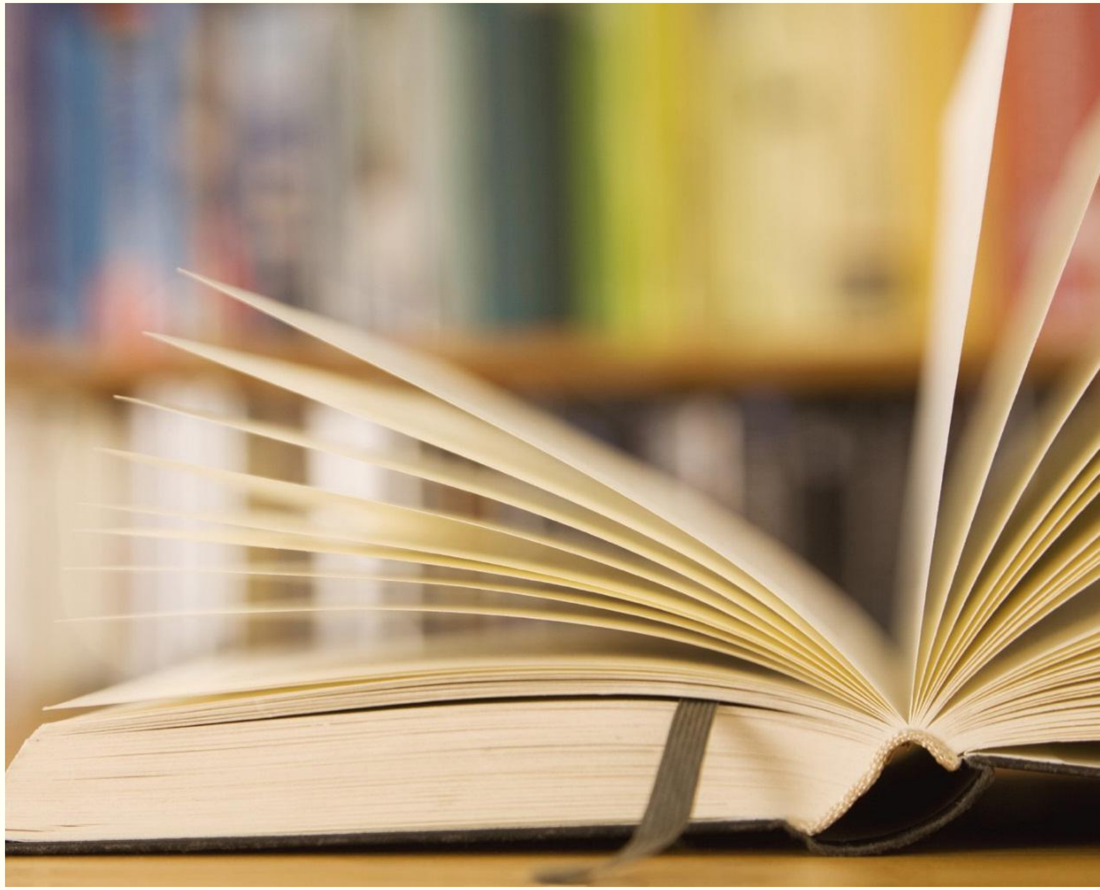
数论2：同余



目录

同余

基本概念

- **除法定理**：

  对于任何整数 a 和任何正整数 m，存在唯一整数 q 和 r，满足 $0 \leq r < m$ 且 $a = qm + r$,

  其中 $q = \left\lfloor \frac{a}{m} \right\rfloor$ 为商, $r = a \bmod m$为余数。

- **余数**：我们把 a 除以 m 所得的余数 r 记作 $a \bmod m$ 。
- **同余**：如果 $a \bmod m = b \bmod m$，即 a, b 除以 m 所得的余数相等，记作：

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

  - 若 $a \equiv b \pmod{m}$，则 $(a, m) = (b, m)$ 。

  - 若 $a \equiv b \pmod{m}$，且 $d|m$，则 $a \equiv b \pmod{d}$ 。

剩余系

- **剩余系**是指模正整数 n 的余数所组成的集合。

- 如果一个剩余系中包含了这个正整数 n 所有可能的余数（一般地，对于任意正整数 n，有 n 个余数：$0,1,2,\dots,n-1$），那么就被称为是模 n 的一个**完全剩余系**，记作 Z_n；而**简化剩余系**就是完全剩余系中与 n 互素的数，记作 Z_n^* 。

- Z_n 里面的每一个元素代表所有模 n 意义下与它同余的整数。例如 $n=5$ 时，Z_5 的元素 3 实际上代表了 $3,8,13,18,\dots,5k+3(k\in N)$ 这些模 5 余 3 的数。我们把满足同余关系的所有整数看作一个**同余等价类**。

- 自然地，在 Z_n中的加法，减法，乘法，结果全部要在模 n 意义下面了
  例如在Z_5中，$3+2=0$，$3\times 2=1$ 。

模运算

- 如果 $a \equiv b(\bmod m)$ 且有 $c \equiv d(\bmod m)$，那么下面的模运算律成立：

$$a + c \equiv b + d(\bmod m)$$

$$a - c \equiv b - d(\bmod m)$$

$$a \times c \equiv b \times d(\bmod m)$$

- 以下用“$\%\, m$”代表 $(\bmod m)$：

$$(a + b)\%\, m = \big((a \,\%\, m) + (b \,\%\, m)\big)\%\, m$$

$$(a - b)\%\, m = \big((a \,\%\, m) - (b \,\%\, m) + m\big)\%\, m$$

$$(a \times b)\%\, m = \big((a \,\%\, m) \times (b \,\%\, m)\big)\%\, m$$

模运算

- 在乘法中，需要注意 $a \bmod m$ 和 $b \bmod m$ 相乘是否会超出 32 位带符号整数所能表示的范围，一般需要用 64 位整数类型即 **`long long`** 保存中间结果，如下所示：

```
int mul_mod(int a,int b,int m){
    a %= m; b %= m;
    return (int)((long long)a * b % m);
}
```

幂取模

- 计算 $a^n \bmod m$ 的值，$a, n, m \leq 10^9$ 。
- 如果简单地使用上述方法进行次 $O(n)$ 乘法，时间复杂度是很不理想的。我们可以利用下面的递归函数来优化：

$$pow(x,n) = \begin{cases} 1 & (n = 0\text{时}) \\ pow(x^2, n/2) & (n\text{为偶数时}) \\ pow(x^2, n/2) * x & (n\text{为奇数时}) \end{cases}$$

- 上述算法称之为“快速幂”，时间复杂度优化到 $O(\log n)$ 。

快速幂

```
int pow_mod(int a,int n,int m){
    if(n == 0) return 1;
    int x = pow_mod(a,n/2,m);
    long long ans = (long long)x * x % m;
    if(n % 2 == 1) ans = ans * a % m;
    return ans;
}
```

快速幂

```
ll qpow(ll a, ll b, ll q)
{
    ll res = 1;// 因为是用乘法模拟乘方，所以res要是1
    while(b) {
        if(b & 1)res = (res * a) % q;
        a = (a * a) % q;// 视情况将 * 换成Mul(龟速乘)
        b >>= 1;
    }
    return res % q;
}
```

龟速乘

- 在模意义下计算乘法，如果C较大(但是不超过 long long范围)，进行乘法的两个数同样很大，直接乘会爆掉（例如快速幂里的乘法)，我们可以用类似快速幂的快速乘计算，时间复杂度为O(logb)。因为慢于O(1)的乘法运算符，所以我们常常把这个叫做龟速乘。经常用与快速幂中代替普通乘法。

```
ll Mul(ll a, ll b, ll p){
    if(b < 0) a = - a, b = - b;
    ll res = 0;// 因为是加法模拟乘法，所以res开始为0
    while(b) {
        if(b & 1) res = (res + a) % p;
        a = (a + a) % p;
        b >>= 1;
    }
 return res;
}
```

O(1)快速乘

- 这样普通的快速幂会变成log^2n。如果需要更快的快速乘，可以用long double数据类型进行计算，复杂度O(1)。

```
inline ll Mul(ll x,ll y,ll p)
{
    if(y < 0) x = - x, y = - y;
    ll z = (long double)x / p * y;
    ll res = (unsigned long long)x * y - (unsigned long long)z * p;
    return (res + p) % p;
}
```

例题： [NOIP2013 D1T1] 转圈游戏

- n 个小伙伴（编号从 0 到 $n-1$）围坐一圈玩游戏。按照顺时针方向给 n 个位置编号，从 0 到 $n-1$。最初，第 0 号小伙伴在第 0 号位置，第 1 号小伙伴在第 1 号位置，……，依此类推。
- 游戏规则如下：每一轮第 0 号位置上的小伙伴顺时针走到第 m 号位置，第 1 号位置小伙伴走到第 $m+1$ 号位置，……，依此类推，第 $n-m$号位置上的小伙伴走到第 0 号位置，第 $n-m+1$ 号位置上的小伙伴走到第 1 号位置，……，第 $n-1$ 号位置上的小伙伴顺时针走到第$m-1$ 号位置。
- 现在，一共进行了 10^k 轮，请问 x 号小伙伴最后走到了第几号位置。
- $1 < n < 10^6$，$0 < m < n$，$1 \le x \le n$，$0 < k < 10^9$。

例题： [NOIP2013 D1T1] 转圈游戏

- 分析：
- 不难发现答案即为 $(x + m * 10^k) \bmod n$ ，化简一下，就成了

$$(x\ +\ m\,(10^k \bmod n) \bmod n) \bmod n$$

- 所以，只需要求出 $10^k \bmod n$ 即可，可以使用快速幂来求解，复杂度 $O(\log k)$ 。

例题： 越狱

- 监狱有连续编号为 $1..n$ 的 n 个房间，每个房间关押一个犯人。有 m 种宗教，每个犯人信仰其中一种。如果相邻房间的犯人信仰的宗教相同，就可能发生越狱。求有多少种状态可能发生越狱。
- 输入两个整数 m 和 n 。对可能越狱的状态数，模 100003 取余。
- 100% 的数据：$1 \le m \le 10^8$，$1 \le n \le 10^{12}$ 。

例题： 越狱

- 分析：
- 所有方案数有：$m^n = m * m^{n-1}$种；
- 所有不发生越狱的方案数为：$m * (m-1)^{n-1}$ 种；
- 发生越狱的方案数为： $m * m^{n-1} - m * (m-1)^{n-1}$
- 分别对 m^{n-1} 和 $(m-1)^{n-1}$ 快速幂即可。

费马小定理

- 若 p 为素数，且 a 和 p 互素，则可以得到

$$a^{p-1} \equiv 1 (\bmod p)$$

- 证明:
  - $p-1$个整数，$a, 2a, 3a, \ldots (p-1)a$ 中没有一个是 p 的倍数，而且没有任意两个模 p 同余。
  - 所以这 $p-1$ 个数对模 p 的同余是 $1,2,3 \ldots (p-1)$ 的排列。
  - 可得：$a * 2a * 3a * \ldots * (p-1)a \equiv 1 * 2 * 3 * \ldots * (p-1) (\bmod p)$
  - 可化简为：$a^{p-1} * (p-1)! \equiv (p-1)! \, (\bmod p)$
  - 即 $a^{p-1} \equiv 1 (\bmod p)$ 得证。

费马小定理

- 一般情况下，在 p 是素数的情况下，对任意整数 a 都有 $a^p \equiv a(mod\, p)$ 。
- 费马小定理应用： p 是素数，a, p互素，则 $a^b \bmod p = a^{\,b \bmod p\,-\,1} \bmod p$
- 如 $p = 5, a = 3$， $3^4 = 81 \equiv 1(mod\, 5)$
- 又如 $3^{2046} = 3^{4\times511+2} \equiv 3^2 (\mathrm{mod}\ 5) \equiv 4(\mathrm{mod}\ 5)$

使用费马小定理来判定素数

- 可多次选取 a 检验 p 是否满足费马小定理，p 为质数的概率随着选取 a 的数量增加而变大。
- 时间复杂度为：选取 k 个 a，判断的过程代价为 $\log p$，总的加起来为 $O(k\log p)$ 。
- 但是这样的算法有缺陷，因为有Carmichael数的存在，可导致上述算法给出一个错误的判断，例如：$561,1105,1729$，这三个数满足费马小定理，但是它们都是合数。
- 这里给出 $1 \sim 10000$ 的Carmichael数：$561,1105,1729,2465,2821,6601,8911$ 。

欧拉定理

- 在 p 不是素数的情况下，可以使用欧拉定理：对于和 m 互素的 a，有：

$$a^{\varphi(m)} \equiv 1 \pmod{m}$$

- 如 $m = 10, a = 3$ 时，$\varphi(10) = 4$， $3^4 = 81 \equiv 1 \pmod{10}$ 。
- 当 m 是素数时，$\varphi(m) = m - 1$，代入可得

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

- 因此欧拉定理也可以看作是费马小定理的加强。

欧拉定理的推论

- a, m 互素 $(m > 1)$，可得:

$$a^b \pmod{m} = a^{b \bmod \varphi(m)} \pmod{m}$$

- 如 $3^{2017} = 3^{4\times504+1} \equiv 3 \pmod{10}$

- 证明:

- 设 $b = q * \varphi(m) + r$，其中 $0 \le r \le \varphi(n)$，即 $r = b \bmod \varphi(m)$。于是:

$$a^b \equiv a^{q*\varphi(m)+r} \equiv \left(a^{\varphi(m)}\right)^q * a^r \equiv 1^q * a^r \equiv a^r \equiv a^{b \bmod \varphi(m)} \pmod{m}$$

欧拉定理的推论2

- a, m 不互素 $(m > 1)$，可得：

$$a^b \pmod m = a^{b \bmod \varphi(m) + \varphi(m)} \pmod m$$

- 证明链接：https://www.cnblogs.com/1024th/p/11349355.html

例题：[POJ 3696] The Luckiest Number

- 给定一个正整数 L ， $L \le 2 * 10^9$ 。问至少多少个 8 连在一起组成的正整数是 L 的倍数?
- 分析:
- x 个 8 连在一起的正整数可写作 $8(10^x - 1)/9$ 。
- 要求出最小的 x ， 满足 $L|8(10^x - 1)/9$ 。设 $d = \gcd(L, 8)$ 。

$$L \mid \frac{8(10^x - 1)}{9} \Leftrightarrow 9L \mid 8(10^x - 1) \Leftrightarrow \frac{9L}{d} \mid (10^x - 1) \Leftrightarrow 10^x \equiv 1 \left(\bmod \frac{9L}{d}\right)$$

- 引理：若 a, n 互素，满足 $a^x \equiv 1 (\bmod n)$ 的最小正整数 x_0 是 $\varphi(n)$ 的约数。

例题：[POJ 3696] The Luckiest Number

- 证明:
- 反证法。假设满足 $a^x \equiv 1 \pmod n$ 的最小正整数 x_0 不能整除$\varphi(n)$ 。
- 设 $\varphi(n) = qx_0 + r\ (0 < r < x_0)$ 。因为 $a^{x_0} \equiv 1 \pmod n$ ，所以 $a^{qx_0} \equiv 1 \pmod n$ 。
- 根据欧拉定理，有 $a^{\varphi(n)} \equiv 1 \pmod n$ ，所以 $a^r \equiv 1 \pmod n$。这与 x_0 最小矛盾。故假设不成立，原命题得证。
- 根据结论，只需求出$\varphi\left(\frac{9L}{d}\right)$ ，枚举它的所有约数，用快速幂逐一检查是否满足条件即可。时间复杂度为 $O\left(\sqrt{L}\log L\right)$ 。

例题：[P4139]上帝与集合的正确用法

- 定义 $a_0=1,a_n=2^{a_{n-1}}$，可以证明 $a_n \bmod p$ 在 n 足够大时为常数，求这个常数。

- 分析：
  - $2^{2^{2^{\cdots\cdots}}} \bmod p = 2^{(2^{2^{\cdots\cdots}}) \bmod \varphi(p) + \varphi(p)} \bmod p$
  - 递归求解

逆元

模 � 意义下乘法的逆

- 如果在 Z_n 中的两元素 a,b 满足 $a*b=1$，比如在 Z_{15}中，$7\times 13=1$，那么我们就说 a,b 互为模 n 意义下乘法的逆元，记作 $a=b^{-1},b=a^{-1}$。
- 在模运算中，除以一个数等于乘上这个数的逆元（如果这个数存在乘法逆元的话）。举例说明，在 Z_5中，$4\div 3=4\times 3^{-1}=4\times 2=3$。
- 剩余系中的每一个元素都对应一个同余等价类，所以 $4\div 3=3$ 的实际含义是：“假定有两个整数 a,b, 满足 a/b 是整数，且 a,b 除以 5 的余数分别是 4 和 3，那么 a/b 除以 5 的余数等于 3”，比如 $a = 9$, $b = 3$时就成立。

逆元

- 当 a, m 互素时，若 $ax \equiv 1 (\bmod m)$，则称 x 是 a 关于模 m 的逆元，记做 a^{-1}。在 $[0, m)$ 的范围内，逆元是唯一的。

- 证明:

反证法，若a有两个逆元$0 < x1 < x2 < m$，即

$$ax_1 \equiv ax_2 \equiv 1 \ (\bmod m)$$

那么有 $m|a(x_2 - x_1)$成立，又由于$(a, m) = 1$，因此

$$m|(x_2 - x_1)$$

其中$0 < x_2 - x_1 < m$，产生了矛盾。

- 将一个整数乘以 a^{-1} 可以与一次乘以 a 的操作抵消，相当于模意义下的除法。因此

$$(a/b) \bmod m = (a * b^{-1}) \bmod m$$

求解逆元

- 求解逆元等价于解方程

$$ax + my = 1$$

- 通过扩展欧几里得算法求逆元的实现如下：

```
int inverse(int a, int b) {
    int x, y;
    extend_gcd(a, b, x, y);
    return x;
}
```

使用欧拉定理求逆元

- $a * a^{\varphi(m)-1} \equiv 1 \pmod{m}$。若$m$是素数：$a * a^{m-2} \equiv 1 \pmod{m}$
- 可得$a^{\varphi(m)-1} \equiv a^{-1} \pmod{m}$，若$m$是素数：$a^{m-2} \equiv a^{-1} \pmod{m}$
- 使用快速幂求解，`powermod(a, m - 2, m)`

```
int powermod(int a, int b, int n){
    int ret = 1;
    while (b) {
        if (b & 1) ret = (long long)ret * a % n;
        a = (long long)a * a % n;
        b >>= 1;
    }
    return ret;
}
```

线性求逆元：递推法

- 如何 $O(n)$ 求 $1\sim n$ 模 p（p为素数）的逆元?
- 假设现在要求 i 的逆元
- 由带余除法可设 $p = iq + r$，则有

$$iq + r \equiv 0 \pmod p$$

- 注意到 p 是质数，因此 r 不为 0，r 的逆元存在。
- 等式两边乘 $i^{-1}r^{-1}$，得到

$$r^{-1} * q + i^{-1} \equiv 0 \pmod p$$

$$i^{-1} \equiv -r^{-1} * q \equiv -(p \bmod i)^{-1} \left\lfloor \frac{p}{i} \right\rfloor \pmod p$$

```
for (inverse[1] = 1, i = 2; i <= n; ++i)
    inverse[i] = inverse[p%i]*(p - p/i) % p;
```

线性求逆元：倒推法

- 先求 $n!$ 的逆元（可以使用扩展欧几里得算法,或者快速幂），然后利用

$$((k-1)!)^{-1} \equiv k*(k!)^{-1} \pmod p$$

- 倒推求出 $1!\ldots(n-1)!$ 的逆元
- 再利用

$$k^{-1} \equiv (k-1)!*(k!)^{-1} (\bmod p)$$

- 就可以求出$1\ldots n$的逆元了

例题：[POJ 1845] Sumdiv

- 求 A^B 的所有约数之和 $\mathrm{mod}\ 9901$ $(1 \le A, B \le 5 * 10^7)$。
- 分析：
- 把 A 分解素因数，表示为 $p_1^{c_1} p_2^{c_2} \ldots p_n^{c_n}$，由“约数之和”得知， A^B 的所有约数之和为：

$$\left(1+p_1+p_1^2+\ldots+p_1^{B*c_1}\right)*\left(1+p_2+p_2^2+\ldots+p_2^{B*c_2}\right)*\ldots*\left(1+p_n+p_n^2+\ldots+p_n^{B*c_n}\right)$$

- 上式的每一项都是一个等比数列。以第一项为例：

$$\left(1+p_1+p_1^2+\ldots+p_1^{B*c_1}\right)=\left(p_1^{B*c_1+1}-1\right)/(p_1-1)$$

- 可以用快速幂计算分子和分母取模 。因为 9901 是素数，只要 p_i-1 不是 9901 的倍数，就只需要计算 p_i-1 的乘法逆元 inv，用乘 inv 代替除以 (p_i-1)，直接计算等比数列求和公式即可。
- 特别的，若 p_i-1 是 9901 的倍数，那么此时乘法逆元不存在，但是 $p_i \bmod 9901 = 1$，所以：

$$\left(1+p_i+p_i^2+\ldots+p_i^{B*c_i}\right)\equiv 1+1+1^2+\ldots+1^{B*c_i}\equiv B*c_i+1 \pmod{9901}$$

威尔逊定理

- 若正整数 p 为质数，那么：

$$(p-1)! \equiv p-1 (\bmod p)$$

形式的确十分的简单，那么我们该如何证明它呢？

- 首先，由于 p 是质数，那么 1 ~ p – 1中的值一定存在模 p 意义下的乘法逆元
- 那么对于任意的 x (2 ≤ x ≤ p – 2) 里必定包含了它的逆元，乘起来结果就为 1 。
- 但是稍加计算后发现 1 的逆元和 p – 1 的逆元都是他们本身，它们就没有被消掉，最后的结果也就是它们的乘积 p – 1 了

例题

- 给你一个正整数 n 求

$$(\mathrm{n}-1)!\ \bmod n$$

- 很显然，在输入的 n 为质数时，套威尔逊定理的即可，结果为 n – 1 。那如果 n 为合数呢？我们先来看如果 n 是完全平方数的情况：
- 只要保证 $2\sqrt{n} \leq n-1$即可，因为 (n – 1) ! = 1 × ⋯ × k × ⋯ × 2 k × ⋯ × (n – 1) ，余数恰好为 0 。唯一不符合这个条件的值是 4 ，代入计算 (4 – 1) ! mod 4 = 2
- 那如果 n 不是完全平方数呢？由于 n 是合数，所以必定存在一对 x , y (1 < x , y < n – 1) 使得 x y = n 。(n – 1) ! = 1 × ⋯ × x × ⋯ × y × ⋯ × (n – 1) ，余数同样也是 0 。如果 n = 1 ，同样直接代入计算，结果也为 0。
- 如果 n = 1 ，同样直接代入计算，结果也为 0 。

例题 CERC2017 F-Faulty Factorial

- 给三个数，n,p,r（p > r）,在n的阶乘中找到一个数k（2<=k <= n）,将k换成比k小的数v（1<=v<k），使得换完之后的阶乘结果res对p取模结果为r，即res≡r(mod p)。
- $2 \leq n \leq 10^{18}$, $2 \leq p < 10^{7}$, $0 \leq r < p$

例题 CERC2017 F-Faulty Factorial

- 讨论n，p的大小。
- n>=2p时，不管修改哪一个值，得到的结果总是p的倍数，r只能为0，所以r = 0时，修改任意一个值（例如使2改为1）即可，r ≠ 0，输出-1
- p<=n<2p时，若r = 0，则修改的一定不是p，找到任意一个不是p的值改为1即可，找不到输出-1（例如n = 2，p = 2，找不到）；若r ≠ 0，则修改的一定是p，至于修改成什么，可以枚举小于p的值，找到一个即可。
- n<p时，这时候用到数论的知识，根据res≡r(mod p)，可以整理成v≡ $\frac{r*k}{n!}$(mod p),p是素数，根据费马小定理，求逆元v ≡ ((r * k % p) * ksm(n!, p-2, p)) % p ,枚举k,若得到的v>=1&&v<k,则找到一组，若找不到输出-1

扩展欧几里得算法

裴蜀定理（Bézout 定理）

- 对任何整数 a, b，关于未知数 x 和 y 的线性不定方程（称为裴蜀等式）：

$$ax + by = c$$

方程有整数解（当且仅当 c 是 $\gcd(a, b)$ 的倍数）。裴蜀等式有解时必然有无穷多个解。

- $ax + by = c$ 有解的充要条件为 $\gcd(a, b)|c$ 。
- 一定存在 x, y 满足 $ax + by = \gcd(a, b)$ 。
- 推论：a, b 互素等价于 $ax + by = 1$ 有解。

例题：[BZOJ 1441] Min

- 给出 n 个数 $(A_1 \dots A_n)$，现求一组整数序列 $(X_1 \dots X_n)$ 使得 $S = A_1X_1 + \dots A_nX_n > 0$，且 S 的值最小。

- 题解：gcd 和裴蜀定理

- $\gcd(a, b)$ 就是最小的可以表示成 $ax + by$ 的正整数。
- 所以我们直接对于所有读入的 a 求 gcd 即可

$$\gcd(a, b, c) = \gcd(\gcd(a, b), c)$$

扩展欧几里得算法

- 根据欧几里得算法，可得：

$$ax + by \ = \ \gcd(a, b) \ = \ \gcd(b, a \bmod b) \ = \ bx' + (a \bmod b)y'$$

- 其中 $a \bmod b$ 为 $a - \left\lfloor \frac{a}{b} \right\rfloor b$，代入上式后，可得：

$$bx' + (a \bmod b)y' \ = \ bx' + (a - \left\lfloor \frac{a}{b} \right\rfloor b)y' = ay' + b(x' - \left\lfloor \frac{a}{b} \right\rfloor y')$$

- 可以得出x, y和x', y' 的关系

$$x = y', y = x' - \left\lfloor \frac{a}{b} \right\rfloor y'$$

- 边界情况分析：$ax' + by' = d$（$d = \gcd(a, b)$），当 $b = 0$ 时，a 为 $\gcd(a, b)$，当且仅当 $x' = 1$ 时等式成立。y' 可以为任何值，为方便起见，设 $y' = 0$ 。

- 根据 $x = y', y = x' - \left\lfloor \frac{a}{b} \right\rfloor * y'$，可以倒推出 x 和 y 的多组解。

扩展欧几里得算法

- 举例：$15x+9y=3$，根据 $x=y', y=x'-\left\lfloor\frac{a}{b}\right\rfloor y'$，$a,b,x,y$在不同时刻的值如下所示：

(x,y)自下而上

a	b	x	y
15	9	-1	2
9	6	1	-1
6	3	0	1
3	0	1	0

(a,b)自上而下

扩展欧几里得算法

```
int extend_gcd(int a, int b, int &x, int &y) {
    if (b == 0) {
        x = 1; y = 0;
        return a;
    }
    else {
        int ret = extend_gcd(b, a % b, y, x);
        y -= x * (a / b);
        return ret;
    }
}
```

扩展欧几里得算法

- $ax + by = c$ 有无穷组解，扩展欧几里得算法计算出来的解是其中一个特解 (x_0, y_0)，可以以下方式来获得其他解。
- 假如把方程的所有解按 x 的值从小到大排序，特解 (x_0, y_0) 的下一组解 (x_1, y_1) 可以表示为 $(x_0 + d_1, y_0 + d_2)$，其中 d_1 是符合条件的最小的正整数，则满足：

$$a(x_0 + d_1) + b(y_0 + d_2) = c$$

- 由于$ax + by = c$，所以$ad_1 + bd_2 = 0$，即

$$\frac{d_1}{d_2} = -\frac{b}{a} = -\frac{\left(\frac{b}{\gcd(a,b)}\right)}{\left(\frac{a}{\gcd(a,b)}\right)}$$

- 因此方程 $ax + by = c$ 的一般解可以表示为：

$$x = x_0 + k\left(\frac{b}{\gcd(a,b)}\right), y = y_0 - k\left(\frac{a}{\gcd(a,b)}\right) \quad (k \in Z)$$

扩展欧几里得算法

- 求 $6x + 5y = 2$ 的通解
- 扩展欧几里得算法可得特解为 $(2, -2)$
- 因此方程 $ax + by = c$ 的一般解可以表示为:

$$x = x_0 + k\left(\frac{b}{\gcd(a, b)}\right) = 2 + 5k$$

$$y = y_0 - k\left(\frac{a}{\gcd(a, b)}\right) = -2 - 6k \quad (k \in Z)$$

线性同余方程

线性同余方程

- 形如$ax \equiv c \pmod m$的方程，称为线性同余方程，其中“线性”表示方程的未知数 x 的次数是一次。显然，可以简单的尝试，依次用 $x = 0,1,\ldots,m-1$ 来代入该方程，找出其中在模 m 时满足该方程的整数 x 。但时间复杂度取决于 m 的大小，效率不高。
- $ax \equiv c \pmod m$ 可以转化为 $ax + my = c$ ，即可将线性同余方程转换为扩展欧几里得算法求解。根据裴蜀定理， $ax + my = c$ 的有解条件为 $\gcd(a,m)|c$，否则方程无解。
- 在有解时，使用扩展欧几里得算法求出一组整数解，满足 $ax_0 + my_0 = \gcd(a,m)$ 。
- 在模 m 的完全剩余系 $\{0,1,\ldots,m-1\}$ 中，恰有 d 个解，第一个解为 x_0 ， 其余 $d-1$ 个解可以通过以下式子得到，即：

$$x_i = (x_0 + i\left(\frac{m}{d}\right)) \bmod m \; (1 \le i \le d-1)$$

线性同余方程

- 使用欧拉定理求解：
- 令 $d = \gcd(a, m)$，若 $d \nmid c$ 则方程组无解，否则方程组可变为：

$$a'x \equiv c' \pmod{m'},\ a' = \frac{a}{d}, m' = \frac{m}{d}, c' = \frac{c}{d}, \gcd(a', c') = 1$$

$$x \equiv a'^{\varphi(m')-1} c' \pmod{m'}$$

$$x \equiv a'^{\varphi(m')-1} c' + km' \pmod{m}, (0 \le k < d)$$

线性同余方程

- 在方程 $3x \equiv 2 \pmod 6$ 中，$d = \gcd(3,6) = 3$，3 不整除 2，因此方程无解。
- 在方程 $5x \equiv 2 \pmod 6$ 中，$d = \gcd(5,6) = 1$，1 整除 2，因此方程在 $\{0,1,2,3,4,5\}$ 中恰有一个解: $x = 4$。
- 在方程 $4x \equiv 2 \pmod 6$ 中，$d = \gcd(4,6) = 2$，2 整除 2，因此方程在 $\{0,1,2,3,4,5\}$ 中恰有两个解: $x = 2$，$x = 5$。

例题：线性组合

- 对应整数数列$A_1, A_2, \ldots, A_n$ 是否存在$X_1, X_2, \ldots, X_n$，使得$A_1X_1 + A_2X_2 + \ldots + A_nX_n = C$，其中 $\gcd(A_1, A_2, \ldots, A_n) | C (n \geq 2)$ 。如请找出一组整数解(x_1, x_2, x_3, x_4)满足$12x_1 + 24x_2 + 18x_3 + 15x_4 = 3$ 。

例题：线性组合

- 预处理：

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

$$\gcd(12,24,18) = \gcd(\gcd(12,24), 18) = \gcd(12,18) = 6$$

- 求解方程：

$$\gcd(12,24,18)y_1 + 15x_4 = 3\text{即}6y_1 + 15x_4 = 3$$

- 利用扩展欧几里得算出一组特解：

$$y_1 =- 2, x_4 = 1$$

- 继续列方程：

$$12x_1 + 24x_2 + 18x_3 = 6y_1 =- 12$$

- 先不求解，而是求解

$$\gcd(12,24)y_2 + 18x_3 =- 12$$

即

$$12y_2 + 18x_3 =- 12$$

例题：线性组合

- 同样利用扩展欧几里得算出一组特解：

$$y_2 = 2, x_3 =- 2$$

- 最后求解

$$12x_1 + 24x_2 = 12y_2 = 24$$

  得到特解

$$x_1 = 2, x_2 = 0$$

- 最后得到一组整数解 $(2,0,-2,1)$，可以根据推导过程写出程序。

线性同余方程组（模互素）

- 考虑形如$x \equiv a_i \pmod{m_i}$的若干方程联立得到的方程组，如：

$$\begin{cases} x \equiv 2 \pmod{3} \text{ ...... } (1) \\ x \equiv 3 \pmod{5} \text{ ...... } (2) \\ x \equiv 5 \pmod{7} \text{ ...... } (3) \end{cases}$$

- 下面是一种可行的解法：
  - 由(1)设$x = 3y + 2$，代入(2)得到 $3y + 2 \equiv 3 (\mathrm{mod}\ 5)$，解得 $y \equiv 2 (\mathrm{mod}\ 5)$
  - 设 $y = 5z + 2$，代入(3)得到 $3(5z + 2) + 2 \equiv 5 (\mathrm{mod}\ 7)$，解得 $z \equiv 4 (\mathrm{mod}\ 7)$
  - 设 $z = 7k + 4$，则 $x = 3(5(7k + 4) + 2) + 2 = 105k + 68$
  - 因此 $x \equiv 68 (\mathrm{mod}\ 105)$

中国剩余定理

- 对于同余方程组 $x \equiv a_i (\mathrm{mod}\ m_i)\ (i = 1 \ldots n)$，若 m_i **两两互素**，则 x 在 $\mathrm{mod}\ M, (M = m_1 m_2 \ldots m_n)$ 下有唯一解 。
- 中国剩余定理同时也给出了构造解的方法，令 $M = m_1 m_2 \ldots m_n$，$M_i = \frac{M}{m_i}$，显然 $(M_i, m_i) = 1$，所以 M_i 关于模 m_i 的逆元存在。
- 把逆元设为t_i，于是有：

$$M_i t_i \equiv 1\ (\mathrm{mod}\ m_i), M_i t_i \equiv 0\ (\mathrm{mod}\ m_j)\ (j \neq i)$$

- 进一步：

$$a_i M_i t_i \equiv a_i (\mathrm{mod}\ m_i), a_i M_i t_i \equiv 0\ (\mathrm{mod}\ m_j)\ (j \neq i)$$

- 解为

$$x \equiv \sum_{i=1}^{n} a_i M_i t_i\ \ (\mathrm{mod}\ M)$$

中国剩余定理

- 今有物不知其数，三三数之剩二，五五数之剩三，七七数之剩二，问物几何?

$$\begin{cases} x \equiv 2 \pmod 3) \ldots\ldots (1) \\ x \equiv 3 \pmod 5) \ldots\ldots (2) \\ x \equiv 2 \pmod 7) \ldots\ldots (3) \end{cases}$$

- $a_i = \{2, 3, 2\}, m_i = \{3, 5, 7\}, M = 3 \times 5 \times 7 = 105$
- $M_i = \{\frac{105}{3}, \frac{105}{5}, \frac{105}{7}\} = \{35, 21, 15\}$
- $t_i = \{\mathrm{inverse}(35, 3), \mathrm{inverse}(21, 5), \mathrm{inverse}(15, 7)\} = \{2, 1, 1\}$
- $x \equiv 2 \times (35 \times 2) + 3 \times (21 \times 1) + 2 \times (15 \times 1)$
- $x \equiv 233 \equiv 23 \pmod{105}$
- 通解 $x = 23 + 105k (k \in Z)$

中国剩余定理

```
// Chinese Remainder Theorem
int CRT(const int a[], const int m[], int n) {
    int M = 1, ret = 0;
    for (int i = 1; i <= n; ++i) M *= m[i];
    for (int i = 1; i <= n; ++i) {
        int Mi = M / m[i], ti = inv(Mi, m[i]);
        ret = (ret + a[i] * Mi * ti) % M;
    }
    return ret;
}
```

```
// 利用extend_gcd求逆元
int inverse(int a, int b) {
    int x, y;
    extend_gcd(a, b, x, y);
    return x;
}
```

线性同余方程组（模不互素）

$$\begin{cases} x \equiv \mathrm{c}_1 (\mathrm{mod}\ m_1) \\ x \equiv c_2 (\mathrm{mod}\ m_2) \\ \quad\cdots \\ x \equiv c_n (\mathrm{mod}\ m_n) \end{cases} \quad (m_1, m_2, \ldots, m_n\text{不互素})$$

- 仅考虑方程数量为 2 的情况（方程数量> 2时可以迭代求解）

$$\begin{cases} x \equiv \mathrm{c}_1 (\mathrm{mod}\ m_1) \\ x \equiv c_2 (\mathrm{mod}\ m_2) \end{cases}$$

- 设$y = x - \mathrm{c}_1$，则

$$\begin{cases} y \equiv 0 (\mathrm{mod}\ m_1) \\ y \equiv c_2 - c_1 (\mathrm{mod}\ m_2) \end{cases}$$

- 设 $d = \mathrm{gcd}(m_1, m_2)$， 若 $d \nmid (c2 - c1)$则方程组无解。

线性同余方程组（模不互素）

- 否则：

$$\begin{cases} y' \equiv 0 (\bmod m_1') \\ y' \equiv c' (\bmod m_2') \end{cases}$$

- 其中 $y' = \frac{y}{d}, c' = \frac{c2-c1}{d}, m_1' = \frac{m_1}{d}, m_2' = \frac{m_2}{d}$ 且 $\gcd(m_1', m_2') = 1$。
- 可得

$$y' \equiv k m_1' \equiv c' (\bmod m_2')$$

- 用欧拉定理解得
- 所以 $y' \equiv c' {m_1'}^{\varphi(m_2')} (\bmod m_1' m_2')$
- 代入得 $x \equiv d c' {m_1'}^{\varphi(m_2')} + c_1 (\bmod d m_1' m_2')$
- 至此，两个同余方程合并成了一个同余方程。迭代若干次可得到原方程组的解。

例题：[POJ 2891] Strange Way to Express Integers

- 给定 $2n$ 个正整数 $a_1, a_2, \ldots, a_n$ 和 $m_1, m_2, \ldots, m_n$ ，求出一个最小的正整数 x，求满足 $x \equiv a_i \pmod{m_i}$, $i \in [1, n]$，或者给出无解。
- 题中 m_i 不一定两两互素，中国剩余定理不再适用。
- 假设已经求出了前 $k-1$ 个方程构成的方程组的一个解 x。记 $m = \prod_{i=1}^{k-1} m_i$ ，则 $x + i * m$ 是前 $k-1$ 个方程的通解。
- 考虑第 k 个方程，求出一个整数 t，使得 $x + t * m \equiv a_k \pmod{m_k}$ 。该方程等价于 $x + t * m \equiv a_k - x \pmod{m_k}$ ，其中 t 是未知量。
- 这就是一个线性同余方程，可以用扩展欧几里得算法判断是否有解，并求出它的解。如有解，则 就是前 k 个方程构成的方程组的一个解。
- 综上所述，使用 n 次扩展欧几里得算法，就可求出方程组的解。

高次同余方程

第一类高次同余方程

- 已知 a,p 互素，求解同余方程：

$$a^x \equiv b(\operatorname{mod} p)$$

- 因为 a,p 互素，所以可以在模 p 意义下执行关于 a 的乘法、除法运算。
- 令 $x = kt - m$，其中 $t = \lceil\sqrt{p}\rceil, 0 \le k \le t$， 则有 $a^{kt-m} \equiv b(\operatorname{mod} p)$, 即 $a^{kt} \equiv a^m b(\operatorname{mod} p)$。
- 对于所有的 $m \in [0, t-1]$，把 $a^m b(\operatorname{mod} p)$ 的结果存入Hash表。
- 接着枚举 $k \in [0, t]$，计算$a^{kt}(\operatorname{mod} p)$，并在Hash表中查找是否有对应的 m 值。若有，即找到了满足条件的 k 和 m。
- 时间复杂度$O(\sqrt{p})$，这种算法称为Baby Step Giant Step。

Baby Step Giant Step

```
int baby_step_giant_step(int a, int b, int p){
	map<int, int> hash;
	hash.clear();
	b %= p;
	int t = (int)sqrt(p) + 1;
	for(int j = 0; j < t; j++){
		int val = (long long)b * power(a, j, p) % p; //b*a^j;
		hash[val] = j;
	}
	a = power(a, t, p); //a^t
	if(a == 0) return b == 0 ? 1 : -1;
	for(int i = 0; i <= t; i++){
		int val = power(a, i, p); //(a^t)^i
		int j = hash.find(val) == hash.end() ? -1 : hash[val];
		if(j >= 0 && i * t - j >= 0) return i * t - j;
	}
	return -1;
}
```

第二类高次同余方程

- 求解同余方程：

$$x^k \equiv b \pmod p$$

原根

- 设 p 是质数，若 $a^0, a^1, a^2, \dots, a^{p-2}$ 互不相等（a 取遍 $1,2,3, \dots,\ p-1$）,则称 a 是 p 是一个原根。
- 如何快速判断 a 是否 p 的原根？由于费马小定理成立，因此方程 $a^x \equiv 1 (\mathrm{mod}\, p)$ 的一个解是 $x = p - 1$，所以它的最小整数解 $x_{\min} | (p-1)$。
- 若 $x_{\min} = (p-1)$则 a 是 p 的原根。因此逐个尝试 $p-1$的约数即可。

第二类高次同余方程

- 求解同余方程：

$$x^k \equiv b(\mathrm{mod}\, p)$$

- 假设 a 是 p 的一个原根，通过第一类高次同余方程的解法求得 $b \equiv a^m(\mathrm{mod}\, p)$。
- 又假设 $x \equiv a^y\ (\mathrm{mod}\, p)$，则

$$a^{ky} \equiv a^m(\mathrm{mod}\, p)$$

$$a^{ky-m} \equiv 1(\mathrm{mod}\, p)$$

- 根据原根的性质有

$$ky - m \equiv 0\ (\mathrm{mod}\ (p-1))$$

- 若 k 与 $p-1$ 不互质，则有可能有多解或者无解。

同余方程求解小结

- 线性同余方程：

  先处理不互质的情况，然后通过欧拉定理求解。

- 线性同余方程组：

  先分别解每个线性同余方程，然后每次合并两个方程求解。

- 第一类高次同余方程（取对数）：

  BSGS

- 第二类高次同余方程（开根号）：

  表示成原根的若干次幂的形式后解线性方程。