数论讲解——从入门到入梦

主讲人：叶佳伟

目录

01. 同余意义下的运算
02. 数 论 四 大 定 理
03. 素 数 浅 谈
04. 拓展欧几里得算法
05. 中 国 剩 余 定 理
06. B S G S 算 法
07. 卢 卡 斯 定 理
08. 积性函数和卷积运算
09. 莫 比 乌 斯 反 演
10. 数 论 分 块
11. 杜教筛和Min25筛
12. 多项式乘法和卷积

符号介绍

$a|b$：a整除b

(a,b)：a和b的最大公约数

$[a,b]$：a和b的最小公倍数

$\left\lfloor\frac{a}{b}\right\rfloor$:$a$除以$b$向下取整

$\left\lceil\frac{a}{b}\right\rceil$:$a$除以$b$向上取整

$[a=1]$:逻辑判断，当括号内逻辑正确时，值为1，反之值为0

01

同余意义下的运算

什么是同余、同余的加减乘除、求解简单同余式、二次剩余

什么是同余?

设m是正整数，若a和b是整数，且m|(a-b)，则称a和b模意义下同余，记作a≡b(mod m)。

比如2≡5(mod 3)， -1≡3(mod 4) 。

设m是正整数，模m的同余满足下面性质。

(i)自反性，若a是整数，则a≡a(mod m) 。

(ii)对称性，若a和b是整数，且a≡b(mod m)，则 b≡a(mod m) 。

(iii)传递性，若a，b和c是整数，且a≡b(mod m)和b≡c(mod m) ，则a≡c(mod m) 。

模意义下的运算

加法：(a+b)%m

减法：(a-b%m+m)%m

乘法：a*b%m

除法：a*inv(b)%m，其中inv(b)是b模m意义下的逆元

如果m|p，那么a%m==a%p%m成立。

什么是逆元?

$a \times a^{-1} \equiv 1 \pmod p$，则称$a^{-1}$是a在模p意义下的逆元。

当我们进行除法运算时，会有除不尽的情况，而同余运算只适用于整数的运算。所以我们的除法用乘上逆元的方式来代替。那么求解逆元就是一项要做的任务，在后面会介绍。

求解简单同余方程式

已知正整数a,b,m求解x在模m意义下的值

方程1：$x+a\equiv b \pmod m$ 。

方程2：$ax\equiv b \pmod m$ 。其中$\gcd(a,m)=1$。

方程3：$b(x+a)\equiv c \pmod m$ 。其中$\gcd(b,m)=1$。

二次剩余

对于二次同余方程$x^2 \equiv n \pmod{p}$，若[gcd(n,p)=1]，且存在一个x满足该方程，则称n是模p意义下的二次剩余，若无解，则称n为p的二次非剩余。

我们要做的就是求解方程$x^2 \equiv n \pmod{p}$，其中p为奇素数。

做法：(logn)

略（看博客）：https://blog.csdn.net/weixin_43785386/article/details/104086765

02

数论四大定理

威尔逊定理，欧拉定理，孙子定理，费马小定理，费马小定理求逆元

数论四大定理

- 威尔逊定理：p可整除(p-1)!+1是p为质数的充要条件
- 欧拉定理：若gcd(a,n)=1，则$a^{\varphi(n)}\equiv 1(\mathrm{mod}\ n)$，其中φ(n)为欧拉函数。
- 孙子定理：即中国剩余定理，后面会详细介绍。
- 费马小定理：若p为素数，且gcd(a,p)=1，则$a^{p-1}\equiv 1(\mathrm{mod}\ p)$。（其实当p为素数时，φ(n)=p−1）

费马小定理求逆元

什么是逆元?

$a \times a^{-1} \equiv 1 (\text{mod } p)$，则称$a^{-1}$是a在模p意义下的逆元。

那么我们要求解方程$a \times x \equiv 1 (\text{mod } p)$，得出的解x就是a在模p意义下的逆元。

回顾一下费马小定理：若p为素数，且gcd(a,p)=1，则$a^{p-1} \equiv 1 (\text{mod } p)$。

那么当p为素数的时候就有：$a \times a^{p-2} \equiv 1 (\text{mod } p)$。

所以a^{p-2}就是要求的逆元。

注意只适用于p为素数的情况，因为此时费马小定理成立。

```
ll fpow(ll a,ll n,ll mod)//快速幂
{
    ll sum=1,base=a%mod;
    while(n!=0)
    {
        if(n%2)sum=sum*base%mod;
        base=base*base%mod;
        n/=2;
    }
    return sum;
}

ll inv(ll a,ll mod)//逆元
{
    return fpow(a,mod-2,mod);
}
```

素数浅谈

o($\sqrt{n}$)素数判定、o(nlogn)埃氏筛、o(n)线性欧拉筛
o(tlogn)素数测试、o($n^{1/4}$)大数分解、素数密度、
唯一分解定理

o(√n)素数判定

试除法：判断一个数是否为素数。特判1不是素数。枚举2到√n的所有数，尝试去整除n，若存在可以整除n的数，则n不是素数，反之，n为素数。

```
bool isPrime(ll n)
{
    if(n==1)return false;
    for(ll i=2;i*i<=n;i++)if(n%i==0)return false;
    return true;
}
```

o(nlogn)埃氏筛

埃氏筛： o(n)预处理出1到n的素性情况。特判1不是素数。每发现一个素数后，就将它的倍数全部标记为非素数。每次遍历到的第一个未被标记的数，就是素数。从而o(nlogn)预处理出n以内所有数的素性情况。

复杂度证明：调和级数o(n(1/1+1/2+1/3+…+1/n))=o(nlogn)

```
notPrime[1]=1;
for(ll i=1;i<=n;i++)if(!notPrime[i])
{
    for(ll j=2*i;j<=n;j+=i)notPrime[j]=1;
}
```

o(nlogn)埃氏筛

例1：q次询问，每次询问[a,b]中其中一个数是否为素数。
其中(q<1e5，a<1e9，b<1e9，b-a<1e6)

o(n)线性欧拉筛

欧拉筛：o(n)预处理出1到n的素性情况。考虑埃氏筛，每个非素数都会被它的素因子标记一次，从而造成了不必要的多次标记。因此我们可以想办法优化一下。我们让每个非素数只被它最小的素因子标记，这样优化到o(n)。具体做法看代码。

```
int prime[MAXN], vis[MAXN], tot;
void GetPrime(int N) {
    vis[1] = 1;
    for(int i = 2; i <= N; i++) {
        if(!vis[i]) prime[++tot] = i;
        for(int j = 1; j <= tot && i * prime[j] <= N; j++) {
            vis[i * prime[j]] = 1;
            if(!(i % prime[j])) break;
        }
    }
}
```

o(tlogn)素数测试

素数测试（Miller-Rabin算法）：随机算法，通过多次测试判断一个数是否为素数。每次测试复杂度o(logn)，测试t次复杂度为o(tlogn)。检测一次的正确率大概为1/4。

原理：

- 费马小定理：若p为素数，且gcd(a,p)=1，则$a^{p-1}\equiv 1 \pmod{p}$ 。
- 二次探测：若p为素数，则方程$x^2\equiv 1 \pmod{p}$ 的解为x=1或x=p−1。

板子网上有很多，太长就不贴了。

o($n^{1/4}$)大数分解

大数分解（ Pollard-Rho算法）： 将一个数分解质因子。传统算法可以通过o(√n) 将一个数分解质因子。而Pollard-Rho算法算法可以做到o($n^{1/4}$)。不过同样是随机算法，这个比较不稳定，不到万不得已要少用。

素数密度定理

素数密度定理：两个素数不会相距太远。可以理解为两个素数之间的距离为o($\log^2 n$)。大概1e18以内的素数相距都不超过几百。具体是几百，百度找一下吧。

例1：找到最大的不超过n的素数。（n<1e18）

唯一分解定理

所有正整数都可以分解为$p_1^{k1}p_2^{k2}p_3^{k3}\ldots p_n^{kn}$的形式，其中$p_i$为质数。

比如：$180=2^2 3^2 5$、$125=5^3$

这样一来，可以改变很多问题的看法。

比如，两个数的gcd就可以理解为，这两个数的每一位质因子的幂取一个最小值。
两个数的lcm就可以理解为，这两个数的每一位质因子的幂取一个最大值。

如，$\gcd(180,125)=\gcd(2^2 3^2 5, 5^3)= 2^{\min(2,0)}3^{\min(2,0)}5^{\min(1,3)}$
$\mathrm{lcm}(180,125)=\mathrm{lcm}(2^2 3^2 5, 5^3)= 2^{\max(2,0)}3^{\max(2,0)}5^{\max(1,3)}$

当然，求两个数的gcd还是直接求，但是就是会有用到它的例题吧，一下子也找不到了。

拓展欧几里得算法

gcd性质、拓欧求二元一次方程、
拓欧求逆元、同余最短路

gcd性质

求解gcd：辗转相除法。做法略。

性质1：gcd(a,b)=gcd(a-b,b)，gcd(a,b)=gcd(a%b,b)

例1：给定长度为n的序列a_1、a_2……a_n，求最大的x
,使得a_1+x、a_2+x…… a_n+x的gcd最大。
(n<1e6，ai<1e18)

拓欧求二元一次方程

裴(pei)蜀定理：若a和b为整数，二元一次方程ax+by=m有解的充要条件是gcd(a,b)|m。
推论：a,b互质的充要条件是存在整数x,y使ax+by=1。

拓展欧几里得算法：在处理gcd的过程中，顺便求解二元一次方程。

二元一次方程：ax+by=m

做法：

若gcd(a,b)|m不成立，则方程无解。
否则把方程看成ax+by=t×gcd(a,b)的形式。
那么我们只要考虑求解方程ax+by=gcd(a,b)，最后答案乘上t即可。

考虑辗转相除a和b求gcd过程的同时，在每层的a和b，都求出解x和y，然后将这一层的解上传用于求出上一层的解。

比如，在最底层的时候，gcd(a,b)x+0y=gcd(a,b)，此时x=1,y=0是一组特解。

拓欧求二元一次方程

最后求出一组特解x_0和y_0后，通解就是$x=x_0t+kb/gcd(a,b)$，$y=y_0y+ka/gcd(a,b)$，其中k为整数。

```
ll ex_gcd(ll a,ll b,ll& x,ll& y)
{
    if(b==0)
    {
        x=1;y=0;
        return a;
    }
    ll g=ex_gcd(b,a%b,x,y);
    ll tmp=x;
    x=y;
    y=tmp-a/b*y;
    return g;
}
```

拓欧求逆元

用费马小定理求逆元的时候，限制了模数p为素数。
为了处理模数不为素数的情况，我们需要另一个办法来求逆元。
这时候，拓欧又派上用场了。

考虑方程ax≡1(mod p)，
等价于方程ax+py=1。
所以就变成求解二元一次方程的问题了，求出的x就是逆元。
那么就用拓欧来求解。

这时候，我们注意方程有解的条件，
就发现，a在模p意义下逆元存在的充要条件是：gcd(a,p)=1。

```
ll ex_gcd(ll a,ll b,ll& x,ll& y)
{
    if(b==0)
    {
        x=1;y=0;
        return a;
    }
    ll ans=ex_gcd(b,a%b,x,y);
    ll tmp=x;
    x=y;
    y=tmp-a/b*y;
    return ans;
}
ll inv(ll a,ll mod)//存在逆元条件：gcd(a,mod)=1
{
    ll x,y;
    ll g=ex_gcd(a,mod,x,y);
    if(g!=1)return -1;
    return (x%mod+mod)%mod;
}
```

同余最短路

例1（P3403 跳楼机）：有n层楼，小S刚开始在第0层，每次可以上a层，或b层，或c层，或者回到0层，问有最多有多少层楼可以抵达。
(n<1e18,a<1e6,b<1e6,c<1e6)

解:

如果楼层数范围为1e6，那么这题就是一个简单的dp题。

令抵达的楼层f=ax+by+cz，为了缩小楼层数，有f≡ax+by(mod c)。

这样理解的话，当走到楼层i，那么i+kc的楼层都可抵达。

所以我们要求出抵达每个剩余的最小楼层，那么从这个楼层开始，往上走若干个c都是合法的。

我们令dp[i]表示抵达取余后剩余i的最小楼层。

那么得到转移方程dp[(i+a)%c]=min(dp[(i+a)%c]+a,dp[i])和dp[(i+b)%c]=min(dp[(i+b)%c]+a,dp[i])。

那么这个式子就是一个最短路。建边之后，从点0跑一个单源最短路，就可以处理出所有的dp值。

最后遍历每个剩余 i，累加可以加的c的个数，就是答案了。

中国剩余定理

同余方程组、中国剩余定理crt、
扩展中国剩余定理excrt

同余方程组

形如：

$$\begin{cases} x \equiv a_1 \ (mod\ m_1) \\ x \equiv a_2 \ (mod\ m_2) \\ \qquad \dots \dots \\ x \equiv a_n \ (mod\ m_\mathrm{n}) \end{cases}$$

的方程，我们称之为同余方程组。

这章我们要解决的问题就是求解同余方程组。

对于上述同余方程组，我们求解出的最后形式应为

$x \equiv a \ (mod\ lcm(m_1, m_2, \dots, m_n))$

这个a就是我们要求的解。

中国剩余定理

中国剩余定理(crt)：又叫孙子定理。最早在《孙子算经》中提出“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？”。讲的就是同余方程组的问题。

普通版本：方程组的模数两两互质。

那么要解决这个问题，孙子定理采用的办法是构造法。

我们令$M=m_1m_2\ldots m_n$，令$M_\mathrm{i}=\frac{M}{m_i}$。

然后构造出解$x_0=a_1M_1M_1^{-1}+a_2M_2M_2^{-1}+\cdots+a_\mathrm{n}M_nM_n^{-1}$，其中$M_i^{-1}$是$M_\mathrm{i}$在模$m_i$意义下的逆元。

这样，当x_0代入到第i个方程时，除第i项会被保留，其他项都会被约掉，就只剩下第i项，然后第i项代入成立。

特别的，为了满足M_i在模m_i意义下存在逆元，需要模数两两互质。

中国剩余定理

```
ll a[100005],m[100005];
ll crt(ll *a,ll *m,ll n)//长度为0到n-1
{
    ll M=1;
    for(int i=0;i<n;i++)M=M*m[i];
    ll ans=0;
    for(int i=0;i<n;i++)
    {
        ll MM=M/m[i];
        ans=(ans+fmul(fmul(a[i],MM,M),inv(MM,m[i]),M))%M;
    }
    return ans;
}
```

不保证模数为质数，逆元要用拓欧来求，其中fmul为快速乘，具体可以看博客。

扩展中国剩余定理

现在把问题升级，要求模数不用两两互质。然后就有了扩展中国定理(excrt)。

我们假设前$i-1$个方程的通解为$c+kM$，其中M为前$i-1$个方程模数的lcm。

将它代入第i个方程$x \equiv a_i \ (mod\ m_i)$。得到$kM+m_i y=a_i-c$ 。

其中k和y是未知，我们要求出新的k和新的c，用拓展欧几里得求就行了。

扩展中国剩余定理

```
ll a[100005],m[100005];
ll ex_crt(ll *a,ll *m,ll n)//长度为0到n-1
{
    ll M=1,c=0;
    for(int i=0;i<n;i++)
    {
        ll t=(a[i]%m[i]-c%m[i]+m[i])%m[i];
        ll x,y;
        ll g=ex_gcd(M,m[i],x,y);
        if(t%g!=0)return -1;
        ll tM=M;
        M*=m[i]/gcd(M,m[i]);
        x=fmul(t/g,(x%M+M)%M,M);
        c=(c%M+fmul(x,tM,M)+M)%M;
    }
    return (c%M+M)%M;
}
```

应用

比如我们要求解模m的答案，然后m是若干个质数的乘积，且每个质数的次数都是1。假如我们可以求出模数为质数的解，那么我们就可以求出模每个质因子的答案，然后合并即可。

那么如果要求解任意模数的答案，就需要可以求解模质数幂次的答案。

比如前面的二次剩余，我们知道可以o(log)求出模数为奇素数的二次同余方程，有了中国剩余定理定理后，就可以求出模数为素数乘积的答案，但是仅限于每个素数的幂次都为1。

杂技：BM算法可以快速求解线性递推的解，但仅限于模数为素数的情况。有些题目为了防这个，会把模数设为两个素数相乘的模数。然后学会这个杂技之后，我们可以对它的质因子都用BM求一个答案，最后用中国剩余定理合并就好了。当然如果题目把模数设成质因子模数不为1的时候就会出问题了。

06

BSGS算法

指数同余方程、BSGS、exBSGS、原根、高次同余方程

指数同余方程

形如$a^x \equiv b \ (mod\ p)$的方程，我们称为指数同余方程。

BSGS算法是用于处理$\gcd(a,p) = 1$时，方程的解。

而当$\gcd(a,p) \neq 1$时，就需要用扩展BSGS算法了。

BSGS

BSGS：求出$\gcd(a,p)=1$的指数同余方程$a^x \equiv b \ (mod\ p)$的解。

做法：（折半）

先说最暴力的做法，我们枚举从0到$p-1$枚举x，然后判断方程是否成立。这样复杂度是$o(p)$的

那么折半的话，我们设解 $x_0 = kn + i$，那么方程变成 $a^{kn+i} \equiv b \ (mod\ p)$，$n$ 是我们自己定的一个数。

进而转化成$a^{kn} \equiv b(a^i)^{-1} \ (mod\ p)$。

我们枚举从0到$n-1$所有的i，将右边的结果存入哈希表中。（也可以用map，会多个log）

然后枚举k，枚举范围是$kn < p$，算出左边的值，去表中查值即可。

这样复杂度为$o(\sqrt{p})$。

BSGS

```
ll BSGS(ll a,ll b,ll p)
{
    b%=p;
    if(b==1||p==1)return 0;
    ll n=sqrt(p);
    static unordered_map<ll,ll>Bmp;
    Bmp.clear();
    ll inva=inv(fpow(a,n-1,p),p)*b%p;
    for(ll i=n-1;i>=0;i--)
    {
        Bmp[inva]=i;inva=inva*a%p;
    }
    ll ta=1,powa=fpow(a,n,p);
    for(ll k=0;k<=p;k+=n)
    {
        if(Bmp.count(ta))return k+Bmp[ta];
        ta=ta*powa%p;
    }
    return -1;
}
```

exBSGS

exBSGS：求出$\gcd(a,p)$没有要求的指数同余方程$a^x \equiv b \ (mod\ p)$的解。

做法：（略）

看博客：https://blog.csdn.net/weixin_43785386/article/details/104108230

exBSGS

```
ll exBSGS(ll a,ll b,ll p)
{
    b%=p;
    if(a==0&&b==0)return 1;
    else if(a==0&&b!=0)return -1;
    if(b==1||p==1)return 0;
    ll d=gcd(a,p);
    if(b%d!=0)return -1;
    p=p/d;
    b=b/d*inv(a/d,p)%p;
    if(d!=1)
    {
        ll ans=exBSGS(a,b,p);
        if(ans==-1)return -1;
        return ans+1;
    }
    ll ans=BSGS(a,b,p);
    if(ans==-1)return -1;
    return ans+1;
}
```

原根

原根：如果g^1 ，$g^2 \dots g^{\varphi(p)}$是模p意义下的既约剩余系。则我们称g为p的一个原根。

定理一：当且仅当x是$\varphi(n)$的倍数时，使得 $a^x \equiv 1(mod\ n)$ 成立，此时称a为n的原根。

定理二：如果一个数n有原根，那么他有$\varphi(\varphi(n))$个不同余的原根。

定理三： 2和4，和奇素数的正整数幂次p^k ，以及2乘上奇素数的正整数幂次$2p^k$都是有原根的。

当p为素数时，g的幂次组成了模p意义下的完全剩余系。

有了这个性质之后，当p为素数时，我们就可以在模p意义下把x用g^x代替掉，这样我们求出新的x的解之后，那么原来的解就是g^x。这对于我们后面求解高次同余方程很有帮助。

高次同余方程

形如$x^a \equiv b\ (mod\ p)$的方程，我们称为高次同余方程。

只考虑求解p为素数的情况，因为此时p的原根g的幂次组成了完全剩余系。

于是我们可以把x换成g^x，这样只要求出新的x，然后g^x就是方程的解了。

现在方程就变成了求解$g^{ax} \equiv b\ (mod\ p)$

用前面的BSGS求出方程的解为$ax \equiv c\ (mod\ p-1)$

就变成$ax+(p-1)y=c$，用扩展欧几里得求解就好了。

07

卢卡斯定理

组合数取模、卢卡斯定理、扩展卢卡斯定理

求解组合数取模

法一：暴力求解

法二：利用杨辉三角预处理出二维数组。

求解组合数取模

法三：利用$C_n^k = \frac{n!}{k!(n-k!)}$。

我们要求解$C_n^k \ (mod\ p)$。

可以预处理出1到n的阶乘，以及1到n阶乘的逆元，然后直接套公式。

```
ll jie[1000005],rjie[1000005];
void init_jie(ll n,ll mod)
{
    jie[0]=1;
    for(ll i=1;i<=n;i++)jie[i]=jie[i-1]*i%mod;
    for(ll i=0;i<=n;i++)rjie[i]=inv(jie[i],mod);
}
ll C(ll n,ll k,ll mod)
{
    return jie[n]*rjie[k]%mod*rjie[n-k]%mod;
}
```

但是，这样还是会涉及到逆元是否存在的问题。
那么当$\gcd(n,p) \neq 1$时，就会有逆元不存在的问题了。

对于p是质数，且n比p小的时候还是可以简单这样写写。

但是对于逆元可能不存在的情况，我们就需要用到卢卡斯定理了

卢卡斯定理

卢卡斯(Lucas)定理：$C_n^k \equiv C_{n/p}^{k/p} \times C_{n\%p}^{k\%p}\ (mod\ p)$ ，其中p为素数。

当p为素数时，只有当$n \geq p$时会出现$\gcd(n,p) \neq 1$的情况，即n是p的倍数的情况。

那么这种情况，有了卢卡斯定理之后就可以最坏情况下$o(logn)$求解了。

```
ll Lucas(ll n,ll k,ll mod)//返回n取k对mod取模
{
    if(n<k)return 0;
    if(n>=mod)return
Lucas(n/mod,k/mod,mod)*Lucas(n%mod,k%mod,mod)%mod;
    else return jie[n]*rjie[n-k]%mod*rjie[k]%mod;
}
```

扩展卢卡斯定理

当p不为素数时，用前面的方法就不能求了。

这时候要用扩展卢卡斯定理(exLucas)，复杂度为$o(p)$。

做法：略

看博客：https://blog.csdn.net/weixin_43785386/article/details/104094632

08

积性函数和卷积运算

积性函数、常见积性函数、欧拉函数、
莫比乌斯函数、迪利克雷卷积

积性函数

算术函数：对所有正整数定义的函数，也就是定义域为正整数的函数。

积性函数：一种特殊的算术函数，满足$f(1)=1$，且对于任意互质的两个数p和q，有$f(pq)=f(p)f(q)$。

注：所有积性函数都可以用线性筛处理出来，建立在$o(n)$素数筛的基础上。
具体做法看博客：https://blog.csdn.net/weixin_43785386/article/details/104489219

后面介绍一些常见的积性函数。

常见积性函数

单位函数e：满足$e(n)=\begin{cases}1, n=1\\ 0,其他\end{cases}$，或者说$e(n)=[n=1]$。

常数函数1：满足$1(n)=1$。

因子和函数σ：满足$\sigma(n)=\sum_{d|n} d$ 。

因子个数函数τ：满足$\tau(n)=\sum_{d|n} 1$ 。

欧拉函数

欧拉函数φ：$\varphi(n)$的值为小于n且与n互质的正整数个数。

如：$\varphi(6) = 2$

性质1：当p为素数时，有$\varphi(p) = p - 1$。

性质2：$\varphi\left(p^k\right) = p^k - p^{k-1}$，其中$p$是素数，$k$是正整数。

性质3：有通项公式$\varphi(x) = x\prod_{i=1}^{n}(1 - \frac{1}{p_i})$，其中$p_1, p_2, \ldots, p_\mathrm{n}$是$x$的所有质因子。这个式子可以用于$o(\sqrt{\mathrm{x}})$求解欧拉函数值。

性质4：若n为奇数，有$\varphi(2n) = \varphi(n)$ 。

性质5：若$i\%p == 0$，有$\varphi(ip) = \ p \times \varphi(i)$;

性质6：若x与p互质，则$p - x$也与p互质，因此小于p且与p互质的数之和为$\frac{\varphi(x)x}{2}$。

欧拉函数求逆元

前面介绍了费马小定理求模数为素数的逆元，以及拓欧求逆元。

现在再介绍一个欧拉函数求逆元，其实和费马小定理求逆元差不多，不过没有了模数为素数的限制。

欧拉定理：若$\gcd(a,n)=1$，则$a^{\varphi(n)} \equiv 1(mod\ n)$，其中$\varphi(n)$为欧拉函数。

然后我们就得到了 $a^{-1} \equiv a^{\varphi(n)-1}(mod\ n)$。

当然根据欧拉定理的限制条件，我们知道逆元存在条件是$\gcd(a,n)=1$。

欧拉降幂

扩展欧拉定理：$a^b \equiv \begin{cases} a^b & , b < \varphi(m) \\ a^{b \bmod \varphi(m) + \varphi(m)}, & b \geq \varphi(m) \end{cases} \pmod m$

当指数巨大时，就可以用扩展欧拉定理进行降幂。

例1：计算A^B mod C，其中1<=A,C<=1000000000,1<=B<=10^1000000

莫比乌斯函数

莫比乌斯函数μ：$\mu(n)=\begin{cases}1 & ,x=1\\(-1)^r,x=p_1p_2\dots p_r\text{，即}x\text{的素因子次数均为}1\text{且素因子个数为}r\\0 & \text{，其他情况，即存在平方因子}\end{cases}$

这个函数的作用主要是莫比乌斯反演，至于为什么长这个样子，后面会讲。

迪利克雷卷积

迪利克雷卷积运算：是一种算术函数之间的运算法则，函数与函数之间的运算，结果也是一个函数。

迪利克雷卷积公式：$h(n) = (f * g)(n) = \sum_{d|n} f(d)g(\frac{n}{d})$

也就是两个函数下标相乘为n的函数值乘起来，求和得到的结果。

可以把卷积理解成两个函数进行的运算，运算后得到一个新的函数。

像这里把下标乘积作为新的下标的卷积就是迪利克雷卷积。

也有把下标和作为新下标的卷积，卷积形式为$h(n) = (f * g)(n) = \sum_{i=0}^{n} f(i)g(n-i)$，这个用FFT或NTT求解。

又或者用下标位运算结果作为新的下标，卷积形式为$C_k = \sum_{i|j=k} A_iB_j$， $C_k = \sum_{i\&j=k} A_iB_j$， $C_k = \sum_{i\wedge j=k} A_iB_j$，这类题目就是FWT了。

迪利克雷卷积

知道这个卷积运算后，把这些常见的函数都卷一卷，会有一些神奇的东西。

$\tau = 1 * 1$

$\sigma = 1 * id$

$1 * \mu = e$

$\varphi * 1 = id$

$\varphi = id * \mu$

其中$id(n) = n$，是算术函数，但不是积性函数。

可以通过这些式子进行一些变换。

同时我们发现$1 * \mu = e$，即莫比乌斯函数是1函数的逆函数，这个性质对莫比乌斯反演有很大的帮助，又或者说，这个函数是根据这个式子构造出来的。

09

莫比乌斯反演

两个反演公式

莫比乌斯反演

反演公式一：已知$F(n)=\sum_{d|n}f(d)$，则有$f(n)=\sum_{d|n}\mu(d)F\left(\frac{n}{d}\right)$，(或者$\sum_{d|n}\mu\left(\frac{n}{d}\right)F(d)$)

反演公式二：已知$F(n)=\sum_{n|d}f(d)$，则有$f(n)=\sum_{n|d}\mu\left(\frac{d}{n}\right)F(d)$

反演公式一的证明：

$F=f*1$

两边同乘μ得

$F*\mu=f$

证毕

那么当$f(x)$不好求，而$F(x)$好求的时候，就可以用莫比乌斯反演将式子化解。

莫比乌斯反演

一个常见的f和F的使用。

$f(t)$表示当$t = g$时的答案，
$F(t)$表示当$t|g$时的答案。
那么此时满足$F(n) = \sum_{n|t} f(t)$

例1：给出两个区间$[a, b]$，$[c, d]$。问有多少对互质的x和y，满足$a \le x \le b$，$c \le y \le d$。（$a, b, c, d \le 1e7$）

解：

我们令$f(t)$表示$t = \gcd(x, y)$时的答案。

我们令$F(t)$表示$t|\gcd(x, y)$时的答案。

此时有$F(n) = \sum_{n|t} f(t)$。而我们要求的是$f(1)$。我们发现f很难求，

而F很好 求，即$F(n) = (\left\lfloor\frac{b}{t}\right\rfloor - \left\lfloor\frac{a}{t}\right\rfloor + 1)(\left\lfloor\frac{d}{t}\right\rfloor - \left\lfloor\frac{c}{t}\right\rfloor + 1)$。

反演后得到$f(t) = \sum_{t|n} \mu(\frac{n}{t})F(n)$

则$f(1) = \sum_{1|n} \mu(n)F(n)$，然后枚举就好了。

10

数论分块

整除分块

数论分块

数论分块：也叫整除分块，形如$\sum_{i=1}^{n} f(\left\lfloor\frac{n}{i}\right\rfloor)$的式子都可以用数论分块将复杂度优化到o($\sqrt{n}$)。

原理：确定某个数n，对于任意的整数i，$\left\lfloor\frac{n}{i}\right\rfloor$的总共的值不超过o($\sqrt{n}$)个，且相同值的$i$是连续的。所以我们可以对于每一种取值，把区间长度找出来，然后算到答案贡献中。

例1：求$\sum_{i=1}^{n}\left\lfloor\frac{n}{i}\right\rfloor$。$(n \leq 1e12)$

```
ll ans=0;
for(ll l=1,r;l<=n;l=r+1)
{
        r=n/(n/l);
        ans+=(r-l+1)*(n/l);
}
```

杜教筛和Min25筛

积性函数前缀和

杜教筛

积性函数前缀和：求解积性函数时，我们可以通过线性筛$o(n)$预处理，或者$o(\sqrt{n})$求出单点的值。然而在求解积性函数前缀和的时候，却神奇的有了更快的算法 $o(n^{\frac{3}{4}})$， 这就是杜教筛，给了出题人更多的可能性。

做法：

比如要求解积性函数的前缀和$S(n)=\sum_{i=1}^{n} f(i)$。

我们用另一个积性函数和它卷积 $\sum_{i=1}^{n}(f*g)(i)$

$$=\sum_{i=1}^{n}\sum_{d|n} f(d)g(\frac{n}{d})$$

$$=\sum_{d=1}^{n} g(d)\sum_{i=1}^{\left\lfloor\frac{n}{d}\right\rfloor} f(i)$$

$$=\sum_{d=1}^{n} g(d)\,S(\left\lfloor\frac{n}{d}\right\rfloor)$$

杜教筛

前面得到了这个式子 $\sum_{i=1}^{n}(f*g)(i)=\sum_{d=1}^{n}g(d)\,S(\lfloor\frac{n}{d}\rfloor)$

由于g是积性函数，有$g(1)=1$。

所以我们要求的就是

$$\sum_{i=1}^{n}f(i)g(1)=\sum_{d=1}^{n}g(d)S\left(\lfloor\frac{n}{d}\rfloor\right)-\sum_{d=2}^{n}g(d)S\left(\lfloor\frac{n}{d}\rfloor\right)$$
$$=\sum_{i=1}^{n}(f*g)(i)-\sum_{d=2}^{n}g(d)S\left(\lfloor\frac{n}{d}\rfloor\right)$$

于是，我们只要找到一个积性函数g，使得g的前缀和容易求，且$f*g$的前缀和也容易求，就可以递归求f的前缀和了。

复杂度是$o(n^{\frac{3}{4}})$，做的时候我们可以先预处理出，前$1e6$位的前缀和，然后函数里$o(1)$查这些位，就可以优化到$o(n^{\frac{2}{3}})$。

杜教筛求欧拉函数前缀和

板子略。

杜教筛求莫比乌斯函数前缀和

板子略。

12

多项式乘法和卷积

FFT、NTT、FWT、生成函数

快速傅里叶变换FFT

快速傅里叶变换(FFT)：用于在o(nlogn)复杂度内求解多项式乘积。

实现原理：通常多项式都是用系数表示法，但是两个系数表示法的多项式相乘，朴素做法中是$o(n^2)$的。而两个点值表示法的多项式相乘是o(n)的，而FFT做的就是o(logn)将多项式在系数表示法和点值表示法中转换。

做法：o(nlogn)转化为点值表示法，
o(n)将两个多项式相乘，
o(nlogn)转化为系数表示法。

转化的时候利用了复数i的变换。

快速数论变换NTT

快速数论变换(NTT)：同样用于求解多项式乘积，多了一个取模问题，和FFT不同的是，利用了原根的变换。但并不是有原根就可以NTT，这个对模数有一定要求。常用的模数为998244353，原根为3。

生成函数

(讲组合数学的时候ztc应该也会讲)

生成函数：比如数列$a_0, a_1, \dots, a_n$,那么它的生成函数就是$a_0 + a_1x + \cdots a_nx^n$。

一些简单运用：

很多个木棍，长度为i的有a_i根。

我们写出生成函数$a_0 + a_1x + \cdots a_nx^n$

又有很多个木棍，长度为i的有b_i根。

我们从第一堆木棍和第二堆木棍中，各取出一根木棍拼接成新的长度。

我们现在要求拼接后没种长度木棍的方案数c_i。

那么c_i的生成函数就是$(a_0 + a_1x + \cdots a_nx^n)(b_0 + bx + \cdots b_nx^n)$

这时候就需要用到多项式乘法。

卷积运算

或者我们换种方式考虑刚才的问题，

$a(i)$为第一堆长度为i的木棍数， $b(i)$为第二堆长度为i的木棍数。

我们要求的$c(n) = \sum_{i+j=n} a(i) * b(j)$

这是我们前面也提到过的卷积形式。也就是用多项式乘法所解决的问题。

卷积运算

刚才讲了FFT和NTT处理加法卷积，现在说几个位运算的卷积形式。

$$c(n) = \sum_{i|j=n} a(i) * b(j)$$

$$c(n) = \sum_{i\&j=n} a(i) * b(j)$$

$$c(n) = \sum_{i \wedge j=n} a(i) * b(j)$$

这三种卷积形式，就是用快速沃尔什变换FWT解决的了。
具体还请自行学习。

下课！