# 同余定理,快速幂，费马小定理

## 同余定理:

Pow(a,n)代表a的n次方，inv(a,mod)代表a对mod的逆元

(a+b)%mod=(a%mod+b%mod)%mod

A=20，b=30,mod=7

(a+b)%mod=50%7=1 (a+b)%mod=(20%7+30%7)%7=(6+2)%7=1

(a\*b)%mod=(a%mod\*b%mod)%mod

A=20，b=30,mod=7

(a\*b)%mod=600%7=5 (a\*b)%mod=(20%7\*30%7)%7=(6\*2)%7=5

(a-b)%mod=(a%mod-b%mod+mod)%mod

A=30,b=20,mod=7

(a-b)%mod=10%7=3 (a-b)%mod=(30%7-20%7+mod)%7=

(2-6+7)%7=3

pow(a,n)%mod=pow(a%mod,n) %mod;;

* (a/b)%mod=a%mod\*inv(b,mod)%mod;

## 快速幂:

pow(a,n)代表a的n次方

pow(a,n)%mod= pow(a%mod,n)%mod;

慢速的算法求pow(a,n)：

ans=1;

for(i=0;i<n;i++){

ans=ans\*(a%mod)%mod;

}

时间复杂度N

快速幂思路：

任何数能表示成pow(2,k)相加的形式，k是某个非负数

比如11=pow(2,3)+ pow(2,1)+ pow(2,0)=8+2+1

11的2进制: 1011

pow(a,n)求法是分解n,把n变成2进制，设2进制数的第i位是bin[i]

n=

只需要求出pow(a,)即可

11的2进制: 1011

pow(a,11)= pow(a,pow(2,3))\* pow(a, pow(2,1))\* pow(a, pow(2,0))

= pow(a,8))\* pow(a,2)\* pow(a,1)

怎么分解n:

while(n>0){

printf(“%d”,n%2);

n/=2;

}

快速幂模板代码：

ll quickpow(ll a,ll n,ll mod){

ll ans=1,s=a%mod;

if(mod==1)

return 0;

while(n){

if(n%2==1){

ans=ans\*s%mod;

}

s=s\*s%mod;

n/=2;

}

return ans;

}

时间复杂度log2(N)

求逆元:

* (a/b)%mod=a%mod\*inv(b,mod)%mod;

inv(b,mod)代表逆元

逆元用途之一是来计算(a/b)%mod=a\*(b关于mod的逆元)%mod

定义：a\*x%mod==1,则称最小正整数的x是a模mod的逆元

逆元不只有一个，符合上述条件的都叫逆元

逆元本身就是消除一个运算的操作，通常指的是乘法逆元

# 费马小定理：

对于质数p,和任意正整数a

pow(a,p-1)%p==1

因此很容易知道 a\* a^(p-2) mod p==1

a\*x%p==1 x是逆元

a\* a^(p-2) mod p==1

a^(p-2)是逆元

P是素数时候，用快速幂算逆元a^p-2 就是a模p逆元

求逆元inv(a,p)等价于求pow(a,p-2)%p

* (a/b)%mod=a%mod\*inv(b,mod)%mod;

对于mod是素数的情况，

* (a/b)%mod=a%mod\*inv(b,mod)%mod=

a%mod\*pow(n,mod-2)%mod=

a%mod\*quickmi (n,mod-2,mod)%mod