数论

素性检测和反素数

素数计数函数：近似为$\pi(x) \sim x/lnx$

二次探测定理：p为素数，$x^2\%p=1$的解为$x_1=k*p+1, x_2=k*p-1$

Pollard-Rho算法

可算出其非平凡因子，也就是1<a<n的a

生日悖论：一个班k个人，找一个人生日为4月2日的概率很低；但是，找到有人生日重复的概率很高，$p=1-\frac{364}{365}\frac{363}{365}\cdots\frac{365-k}{365}$，当$k\geq 23, p\geq 50\%; k\geq 60, p\geq 99\%$

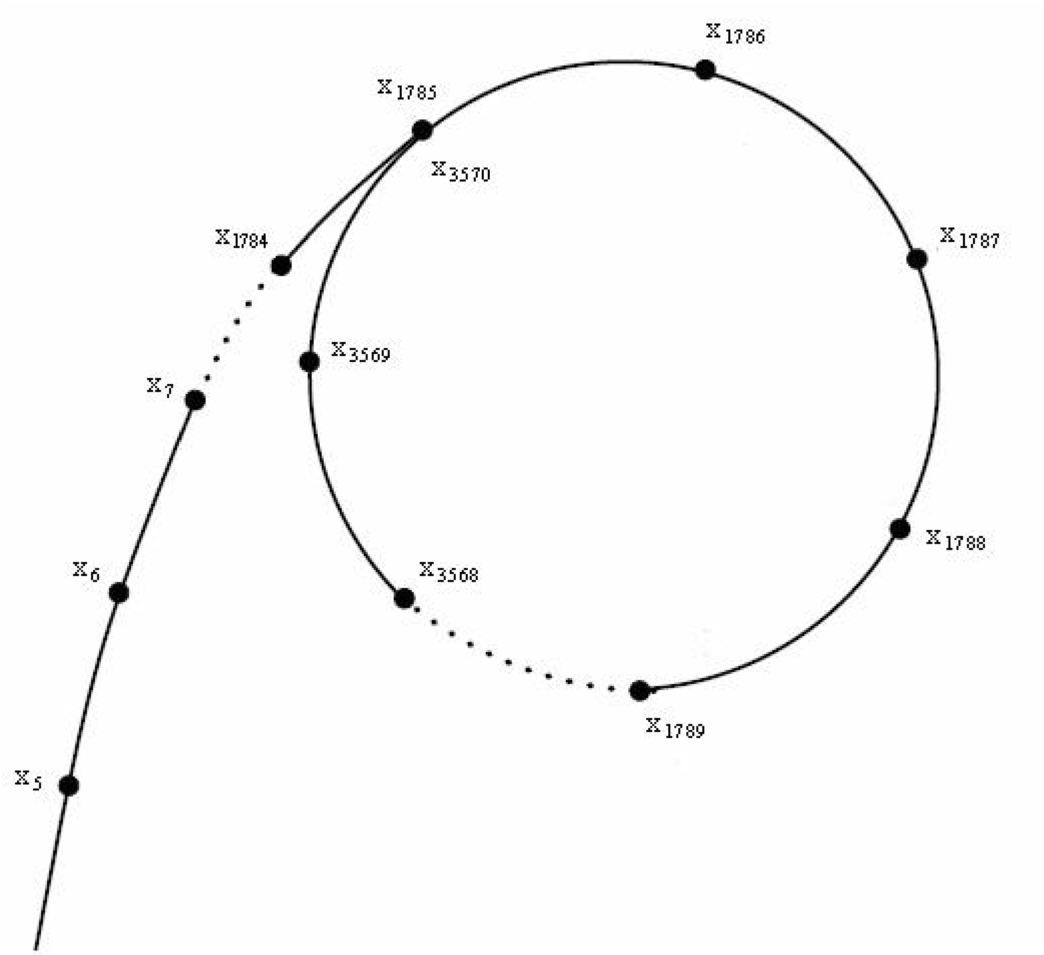
生日悖论的实质是:由于采用"组合随机采样"的方法,满足答案的组合比单个个体要多一些.

我们不放选取一组数$x_1,x_2,x_3,\ldots,x_n$,若有$gcd(|x_i-x_j|,N)>1$,则称$gcd(|x_i-x_j|,N)$是N的一个因子.早期的论文中有证明,需要选取的数的个数大约是$O(N^{\frac{1}{4}})$.但是,我们还需要解决储存方面的问题.如果$N=10^{18}$,那么我们也需要取10^4个数.如果还要$O(n^2)$来两两比较,时间复杂度又会弹回去.

我们不妨考虑构造一个伪随机数序列,然后取相邻的两项来求gcd.为了生成一串优秀的随机数,Pollard设计了这样一个函数: $f(x)=(x^2+c)\%N$ 其中c是一个随机的常数.

我们随便取一个x_1,令$x_2=f(x_1), x_3=f(x_2),\ldots\ldots,x_i=f(x_{i-1})$.在一定的范围内,这个数列是基本随机的,可以取相邻两项作差求gcd.

这个数列是混沌的，之所以叫伪随机，是因为这个数列里面会包含有"死循环"，如$c=7,N=20,x_1=1$,生成的数列像希腊字母ρ,所以叫Pollard-Rho算法。



基于Floyd算法优化的Pollard Rho

为了判断环的存在,可以用一个简单的Floyd判圈算法,也就是"龟兔赛跑". 假设乌龟为t,兔子为r,初始时$t=r=1$.假设兔子的速度是乌龟的一倍. 过了时间i后,$t=i, r=2i$.此时两者得到的数列值$x_t=x_i, x_r=x_{2i}$. 假设环的长度为c,在环内恒有:$x_i=x_{i+c}$. 如果龟兔"相遇",此时有:$x_r=x_t$,也就是$x_i=x_{2i}=x_{i+kc}$.此时两者路径之差正好是环长度的整数倍。

这样以来,我们得到了一套基于Floyd判圈算法的Pollard Rho 算法.

```
ll gcd(ll x,ll y){
    return x%y?gcd(y,x%y):y;
}
ll f(ll x,ll c,ll n){
    return (x*x+c)%n;
}
ll Pollard_Rho(ll N){
    srand(time(0));
    ll c=rand()%(N-1)+1;
    ll t=f(0,c,N);
    ll r=f(f(0,c,N),c,N);
    while(t!=r){
        ll d=gcd(abs(t-r),N);
        if (d>1)return d;
        t=f(t,c,N);
        r=f(f(r,c,N),c,N);
    }
    return N;//没有找到,重新调整参数c
}
```

使用gcd求解的时间复杂度为$O(logn)$,频繁使用会使算法运行很慢

倍增优化

我们每过一段时间将这些差值进行gcd运算，设$s=\prod|x_0-x_j|\%n$，如果某一时刻得到$s=0$那么表示分解失败，退出并返回n本身。每隔2^k-1个数，计算是否满足$1<gcd(s,n)<n$。此处取$k=7$，可以根据实际情况进行调节。

```
ll gcd(ll x,ll y){
    return x%y?gcd(y,x%y):y;
}
ll f(ll x,ll c,ll n){
    return ((__int128)x*x+c)%n;
}
ll Pollard_Rho(ll x){
    ll s=0,t=0,c=rand()%(x-1)+1;
    int step=0,goal=1;
    ll val=1;
    for (goal=1;;goal<<=1,s=t,val=1) {
        for (step=1;step<=goal;++step) {
            t=f(t,c,x);
            val=(__int128)val*abs(t-s)%x;
            if((step % 127)==0){
                ll d=gcd(val,x);
                if(d>1)return d;
            }
        }
        ll d=gcd(val,x);
        if (d>1)return d;
    }
}
```

求最大质因数

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
ll qpow(ll a,ll n,ll p){
    ll ans=1;
    while(n>0){
        if((n&1)>0)ans=(__int128)ans*a%p;
        a=(__int128)a*a%p;
        n>>=1;
    }
    return ans;
}
bool mr(ll x,ll b){
    ll k=x-1;
    while(k){
        ll cur=qpow(b,k,x);
        if(cur!=1&&cur!=x-1)return false;
        if((k&1)==1||cur==x-1)return true;
        k>>=1;
    }
    return true;
}
bool prime(ll x){
```

```
    if(x==46856248255981ll||x<2)return false;
    if(x==2||x==3||x==7||x==61||x==24251)return true;
    return mr(x,2)&&mr(x,61);
}
ll gcd(ll x,ll y){
    return x%y?gcd(y,x%y):y;
}
ll f(ll x,ll c,ll n){
    return ((__int128)x*x+c)%n;
}
ll Pollard_Rho(ll x){
    ll s=0,t=0,c=rand()%(x-1)+1;
    int step=0,goal=1;
    ll val=1;
    for (goal=1;;goal<<=1,s=t,val=1) {
        for (step=1;step<=goal;++step) {
            t=f(t,c,x);
            val=(__int128)val*abs(t-s)%x;
            if((step % 127)==0){
                ll d=gcd(val,x);
                if(d>1)return d;
            }
        }
        ll d=gcd(val,x);
        if (d>1)return d;
    }
}
ll max_factor;
void fac(ll x){
    if(x<=max_factor||x<2)return;
    if(prime(x)){
        max_factor=max_factor>x?max_factor:x;
        return;
    }
    ll p=x;
    while(p>=x)p=Pollard_Rho(x);
    while((x%p)==0)x/=p;
    fac(x),fac(p);
}
signed main(){
    int T;
    srand(time(0));
    cin>>T;
    while (T--){
        ll n;
        max_factor=0;
        scanf("%lld",&n);
        fac(n);
        if (max_factor==n)puts("Prime");
        else printf("%lld\n",max_factor);
    }
    return 0;
}
```

Miller-Rabin 素性测试

```
ll qpow(ll a,ll n,ll p){
    ll ans=1;
    while(n>0){
        if((n&1)>0)ans=(__int128)ans*a%p;
        a=(__int128)a*a%p;
        n>>=1;
    }
    return ans;
}
bool mr(ll x,ll b){
    ll k=x-1;
    while(k){
        ll cur=qpow(b,k,x);
        if(cur!=1&&cur!=x-1)return false;
        if((k&1)==1||cur==x-1)return true;
        k>>=1;
    }
    return true;
}
bool prime(ll x){
    if(x==46856248255981ll||x<2)return false;
    if(x==2||x==3||x==7||x==61||x==24251)return true;
    return mr(x,2)&&mr(x,61);
}
```

反素数：任何小于n的正数的约数个数都小于n的约数个数。素数就是因子只有两个的数，那么反素数，就是因子最多的数。

给定因子数的最小数：

```
#include<bits/stdc++.h>
using namespace std;
#define ULL unsigned long long
#define INF ~0ULL
ULL p[16] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53};
ULL ans,n;
void dfs(ULL depth,ULL temp,ULL num,ULL up) {
    if (num>n)return;
    if (num==n&&ans>temp){
        ans=temp;
        return;
    }
    for (int i=1;i<=up;i++){
        if (temp>ans)break;
        dfs(depth+1,temp=temp*p[depth],num*(i+1),i);
    }
}
signed main(){
    scanf("%llu",&n);
    ans=INF;
    dfs(0,1,1,64);
    printf("%llu\n",ans);
    return 0;
}
```

gcd与exgcd与类欧几里得

裴蜀定理:不全为0的整数a,b,存在ax+by=gcd(a,b)

```
int gcd(int x,int y){
    return y==0?x:gcd(y,x%y);
}

int exgcd(int a,int b,int &x,int &y){
    if(b==0){
        x=1;y=0;return a;
    }
    int r=exgcd(b,a%b,x,y);
    int temp=y;y=x-(a/b)*y;x=temp;
    return r;
}

#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define int long long
int maxd,inv2,inv6;
int n,a,b,c;
//f(a,b,c,n)=sum(i,0,n)[(a*i+b)/c] 斜线下方的所有点
//g(a,b,c,n)=sum(i,0,n)i*[(a*i+b)/c]
//h(a,b,c,n)=sum(i,0,n)[(a*i+b)/c]的平方
struct node{
    int f,g,h;
};
node solve(ll a,ll b,ll c,ll n){
    node ans;
    if (!a){
        ans.f=(n+1)*(b/c)%maxd;
        ans.g=(n+1)*n%maxd*inv2%maxd*(b/c)%maxd;
        ans.h=(n+1)*(b/c)%maxd*(b/c)%maxd;
        return ans;
    }
    if ((a>=c) || (b>=c)){
        node tmp=solve(a%c,b%c,c,n);
        ans.f=(tmp.f+n*(n+1)%maxd*inv2%maxd*(a/c)%maxd+(n+1)*(b/c)%maxd)%maxd;
ans.g=(tmp.g+n*(n+1)%maxd*(n*2+1)%maxd*inv6%maxd*(a/c)%maxd+n*
(n+1)%maxd*inv2%maxd*(b/c)%maxd)%maxd;
ans.h=(tmp.h+n*(n+1)%maxd*(n*2+1)%maxd*inv6%maxd*(a/c)%maxd*(a/c)%maxd+
(n+1)%maxd*(b/c)%maxd*(b/c)%maxd+tmp.f*(b/c)%maxd*2%maxd+tmp.g*(a/c)%maxd*2%maxd+
(a/c)*(b/c)%maxd*n%maxd*(n+1)%maxd)%maxd;
        return ans;
    }
    ll m=(a*n+b)/c;
    node tmp=solve(c,c-b-1,a,m-1);m%=maxd;
    ans.f=(n*m%maxd+maxd-tmp.f)%maxd;
    ans.g=(m*n%maxd*(n+1)%maxd+maxd-tmp.f+maxd-tmp.h)%maxd*inv2%maxd;
    ans.h=(n*m%maxd*(m+1)%maxd-tmp.g*2%maxd-tmp.f*2%maxd-ans.f)%maxd;
    ans.f=(ans.f+maxd)%maxd;ans.g=(ans.g+maxd)%maxd;ans.h=(ans.h+maxd)%maxd;
    return ans;
}
int qpow(int x,int y){
    int ans=1;
```

```
    while (y){
        if (y&1)ans=ans*x%maxd;
        y>>=1;x=x*x%maxd;
    }return ans;
}
signed main(){
    maxd=998244353;
    int T;
    inv2=qpow(2,maxd-2);
    inv6=qpow(6,maxd-2);
    cin>>T;
    while (T--){
        cin>>n>>a>>b>>c;
        node ans=solve(a,b,c,n);
        cout<<ans.f<<" "<<ans.h<<" "<<ans.g<<endl;
    }
    return 0;
}
```

3.逆元

//a,p互质，两者才有逆元，所以0没有逆元

```
qpow(n,mod-2)
```

```
void ex_gcd(int a,int b,int &x,int &y){
    if (!b){x=1,y=0;return;}
    ex_gcd(b,a%b,x,y);
    int t=x;x=y;y=t-(a/b)*y;
}
int inv(int a,int p) {
    int inv_a,y;
    ex_gcd(a,p,inv_a,y);
    return (inv_a%p+p)%p;//因为可能为负
}
```

```
int inv[N];
void get_inv(int n,int p){
    inv[0]=inv[1]=1;
    for (int i=2;i<=n;i++){
        inv[i]=inv[p%i]*(p-p/i)%p;
    }
}
```

4.线性筛

```
void get_prim(int n){
    int num=0;
    for (int i=2;i<=n;i++){
        if (vis[i]==0)prim[++num]=i;
        for (int j=1;j<=num&&1ll*prim[j]*i<=n;j++){//long long
            vis[i*prim[j]]=1;
            if(i%prim[j]==0)break;
        }
    }
}
```

```
void get_phi(int n){
    int num=0;phi[1]=1;
    for (int i=2;i<=n;i++){
        if (vis[i]==0){prim[++num]=i;phi[i]=i-1;}
        for (int j=1;j<=num&&prim[j]*i<=n;j++){
            vis[i*prim[j]]=1;
            if (i%prim[j]==0){
                phi[i*prim[j]]=phi[i]*prim[j];
                break;
            }
            else phi[i*prim[j]]=phi[i]*phi[prim[j]];
        }
    }
}
```

```
void get_mu(int n){
    int cnt=0;mu[1]=1;
    for(int i=2;i<=n;i++){
        if(!vis[i]){prim[++cnt]=i;mu[i]=-1;}
        for(int j=1;j<=cnt&&prim[j]*i<=n;j++){
            vis[prim[j]*i]=1;
            if(i%prim[j]==0)break;
            else mu[i*prim[j]]=-mu[i];
        }
    }
}
```

5.欧拉函数

$\phi(1)=1,\phi(n)=\sum_{i=1}^{n-1}[gcd(n,i)=1]$

$\phi(p^k)=p^{k-1}(p-1)$，因为与p^k不互质的只有p的倍数

欧拉函数性质: $\frac{phi[n]*n}{2}$为与n互质的数的和

欧拉定理 gcd(a,m)=1时,$a^{phi[m]}\%m=1$

$a^b\%c=a^{(b\%phi(c)+phi(c))}\%c$ 欧拉降幂

$$a^b \equiv \begin{cases} a^{b \bmod \varphi(p)}, & \gcd(a,\,p)=1 \\ a^b, & \gcd(a,\,p)\neq 1,\, b<\varphi(p) \\ a^{b \bmod \varphi(p)+\varphi(p)}, & \gcd(a,\,p)\neq 1,\, b\geq\varphi(p) \end{cases} \pmod p$$

```
int phi(int n){
    int ret=n;
    for(int i=2;i<=sqrt(n);i++){
        if(n%i==0){
            ret=ret/i*(i-1);
            while(n%i==0)n/=i;
        }
    }
    if(n>1)ret=ret/n*(n-1);
    return ret;
}
```

6.crt与excrt

```
int crt(){//x%bi[i]=ai[i]
    int M=1,p=0;
    for (int i=1;i<=n;i++)M=M*bi[i];
    for (int i=1;i<=n;i++) {
        int w=M/bi[i],x,y;
        exgcd(w,bi[i],x,y);
        p=(p+ai[i]*w*x)%M;
    }
    return (p%M+M)%M;
}
```

excrt

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
const int N=1e5+10;
int ai[N],bi[N],n;
int qmul(int x,int y,int mod){
    int ans=0;
    while (y){
        if (y&1)ans=(ans+x)%mod;
        x=x*2%mod;y>>=1;
    }return ans;
}
int exgcd(int a,int b,int &x,int &y){
    if(b==0){
        x=1;y=0;return a;
    }
    int r=exgcd(b,a%b,x,y);
    int temp=y;y=x-(a/b)*y;x=temp;
    return r;
}
int excrt(){
    int x,y,k;
    int M=bi[1],ans=ai[1];//第一个方程的解特判
    for(int i=2;i<=n;i++){
        int a=M,b=bi[i],c=(ai[i]-ans%b+b)%b;//ax≡c(mod b)
        int gcd=exgcd(a,b,x,y),bg=b/gcd;
        if(c%gcd!=0)return -1;//判断是否无解
        x=qmul(x,c/gcd,bg);//把x转化为最小非负整数解
        ans+=x*M;//更新前k个方程组的答案
```

```
        M*=bg;
        ans=(ans%M+M)%M;
    }
    return (ans%M+M)%M;
}
signed main(){
    scanf("%lld",&n);
    for(int i=1;i<=n;++i)
        scanf("%lld%lld",&bi[i],&ai[i]);//x%bi[i]=ai[i]
    printf("%lld",excrt());
    return 0;
}
```

7.二次剩余

二次剩余：$x^2\%p=n$,给定n和p，可求x，则称n为模p的二次剩余，若x不存在，则称n为模p的非二次剩余。

求sqrt(n)%p时，x满足x^2%p=n，可用x代替sqrt(n)

这里只用Cipolla算法解决奇素数的情况：

解的数量：$x^2\%p=n$，能满足n是模p的二次剩余的个数有$(p-1)/2$个(不包括0，x=0对应n=0)，非二次剩余也有$(p-1)/2$个。

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
#define int long long
int t;
ll n,p;
ll w;
struct num {  //建立一个复数域
    ll x, y;
};
num mul(num a, num b, ll p) {  //复数乘法
    num ans = {0, 0};
    ans.x = ((a.x * b.x % p + a.y * b.y % p * w % p) % p + p) % p;
    ans.y = ((a.x * b.y % p + a.y * b.x % p) % p + p) % p;
    return ans;
}
ll binpow_real(ll a, ll b, ll p) {  //实部快速幂
    ll ans = 1;
    while (b) {
        if (b & 1) ans = ans * a % p;
        a = a * a % p;
        b >>= 1;
    }
    return ans % p;
}
ll binpow_imag(num a, ll b, ll p) {  //虚部快速幂
    num ans = {1, 0};
    while (b) {
        if (b & 1) ans = mul(ans, a, p);
        a = mul(a, a, p);
        b >>= 1;
    }
    return ans.x % p;
```

```
}
ll cipolla(ll n, ll p) {
    if (n==0)return 0;
    n %= p;
    if (p == 2) return n;
    if (binpow_real(n, (p - 1) / 2, p) == p - 1) return -1;
    //欧拉判别准则n^(p-1)/2%p==1为二次剩余==-1为非二次剩余
    ll a;
    while (1) {  //生成随机数再检验找到满足非二次剩余的a
        a = rand() % p;
        w = ((a * a % p - n) % p + p) % p;
        if (binpow_real(w, (p - 1) / 2, p) == p - 1) break;
    }
    num x = {a, 1};
    return binpow_imag(x, (p + 1) / 2, p);
}
signed main(){
    int T;
    cin>>T;
    while (T--){
        cin>>n>>p;
        int t=cipolla(n,p);
        if (t==-1)puts("Hola!");//没有二次剩余
        else if (t==0)cout<<t<<endl;//重根
        else{
            int s=p-t;
            if (s>t)swap(s,t);
            cout<<s<<" "<<t<<endl;
        }
    }
    return 0;
}
```

8.原根

使得$a^l\%p=1$成立的最小的l，称为a关于模p的阶，记为$ord_m a$

$gcd(a,m)=1$r44时，若$ord_m a=\phi(m)$，则a为$\%m$的一个原根

有无原根判断：m有原根，则它必定是$2,4,p^a,2*p^a$(p为奇素数，a为正数)其中之一

求一个原根：$gcd(g,m)=1$，设$p_1,p_2,...,p_k$是$\phi(m)$的所有不同的素因子，则g是m的原根，当且仅当对任意$1\le i\le k$，都有$g^{\frac{\phi(m)}{p_i}}\%m!=1$

求所有原根：设g为m的一个原根，则集合$S=\{g^s|1\le s\le\phi(m),gcd(s,\phi(m))=1\}$，给出m的全部原根。因此，若$m$有原根，则$m$有$\phi(\phi(m))$个关于模$m$两两互不同余的原根。

```
//暴力求原根，若一个数m存在原根，可以证明他的最小原根在O(m^1/4)级别。
vector<int>v;//存phi[m]素数
int phi(int n){
    int ret=n;
    for(int i=2;i<=sqrt(n);i++){
        if(n%i==0){
            ret=ret/i*(i-1);
            while(n%i==0)n/=i;
        }
    }
    if(n>1)ret=ret/n*(n-1);
```

```
    return ret;
}
int gcd(int x,int y){
    return y==0?x:gcd(y,x%y);
}
int qpow(int x,int y,int p){
    int ans=1;
    while (y){
        if (y&1)ans=ans*x%p;
        y>>=1;x=x*x%p;
    }return ans;
}
bool isprimepow(int x){
    int cnt=0;
    if (x%2==0)return 0;
    for (int i=3;i*i<=x;i++){
        if (x%i==0){
            cnt++;
            while (x%i==0)x/=i;
        }
    }
    if (x>1)cnt++;
    if (cnt==1)return 1;
    return 0;
}
int solve(int p){
    int f=0,r=p;
    if (r==2||r==4)f=1;
    if (r%2==0)r/=2;
    if (isprimepow(r))f=1;
    if (!f)return -1;
    int m=phi(p),t=m;
    v.clear();
    for (int i=2;i*i<=t;i++){
        if (t%i==0){
            v.push_back(i);
            while (t%i==0)t/=i;
        }
    }
    if (t>1)v.push_back(t);
    for (int i=1;i<p;i++){
        bool f=0;
        if (gcd(i,p)!=1)continue;
        for (int j=0;j<v.size();j++){
            if (qpow(i,m/v[j],p)==1){
                f=1;break;
            }
        }
        if (!f)return i;
    }
}
//上面求最小原根，下面求所有原根
vector<int>s;
void find(int n){
    s.clear();
    int g=solve(n);
    if (g==-1)return;
    int m=phi(n);
```

```
    int t=1;
    for (int i=1;i<=m;i++){
        t=t*g%n;
        if (gcd(m,i)==1){
            s.push_back(t);
        }
    }
    sort(s.begin(),s.end());
}
```

9.BSGS和exBSGS

O(sqrt(p))时间求出a^x%p=b的解x

//a^x%p=b，a,p互质

令 $x=A\left\lceil\sqrt{p}\right\rceil-B$，其中 $0\leq A,B\leq\left\lceil\sqrt{p}\right\rceil$，则有 $a^{A\lceil\sqrt{p}\rceil-B}\equiv b\pmod p$，稍加变换，则有 $a^{A\lceil\sqrt{p}\rceil}\equiv ba^B\pmod p$。

我们已知的是 a,b，所以我们可以先算出等式右边的 ba^B 的所有取值，枚举 B，用 `hash` / `map` 存下来，然后逐一计算 $a^{A\lceil\sqrt{p}\rceil}$，枚举 A，寻找是否有与之相等的 ba^B，从而我们可以得到所有的 x， $x=A\left\lceil\sqrt{p}\right\rceil-B$。

注意到 A,B 均小于 $\left\lceil\sqrt{p}\right\rceil$，所以时间复杂度为 $\Theta\left(\sqrt{p}\right)$，用 `map` 则多一个 $\log$。

```
int solve(int a,int b,int p){//a^ans%p==b
    a%=p;b%=p;
    static unordered_map<int,int>s;s.clear();
    int block=sqrt(p)+1,t=1;
    for (int i=0;i<block;i++){//b*a^B,B=i
        s[b*t%p]=i;
        t=t*a%p;
    }
    int r=1;
    for (int i=0;i<=block;i++){//a^(A*block),A=i
        if (s.count(r)){
            int e=i*block-s[r];//递增的
            if(e>=0)return e;
        }
        r=r*t%p;
    }
    return -1;
}
```

//x^a%p=b,p为质数

单解 $x=g^c$，g为原根，$(g^c)^a\%p=(g^a)^c\%p=b$,BSGS求c

所有解

$$\forall\, i\in\mathbb{Z}, x\equiv g^{c+\frac{\varphi(p)}{\gcd(a,\varphi(p))}\cdot i}\pmod p$$

```
int solve(int a,int b,int p){
    int g=solve1(p);//原根
    int t=qpow(g,a,p);
    int c=solve2(t,b,p);//BSGS
    return qpow(g,c,p);
}
```

//a^x%p=b，p无要求

```
int gcd(int x,int y){
    return y==0?x:gcd(y,x%y);
}
int qpow(int x,int y,int p){
    int ans=1;
    while (y){
        if (y&1)ans=ans*x%p;
        y>>=1;x=x*x%p;
    }return ans;
}
void ex_gcd(int a,int b,int &x,int &y){
    if (!b){x=1,y=0;return;}
    ex_gcd(b,a%b,x,y);
    int t=x;x=y;y=t-(a/b)*y;
}
int inv(int a,int p) {
    int inv_a,y;
    ex_gcd(a,p,inv_a,y);
    return (inv_a%p+p)%p;//因为可能为负
}
int solve(int a,int b,int p){
    int d=1,k=0,r=b,e=p;
    int t=gcd(a,e);
    int f=0;
    while (t>1){
        e/=t;d*=t;k++;
        if (r%t){
            f=1;break;
        }
        else r/=t;
        t=gcd(a,e);
    }
    for (int i=0,o=1;i<=k;i++){
        if (o==b)return i;
        o=o*a%p;
    }
    if (f)return -1;
    int ans=solve1(a,b*inv(qpow(a,k,e)%e,e)%e,e);//BSGS
    if (ans>0)return ans+k;
    return -1;
}
```

10.lucas定理和exlucas

对于质数p，

4

p为质数时，$(a+b)^p\%p=(a^p+b^p)\%p$

```
int qpow(int x,int y,int p){
    int ans=1;
    while (y){
        if (y&1)ans=ans*x%p;
        y>>=1;x=x*x%p;
    }return ans;
}
int C(int n,int m,int p){
    if(n<m) return 0;
    if(m>n-m) m=n-m;
    int a=1,b=1;
    for(int i=0;i<m;i++){
        a=(a*(n-i))%p;
        b=(b*(i+1))%p;
    }
    return a*qpow(b,p-2,p)%p;
}
int Lucas(int n,int m,int p) {
    if (m==0)return 1;
    return (C(n%p,m%p,p)*Lucas(n/p,m/p,p))%p;
}
```

exlucas定理(p不一定为素数)

11.数论函数和莫比乌斯反演

狄利克雷卷积:$(f*g)(n)=\sum_{d|n}f(d)g(\frac{n}{d})$,满足交换律，结合律

积性函数：对于任意互质的整数a,b,有 $f(ab)=f(a)f(b)$ 的数论函数。

$f(x),g(x)$为积性函数，则$h(x)=f(x^p),h(x)=f*g,h(x)=f(x)^p,h(x)=f(x)g(x)$也是积性函数

$\phi(n)=\sum_{i=1}^{n}[gcd(i,n)==1]$

$$\mu(n)=\begin{cases}1, & n=1\\(-1)^k, & n\text{无平方因数，}n=p_1\dots p_k\\0, & n\text{有大于}1\text{的平方因数}\end{cases}$$

$\sum_{d|i}\mu(d)=0,if\ \ i!=1$

$\sigma_k(n)=\sum_{d|n}d^k$

$d(n)=\sum_{d|n}1$

完全积性函数：对于任意整数a,b,有 $f(ab)=f(a)f(b)$的数论函数。

$\epsilon(n)=[n==1]$

$I(n)=1$

$id(n)=n$

莫比乌斯反演，条件:$g=f*I$

$g(n)=\sum_{d|n}f(d)\rightarrow f(n)=\sum_{d|n}\mu(d)g(\frac{n}{d})$

$g(n)=\sum_{n|d}f(d)\rightarrow f(n)=\sum_{n|d}\mu(\frac{d}{n})g(d)$

例：f(d)为gcd为d的个数，g(d)为gcd为d倍数的个数

$\mu * I=\epsilon$

$\phi * I=id$

$\mu * id=\phi$

$[gcd(i,j)==1]=\sum_{d|gcd(i,j)}\mu(d)$

$d(ij)=\sum_{d|ij}1=\sum_{x|i}\sum_{y|j}[gcd(x,y)==1]$

整除分块：

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

12.杜教筛

求$Sn=\sum_{i=1}^{n}f(i)$

找一个$g(i)$使得$h=f*g$能够O(1)求解,

$\sum_{i=1}^{n}h(i)$
$=\sum_{i=1}^{n}(f*g)(i)$
$=\sum_{i=1}^{n}\sum_{d|i}f(d)g(\frac{i}{d})$
$=\sum_{d=1}^{n}g(d)\sum_{i=1}^{\lfloor\frac{n}{d}\rfloor}f(i)$
$=\sum_{d=1}^{n}g(d)S(\lfloor\frac{n}{d}\rfloor)$

可以发现$g(1)S(n)=\sum_{i=1}^{n}h(i)-\sum_{i=2}^{n}g(i)S(\lfloor\frac{n}{i}\rfloor)$

只需找一个较好的$g(n)$可以快速处理前面的式子，后面的式子可以数论分块，递归调用S

求μ的前缀和：$\mu * I=\epsilon$

求ϕ的前缀和：$\phi * I=id$

$\sum_{j|i}f(j)=f*I$

求$\sum_{i=1}^{n}\phi(i)*i$：$(\phi(n)*id(n))*id(n)=n^2$

```
int S(int n){
    if (n<N)return s[n];//预处理，越多越快
    if (p[n])return p[n];//记忆化
    int ans=sum_fg(n);//sum_fg为f*g的前缀和
    for(int l=2,r;l<=n;l=r+1){
        r=(n/(n/l));
        ans-=(sum_g(r)-sum_g(l-1))*S2(n/l);//sum_g为g的前缀和
    }
    return p[n]=ans;
}
```

13.单位根反演

$\frac{1}{k}\sum_{i=0}^{k-1}\omega_k^{in}=[k|n]$

$p\equiv 1(mod\ k)$,$\omega_k=g^{\frac{p-1}{k}}$, g为原根

14.min25筛

1.求n以内素数个数

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
namespace Min25{
    #define PB push_back
    typedef long long ll;
    vector<int>vis,prime,g;vector<vector<int> >f;
    ll sqr(ll x){return x*x;}
    int getsqrt(ll n){
        ll t=sqrt(n);
        while(sqr(t+1)<=n)t++;
        return t;
    }
    int getcbrt(ll n){
        ll t=cbrt(n);
        while((t+1)*(t+1)*(t+1)<=n)t++;
        return t;
    }
    void sieve(int n){
        vis.assign(n+1,0);
        prime.clear();
        for(int i=2;i<=n;i++){
        if(!vis[i]){
            prime.PB(i);
            vis[i]=i;
        }
        for(int j=0;j<(int)(prime.size());j++){
            if(prime[j]>vis[i]||prime[j]>n/i)break;
                vis[i*prime[j]]=prime[j];
            }
        }
    }
    ll PI(ll n);
    ll F(ll n,int m){
        if(!m)return n;
        if(prime[m]>n)return 0;
        if(m<(int)(f.size())&&n<(int)(f[m].size()))return f[m][n];
        if(sqr(prime[m])>n)return PI(n)-m+1;
        return F(n,m-1)-F(n/prime[m-1],m-1);
    }
    ll PI(ll n){
        if(n<=prime.back())return g[n];
        int i=PI(getcbrt(n-1)+1);ll tmp=F(n,i)+i-1;
        while(1){
            ll t=prime[i];
            if(t*t>n)break;
            tmp-=PI(n/t)-i;
            i++;
```

```
        }
        return tmp;
    }
    void init(ll n){
        sieve(getsqrt(n+1)*2);
        g.assign(prime.back()+1,0);
        int i,j;
        for(i=j=0;i<=prime.back();g[i++]=j)if(i==prime[j])j++;
        int A=131072,B=min((int)prime.size(),64);
        f.assign(B,vector<int>(A));
        for(j=0;j<B;j++)for(i=0;i<A;i++)if(j==0)f[j][i]=i;
        else f[j][i]=f[j-1][i]-f[j-1][i/prime[j-1]];
    }
}

signed main(){
    Min25::init(100000000000LL);
    ll n;
    while(~scanf("%lld",&n))printf("%lld\n",Min25::PI(n));
}
```

2.求n内p^k和

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
// return the sum of p^k for all p <= m, where m is in form floor(n / i)
// for m <= sqrt{n}, stored in ssum[m]; for m > sqrt{n} stored in lsum[n / m]
// note: if you need all correct value of ssum and lsum, please remove ”mark”
// and make ”delta” always be 1
inline ll sub_mod(ll x,ll y,ll mod){return(x-y+mod)%mod;}
inline ll mul_mod(ll a,ll b,ll mod){
    if(mod<int(2e9))return a*b%mod;
    ll k=(ll)((long double)a*b/mod);
    ll res=a*b-k*mod;
    res%=mod;
    if(res<0)res+=mod;
    return res;
}
inline ll pow_mod(ll a,ll n,ll m){
    ll res=1;
    for(a%=m;n;n>>=1){
        if(n&1)res=mul_mod(res,a,m);
        a=mul_mod(a,a,m);
    }
    return res;
}
pair<vector<ll>,vector<ll>>prime_count(ll n,ll k,ll mod){
    auto pow_sum=[](ll n,ll k,ll mod){
        if(k==0)return n;
        if(k==1)return n*(n+1)/2%mod;
    };
    const ll v=static_cast<ll>(sqrt(n));
    vector<ll>ssum(v+1),lsum(v+1);
    vector<bool>mark(v+1);
    for(int i=1;i<=v;++i){
        ssum[i]=pow_sum(i,k,mod)-1;
```

```
        lsum[i]=pow_sum(n/i,k,mod)-1;
    }
    for(ll p=2;p<=v;++p){
        if(ssum[p]==ssum[p-1])continue;
        ll psum=ssum[p-1],q=p*p,ed=min(v,n/q);
        ll pk=pow_mod(p,k,mod);
        int delta=(p&1)+1;
        for(int i=1;i<=ed;i+=delta)if(!mark[i]){
            ll d=i*p;
            if(d<=v){
                lsum[i]=sub_mod(lsum[i],sub_mod(lsum[d],psum,mod)*pk%mod,mod);
            }else{
                lsum[i]=sub_mod(lsum[i],sub_mod(ssum[n/d],psum,mod)*pk%mod,mod);
            }
        }
        for(ll i=q;i<=ed;i+=p*delta)mark[i]=1;
        for(ll i=v;i>=q;--i){
            ssum[i]=sub_mod(ssum[i],sub_mod(ssum[i/p],psum,mod)*pk%mod,mod);
        }
    }
    return {move(ssum),move(lsum)};
}
signed main(){
    ll n,k,mod;
    scanf("%lld%lld%lld",&n,&k,&mod);
    auto it=prime_count(n,k,mod);
    printf("%lld",it.second[1]);
}
```

3.单次

```
//素数个数
#include <bits/stdc++.h>
using namespace std;
const int N=1000005;
struct Min25{
    typedef long long ll;
    ll prime[N],id[2][N],a[N],g[N],n,m,tot,T;
    int b[N];
    void init(int x){
        n=x;
        #define ID(x) ((x)<=T?id[0][(x)]:id[1][n/(x)])
        T=sqrt(n);
        for(ll l=1;l<=n;l=n/(n/l)+1){
            a[++m]=n/l,g[m]=a[m]-1;
            id[a[m]<=T?0:1][a[m]<=T?a[m]:n/a[m]] = m;
        }
        for(int i=2;i<=T;i++){
            if(!b[i]) prime[++tot]=i;
            for(int j=1;j<=tot && i * prime[j] <= T; j++){
                b[i*prime[j]]=1;
                if(i%prime[j]==0)break;
            }
        }
        for(int i = 1; i <= tot; i++)
            for(int j = 1; j <= m && prime[i]*prime[i] <= a[j]; j++)
                g[j]-=g[ID(a[j]/prime[i])]-(i-1);
```

```
    }
    int get(int x){
        if (x<2)return 0;
        #define ID(x) ((x)<=T?id[0][(x)]:id[1][n/(x)])
        return g[ID(x)];
    }
}min25;
int n;
int main(){
    cin>>n;
    min25.init(n);
    cout<<min25.get(n)<<endl;
    return 0;
}
//素数和
#include <bits/stdc++.h>
#define ll long long
using namespace std;
const int N = 1000010;
int prime[N], id1[N], id2[N], flag[N], ncnt, m;
ll g[N], sum[N], a[N], T;
ll n;
int ID(ll x) {return x <= T ? id1[x] : id2[n / x];}
ll calc(ll x) {return x * (x + 1) / 2 - 1;}
ll f(ll x) {return x;}
ll init(ll n){
    T = sqrt(n + 0.5);
    for (int i = 2; i <= T; i++) {
        if (!flag[i]) prime[++ncnt] = i, sum[ncnt] = sum[ncnt - 1] + i;
        for (int j = 1; j <= ncnt && i * prime[j] <= T; j++) {
            flag[i * prime[j]] = 1;
            if (i % prime[j] == 0) break;
        }
    }
    for (ll l = 1; l <= n; l = n / (n / l) + 1) {
        a[++m] = n / l;
        if (a[m] <= T) id1[a[m]] = m; else id2[n / a[m]] = m;
        g[m] = calc(a[m]);
    }
    for (int i = 1; i <= ncnt; i++)
        for (int j = 1; j <= m && (ll)prime[i] * prime[i] <= a[j]; j++)
            g[j] = g[j] - (ll)prime[i] * (g[ID(a[j] / prime[i])] - sum[i - 1]);
}
ll solve(ll x){
    if(x<=1){return x;}
    return n=x,init(n),g[ID(n)];
}
int main() {
    while(1){
        memset(g,0,sizeof(g));
        memset(a,0,sizeof(a));
        memset(sum,0,sizeof(sum));
        memset(prime,0,sizeof(prime));
        memset(id1,0,sizeof(id1));
        memset(id2,0,sizeof(id2));
        memset(flag,0,sizeof(flag));
        ncnt=m=0;
        scanf("%lld", &n);
```

```
        printf("%lld\n", solve(n));
    }
}
```

20.其他

快速阶乘.

斯特林公式$n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$,可求阶乘位数

分块打表