数论基础

--COCO

1 整除

定义： 整数$a, b \in \mathbb{Z}$, 且$a \neq 0$,若$\exists q$,使得$b = aq$，则称b可被a整除（或a能整除b），记作$a \mid b$；且称b是a的倍数，a是b的约数（或因子、因数）。若a不能整除b，则记作$a \nmid b$

定理

- 1. $a \mid b$，且$b \mid c$，则$a \mid c$
  - **证明：** $\because a \mid b$, 且$b \mid c$; 令$b = pa, c = qb$，$\therefore c = pqa$，根据整除定义即$a \mid c$
- 2. $a \mid b$, 且$a \mid c$, 则$\forall x, y$, 满足$a \mid bx + cy$
  - **证明：** $\because a \mid b$且$a \mid c$, 令$b = pa, c = qa$, 则$bx + cy = pax + qay = (px + qy)a$, $\therefore a \mid bx + cy$
- 3. 若$a \neq 0, b = aq + c$，那么$a \mid b$ 的充分必要条件是$a \mid c$
  - **充分性证明：** $\because a \mid b$, 令$b = pa$, 则$c = (p - q)a$, $\therefore a \mid c$
  - **必要性证明：** $\because a \mid c$, 令$c = pa$, 则$b = (p + q)a$, $\therefore a \mid b$

2 同余

$\forall a, b \in \mathbb{Z}$, 且$a \neq 0, \exists p, c$，使得等式$b = pa + c, 0 \leqslant c \leqslant a - 1$成立

- **取模运算**
  b对a取模的结果即b除以a的余数，符号表示为 $b \mod a$，计算值为$b - \lfloor \frac{b}{a} \rfloor * a$。
- **同余**
  $m \geqslant 1, a, b \in \mathbb{Z}$, 若$m \mid (a - b)$，则称a与b关于模m同余，符号表示为$a \equiv b(mod\ m)$。

2 同余

- **性质**
  - **1. 自反性：** $a \equiv a(mod\ m)$
  - **2. 对称性：** $a \equiv b(mod\ m)$则$b \equiv a(mod\ m)$
  - **3. 传递性：** $a \equiv b(mod\ m)$且$b \equiv c(mod\ m)$则$a \equiv c(mod\ m)$
    - **证明：** $\because a \equiv b(mod\ m)$且$b \equiv c(mod\ m)$则根据同余定义$m \mid (a-b), m \mid (b-c)$,再根据整除的定理2,$m \mid [(a-b)+(b-c)]$, 即$m \mid (a-c)$,再由同余定义得$a \equiv c(mod\ m)$
  - **4. 同余式相加：** $a \equiv b(mod\ m)$且$c \equiv d(mod\ m)$则$a \pm c \equiv b \pm d(mod\ m)$
    - **证明：** $\because a \equiv b(mod\ m)$且$c \equiv d(mod\ m) \therefore m \mid (a-b), m \mid (c-d) \therefore m \mid [(a-b)+(c-d)], m \mid [(a+c)-(b+d)] \therefore a+c \equiv b+d(mod\ m)$同理也能证明$a-c \equiv b-d(mod\ m)$
  - **5. 同余式相乘：** $a \equiv b(mod\ m)$且$c \equiv d(mod\ m)$则$ac \equiv bd(mod\ m)$
    - **证明：** $\because a \equiv b(mod\ m)$且$c \equiv d(mod\ m) \therefore m \mid (a-b), m \mid (c-d) \therefore m \mid [(a-b)c+(c-d)b] \therefore m \mid (ac-bd) \therefore ac \equiv bd(mod\ m)$
  - **6. 除法：** $ac \equiv bc(mod\ m)$则$a \equiv b(mod\ \frac{m}{gcd(c,m)})$，$gcd(c,m)$就是c和m的最大公约数
    - **证明：** 令$d = gcd(c,m)$，则令$c = pd, m = qd$,满足$gcd(p,q) = 1 \because ac \equiv bc(mod\ m) \Rightarrow qd \mid (a-b)pd \Rightarrow q \mid (a-b)p \because gcd(p,q) = 1 \therefore q \mid (a-b) \therefore a \equiv b(mod\ q)$，q即$\frac{m}{gcd(c,m)}$
  - **7. 幂运算：** $a \equiv b(mod\ m)$则$a^n \equiv b^n(mod\ m)$

3 GCD（最大公约数）

- **定义：** a,b的公共的约数中最大的约数，记作$gcd(a,b)$， $gcd(a,b)=max\{d,d\mid a且d\mid b\}$
- **性质：**
  - 1. $gcd(a,b)=gcd(b,a)$
  - 2. $gcd(a,b)=gcd(a,a\pm b)$
  - 3. $gcd(a,b)=gcd(a\pm mb)$
  - 4. $gcd(a,b)=gcd(b,a\%b)$
    - **证明：** $gcd(b,a\%b)=gcd(b,a-\lfloor\frac{a}{b}\rfloor*b)$，根据性质1、3可得gcd(a,b)=gcd(b,a%b)
  - 5.$gcd(ma,mb)=m*gcd(a,b)$
    - **证明：** 令$d=gcd(a,b)$,则$d|a,d|b$,满足$a=pd,b=qd$且$gcd(p,q)=1\therefore gcd(ma,mb)=gcd(mdp,mdq)=md$

3 GCD

辗转相除法（欧几里德法）

- **原理：**
  $gcd(a, b) = gcd(b, a\%b)$
  当b=0时，gcd(a,b)=a，否则根据公式继续迭代。
- **代码实现：**

```
int gcd(int a, int b){
    return b == 0 ? a : gcd(b, a%b);
}
```

4 拓展欧几里得（gcd(a,b)=ax+by,a||b≠0）

- **原理:**

令$d = gcd(a,b)$
$\because ax + by = gcd(a,b)$
$\therefore bx^{'} + (a\%b)y^{'} = gcd(b, a\%b)$
$\therefore bx^{'} + (a\%b)y^{'} = ax + by$
$\therefore ay^{'} + b(x^{'} - \lfloor \frac{a}{b} \rfloor * y^{'}) = ax + by$要使任意a,b都满足此等式,
则$x = y^{'}, y = x^{'} - \lfloor \frac{a}{b} \rfloor * y^{'}$
当b=0时，ax=a,则x=1,y=0;
否则，令用$gcd(a,b) = gcd(b, a\%b)$递归求解，回溯时利用两式之间的关系，更新x,y，可求得方程的一组整数解。

- **代码实现:**

```
int exgcd(int a, int b, int &x, int &y){
    if(!b) {x = 1; y = 0; return a;}
    int d = exgcd(b, a % b, x, y);
    int z = x; x = y; y = z - (a / b) * y;
    return d;
}
```

4 拓展欧几里得（gcd(a,b)=ax+by,a||b≠0）

- **原理:**

  令$d = gcd(a,b)$

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

  $\therefore bx' + (a\%b)y' = gcd(b, a\%b)$

  $\therefore bx' + (a\%b)y' = ax + by$

  $\therefore ay' + b(x' - \lfloor \frac{a}{b} \rfloor * y') = ax + by$要使任意a,b都满足此等式，

  则$x = y', y = x' - \lfloor \frac{a}{b} \rfloor * y'$

  当b=0时，ax=a,则x=1,y=0；

  否则，令用$gcd(a,b) = gcd(b, a\%b)$递归求解，回溯时利用两式之间的关系，更新x,y，可求得方程的一组整数解。

- **代码实现:**

```
int exgcd(int a, int b, int &x, int &y){
    if(!b) {x = 1; y = 0; return a;}
    int d = exgcd(b, a % b, x, y);
    int z = x; x = y; y = z - (a / b) * y;
    return d;
}
```

练习1：青蛙的约会
练习2：五指山

5 素数及其判定

- **素数**（也称**质数**）： 一个正整数n是素数当且仅当只能被1和其自身整除，否则就称n是**合数** 。
  注: 1既不是素数也不是合数，最小的素数是2
- **素数判定**
  - **1. 暴力枚举:**
    - **原理：** 循环遍历i从2到$\sqrt{n}$，只要存在 $n\%i=0$即可判定n不是素数，这样每次判定时间复杂度都是$O(\sqrt{n})$
    - 代码实现

```
bool isprime(int n){
    for(int i = 2; i * i <= n; i ++){
        if(n % i == 0) return false;
    }
    return true;
}
```

- **2. 埃拉托斯特尼筛法，简称埃氏筛:**
  是一种用来求自然数n以内的全部素数的方法 。
  - **原理:** 一个合数必然可以表示一个质数和另一个数相乘。对于一个素数p,那么p的倍数2p,3p,…kp,…必然都是合数。时间复杂度为$O(nlog(n))$
  - 实现代码:

```
/*
    定义数组: isprime[i], 表示数组i是否时素数

*/
const int maxn = 1e5 + 100;
int m;
bool isprime[maxn];
int p[maxn];
void sieze(int n)
{
    m = 0; memset(isprime, true, sizeof(isprime));
    for(int i = 2; i <= n; i ++){
        if(isprime[i]){///表明 i 就是素数
            p[++ m] = i;
            for(int j = 2 * i; j <= n; j += i){
                isprime[j] = false;
            }
        }
    }
}
```

  - **不足:** 仔细分析可以看出，这种方法筛出n以内的所有素数还是有不足之处的。原因在于每个合数会被其每一个质因子筛到一次。因此就有了接下来的线性筛。

- **3. 线性筛**
  它能在$O(n)$的时间复杂度内筛选出n以内的所有素数。
  - **原理：**
    线性筛能保证n以内的任何一个数只能被他的最小质因子筛一次。
    一个数n可以表示成$n = Factory_{max} * p$
    $Factory_{max}$是除n以外的最大因子，p是n的质因子，满足p小于等于 $Factory_{max}$ 的所有质因子。
    用数组v[i]表示i是否是素数，v[i]为false时表示i是素数，
    然后便利之前筛出的所有素数，v[i*p[j]]则不是素数，标记为true。 i % p[j] 为0时，跳出第二层循环，以此保证每个数只被他的最小质因子筛一次。
    当i %p[j]为0时，p[j]已经是i的最小值质因子，且p[j]时i*p[j]的最小值质因子，p[j+1]大于i的最小质因子，也就不是i*p[j]的最小质因子。
  - **代码实现：**

```
const int maxn = 1e5 + 100;
int m;
bool v[maxn];
int p[maxn];
void sieze(int n)
{
    m = 0; memset(v, false, sizeof v);
    for(int i = 2; i <= n; i ++){
        if(!v[i]) {
            p[++ m] = i;
        }
        for(int j = 1; j <= m && i * p[j] <= n; j ++){
            v[i * p[j]] = true;
            if(i % p[j] == 0) break;//这条语句很关键
        }
    }
}
```

6 快速幂

- **问题描述：** 计算a^n为例
- **问题分析：** 显然$O(n)$暴力是不能解决问题的，下面我们介绍一种算法（快速幂），它能在$O(logn)$时间复杂度内解决此问题。
- **快速幂原理:**

将正整数n二进制拆分成$n = b_{n-1}b_{n-2}...b_0$，$b_i \in \{0, 1\}$

即$n = 2^{n-1}b_{n-1} + 2^{n-2}b_{n-2} + ... + 2^0 b_0$

$$\therefore a^n = a^{2^{n-1}b_{n-1}+2^{n-2}b_{n-2}+...+2^0 b_0}$$
$$= a^{2^{n-1}b_{n-1}} * a^{2^{n-2}b_{n-2}} * ... * a^{2^0 b_0}$$

因此我们可以设置一个res（初始为1）和一个base（初始值为a），也即a^{0},在对n进行二进制拆分过程中，当第i位（从右往左第i位）为1时，则将res乘上base,并且base每次都乘上自己。其实在整个过程中，$base_i = a^{2^i}$。时间复杂度$O(log(n))$

```
int ksm(int a, int n, int mod)
{
    int res = 1;
    while(n){
        if(n & 1) res = res * a % mod; //二进制拆分n第i位(从右往左)为1
        a = a * a % mod;
        n >>= 1;
    }
     return res;
}
```

7 逆元

- **定义：** 整数a,b，若$a*b\equiv 1(\mod p)$，那么则称a和b互为模p意义下的逆元。
    - 注：逆元需要在取模下才有意义，a对模p意义下的逆元即a在模p意义下的倒数
- **逆元的意义：** 如何求解 $\frac{a}{b}\%p$（p是质数）？
  整数的除法运算是向下取整，若b不能整除a，进行除法运算后在对p取模，多得到的并是正确结果，这一点应该很好理解。
  逆元的作用就在如此，他能将除以一个数等价于乘上这个的逆元。
  $\frac{a}{b}\equiv a*\frac{1}{b}(\mod p)$
  设b在模p意义下的逆元为invb,则
  $invb*b\equiv 1(\mod p)\Leftrightarrow invb\equiv\frac{1}{b}(\mod p)$
  $\therefore\frac{a}{b}\equiv a*invb(\mod p)$

7 逆元

- **逆元求解方法**
  - **1.费马小定理**
    - **满足条件：** 模数p必须是素数。
    - **原理：** $a^{p-1} \equiv 1 \pmod p$,p是素数（再此不做证明）
      因此$a * a^{p-2} \equiv 1 \pmod p$
      即a^{p-2}就是a在模p意义下的逆元，$inva = a^{p-2}$,利用快速幂可以求解
  - **2.扩展欧几里德**
    - **满足条件：** $gcd(a, p) = 1$即a与p互质。
    - **原理：** $inva * a \equiv 1 \pmod p$
      $\therefore p \mid (inva * a - 1)$
      $\therefore \exists k \in \mathbb{Z}, inva * a - 1 = pk$
      $\therefore inva * a + k * p = 1, \because gcd(a, p) = 1$
      因此次方程一定有解，利用扩展欧几里德算法可求得inva

```cpp
#define ll long long
//拓展欧几里得求逆元 o(logn)
void exgcd(ll a, ll b, ll &x, ll &y)
{
    if(!b) x = 1, y = 0;
    else {
        exgcd(b, a % b, y, x);
        y -= a / b * x;
    }
}

ll inv(ll a, ll b)
{
    ll x, y;
    exgcd(a, b, x, y);
    return (x + b) % b;
}
//费马小定理求逆元 o(logn)
ll qpow(ll a, ll n, ll p)
{
    ll ans = 1;
    for(; n; n >>= 1, a = a * a % p)
        if(n & 1) ans = ans * a % p;
}

ll inv(ll a, ll b)
{
    return qpow(a, b - 2, b);
}
```

7 逆元

- **递推公式**
  - **满足条件：** 模数p必定是素数 。
  - **原理：**

    $p = p\%a + \lfloor \frac{p}{a} \rfloor * a$

    令 $x = p\%a, y = \lfloor \frac{p}{a} \rfloor$

    $p\%p = (x + y * a)\%p$

    $a^{-1}\%p = (p - x) * y^{-1}\%p$

    因此得 $a^{-1} = (p - \lfloor \frac{p}{a} \rfloor) * (p\%a)^{-1}\%p$

    由此递推式可O(n)求解1~n所有数模p的逆元。
  - **代码实现：**

```
int inv[mod+5];
void getInv(int  p)
{
    inv[1] = 1;
    for(int i = 2; i < p; i ++){
        inv[i] = (p - p / i) * inv[p % i] % p;
    }
}
```