恒有：

$$x_0 \equiv a_i (mod\ m_i)(1 \le i \le k)$$

而由于

$$x \equiv a_i (mod\ m_i)$$

得

$$x \equiv x_0 (mod\ m_i)(1 \le i \le k)$$

且由于 $m_1, m_2...m_k$ 两两互素，故得

$$x \equiv x_0(mod\ m)$$

证毕。

5. 例：
Using CRT to solve the system of congruence:

$$x \equiv 1\ (mod\ 5)$$
$$x \equiv 2\ (mod\ 7)$$
$$x \equiv 3\ (mod\ 9)$$
$$x \equiv 4\ (mod\ 11)$$

解：由题得：
令：$m_1 = 5, m_2 = 7, m_3 = 9, m_4 = 11;$
所以

$$M = \prod_{i=1}^{4} m_i = 3465$$

$a_1 = 1,\ a_2 = 2,\ a_3 = 3,\ a_4 = 4;$
$b_i = M/m_i;\ b_i^{'} = b_i^{-1}(mod\ m_i)$
$b_1 = 693,\ b_2 = 495,\ b_3 = 385,\ b_4 = 315;$
因为 $b_i b_i^{-1} = 1(mod\ m_i)$
得 $b_i b_i^{-1} - 1 = c_c m_i$
$b_i b_i^{-1} + c_2 m_i = 1$
由拓展欧几里得算法解得
$b_1^{'} = 2,\ b_2^{'} = 3,\ b_3^{'} = 4,\ b_4^{'} = 8$

$$x = \sum_{i=1}^{4} a_i b_i b_i^{'} (mod\ M)$$

解得 $x = 1731$

9 大整数分解

9.1 Pollard's p-1 算法

1. 目的：求合数 N 的数因子
2. 思路及算法：

   (a) 设数字 m, 且 $(p-1)|m$,p 是素数且为 N 的因子
   那么，由费马小定理得：
   $a^m \equiv 1(mod\ p)$
   故有 $p|gcd(a^m - 1, N);$

(b) 令 $m = n!(2 < n < N)$;

(c) 计算 $gcd((a^{n!} mod\ N) - 1, N)$;

(d) 若结果为 1 或者 N,n 自增 1，重复步骤 2-3，否则，$p = gcd((a^{n!} mod\ N) - 1, N)$，即为答案;

3. 代码：

```
def gcd(a,b):
        if b==0:
                return a
        else:
                return gcd(b,a%b)
def Pollard_PM1(N):
        a=2
        for i in range(2,N):
                a=(a^i)%N
                divisor=gcd(a-1,N)
                if divisor!=1 and divisor!=N:
                        return divisor
        return 1
```

9.2 Pollard's rho 算法

1. 目的：求合数 N 的数因子

2. 思路及算法：

1. 假设 N 有两个数因子 x 和 y

令 $x\ y(mod\ N)$

2. 设有函数 $F()$

令 $x = F(x), y = F(F(x))$

$d = gcd(x - y, N)$, 当 $d! = 1\ and\ d! = N$ 时，d 即为答案

3. 伪代码：

```
#伪代码
def PollardRho(N):
        x1 = x2 = Random_ZN_Star(N)
        for i in range(1, 2^(N.nbits()//2)):
                x1 = F(x1, N)
                x2 = F(F(x2, N), N)
                p = gcd(x1 - x2, N)
        if p != 1 and p != N:
                return p
```

设 $F(x) = x^2 - 1$

```
import random
import cmath
def F(x):
        return x^2-1
def gcd(a,b):
        if b==0:
                return a
        else:
                return gcd(b,a%b)
def PollardRho(N):
        i,k=1,2
        x=random.randint(0,N-1)
        y=x
        while True:
                i=i+1
                x=F(x)%N
                divisor=gcd(y-x,N)
                if divisor!=1 and divisor !=N:
                        return divisor
                if i==k:
                        y,k=x,2*k
```

10 离散对数

g 是群 G 的生成元，|G|=p-1,$g^x = y$, 求 x

10.1 Pohlig-Hellman 算法

1. 令
$$p-1 = \prod_{i=1}^{t} p_i^{k_i} = p_1^{k_1} p_2^{k_2} ... p_t^{k_t}$$

2. $g_i = g^{x_i} = g^{(p-1)/p_i^{k_i}}$
得 $x_i = (p-1)/p_i^{k_i}\ p_i^{k_i}$
那么：$y_i = y^{(p-1)/p_i^{k_i}}$ 有 $g^{x_i} = y_i$
由于 $g^{x_i p_i k_i} = e$, $g^{x_i}\ \ p_i^{k_i}\ \ x_i$
故得 $x = x_i (mod\ p_i^{k_i})$

思路：为求 x，构造一组一次同余式组，以 CRT 求解。
那么，每个 x_i 可以用 p-1 的各个因子去构造。

```
#Problem: Given base g, y \in <g>, y = g^x mod p, to find x.
def Pohlig_Hellman(y, g, p):
```

```
    prime_list = factor(p - 1)
    n = len(prime_list)
    q_list = [prime_list[i][0]^prime_list[i][1] for i in range(n)]
    G = [ g^((p - 1) // q_list[i]) for i in range(n) ]
    Y = [ y^((p - 1) // q_list[i]) for i in range(n) ]
    val_list = [discrete_log(Y[i], G[i]) for i in range(n)]
    modulus_list = [q_list[i] for i in range(n)]
    x = crt(val_list, modulus_list) # solve the CRT problem.
    return x
```

10.2 大步小步算法

1. 大步：
   令 $s = \lfloor \sqrt{p} \rfloor$
   那么有 $g^0, g^s, g^{2s} ... g^{\lfloor p/s \rfloor s}$

2. 小步：
   令 y
   有 $y * g, y * g^2 ... y * g^s$

3. 算 x:
   得 $y * g^r = g^x g^r = g^{st}$
   故 $x = st - r(mod\ p)$

4. 代码：

```
# Input: g,y,n where g is the base, y = g^x mod p,
def BGS(g, y, p):
    s = floor(sqrt(p))
    A = [(y*(g^r) % p) for r in range(0, s)] #Baby Step.
    B = [(g^(t*s) % p) for t in range(1, s+1)] #Giant Step.
    r,t = 0,0
    for u in A:
        for v in B:
            if u == v: #Collision.
                r, t = A.index(u), B.index(v)
    return ((t+1)*s - r) % p # Return x
```

10.3 Pollard's rho 算法

1. 设置变量：
   $lefta = g^{lg} y^{ly}; right = g^{rg} y^{ry};$

2. 设置随机函数 $F()$:
   令 $lefta = F(lefta); righta = F(righta)$
   直到出现 $lefta = righta$

3. 那么：

$$g^{lg}y^{ly} = g^{rg}y^{ry}$$

$$g^{lg}g^{xly} = g^{rg}g^{xry}$$

$$lg + xly = rg + xry$$

$$x = (rg - lg)/(ly - ry)$$

4. 伪代码：

```
#Input: y = g^x mod p; Output: x
def PollardRhoDLOG(y):
    lefta, lg, ly = 1, 0, 0 # lefta = g^lg * y^ly
    righta, rg, ry = y, 0, 1 # righta = g^rg * y^ry
    while lefta != righta:
        lefta, lg, ly = F(lefta, lg, ly)
        righta, rg, ry = F(righta, rg, ry)
        righta, rg, ry = F(righta, rg, ry)
    s, t = lg - rg, ry - ly
    if s == 0:
        return 'fail'
    return s * (t-^(1))
```

5. 关于随机函数：
   Pollard suggests：

$$F(x) = \begin{cases} ga & \text{0<=a<q/3} \\ a^2 & \text{q/3 <= } a < 2q/3 \\ ya & 2q/3 <= a < q \end{cases}$$

6. 时间开销：$\theta(\sqrt{q})$