基础数学

高胜寒

快速幂

- 如何计算$a^b \bmod P$的结果？
- $b \le 10^7$？
- 可以暴力实现：

```
int ans=1;
for(int i=1;i<=b;i++)ans=1ll*ans*a%p;
```

- $b \le 10^{18}$？
- 需要找到一个更好的方法解决。

快速幂

- 我们首先将b表示成二进制
- 举一个例子：
- $3^{13} = 3^{1101_2} = 3^8 \times 3^4 \times 3^1$
- 如果我们知道了$\mathrm{a}^1, a^2, a^4, a^8, a^{2^{\log_2 n}}$后，我们只需要计算$O(\log n)$次乘法。
- 而$\mathrm{a}^1, a^2, a^4, a^8, a^{2^{\log_2 n}}$也可以在$O(\log n)$次之内计算，只需要不断对前一项做平方即可。
- 举个例子：
- $3^1 = 3$ $\qquad$ $3^2 = 3 \times 3 = 9$
- $3^4 = 9 \times 9 = 81$ $\qquad$ $3^8 = 81 \times 81 = 6561$

如何实现

- 一般有两种实现方式，推荐使用非递归，常数更小。
- 建议记住。

```
int pow(int a,int b,int p)
{
    if(b==0)return 1;
    int ans=pow(a,b/2,p);
    if(b%2)return 1ll*ans*ans%p*a%p;
    return 1ll*ans*ans%p;
}
```

```
int pow(int a,int b,int p)
{
    int ans=1;
    while(b)
    {
        if(b%2)ans=1ll*ans*a%p;
        a=1ll*a*a%p;
        b/=2;
    }
    return ans;
}
```

一些基本概念

- 同余：
- 设整数$m \neq 0$，若$m|(a-b)$，则称$a \equiv b \pmod m$，a和b在$\bmod m$意义下同余，也可以理解为$a \bmod m = b \bmod m$。

取模的性质

- $(a+b)\%m = (a\%m + b\%m)\%m$
- $(a-b)\%m = (a\%m - b\%m)\%m$
- $(a\times b)\%m = \left((a\%m)\times(b\%m)\right)\%m$
- 注意：除法不满足类似的性质

同余的性质

- 如果没有特殊说明，我们这一页的同余都是$\bmod m$意义下的
- 自反性：$a \equiv a$
- 对称性：$a \equiv b \Leftrightarrow b \equiv a$
- 传递性：$a \equiv b, b \equiv c \Rightarrow a \equiv c$
- 线性运算：若$a \equiv b, c \equiv d$，则$a + c \equiv b + d, a \times c \equiv b \times d$
- 这些性质实质上与相等的性质是类似的。

素数判断

- 如何判断一个数n是不是素数？
- $n \le 10^7$？
- 暴力枚举每个$\le n$的数，如果可以整除n就不是
- $n \le 10^{14}$？
- 考虑一个性质。
- n如果有$\ge \sqrt{n}$的因子b，那一定存在$a \le \sqrt{n}$，使得$n = ab$
- 这样我们只需要对$1,2,3,\dots,\sqrt{n}$枚举判断即可。
- 同时，这也为快速枚举n的所有约数提供了思路。

如何实现

```cpp
bool isprime(long long n)
{
    if(n<=1)return 0;
    for(int i=2;1ll*i*i<=n;i++)
    {
        if(n%i==0)return 0;
    }
    return 1;
}
```

筛法

- 我们现在知道如何判断一个数n是素数。
- 如果我想知道$1-n$的所有数是不是素数呢?
- 当然可以一个一个判断。复杂度$O(n\sqrt{n})$。

筛法

- 如何更快判断？
- 如果我们将一个数判定为素数，那么这个数的所有倍数都是合数。
- 于是我们可以使用下面的方法：
- 复杂度$O(n\log n)$

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

埃氏筛

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

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

- 容易发现，对于$j<i\times i$，$j=ik$，已经被k筛过了，所以我们从$i\times i$开始筛。
- 复杂度$O(n\log\log n)$。
- 证明如果有兴趣可以自行阅读OI Wiki
- https://oiwiki.org/math/number-theory/sieve/

线性筛

- 在之前的筛法中，仍然可能出现一个合数被筛了多次的情况，比如$12 = 2 \times 6 = 3 \times 4$，仍然没有达到理论复杂度下限$O(n)$。
- 我们需要保证一个合数只被筛一次。

线性筛

- 建议记住。

```
for(int i=2;i<=n;i++)
{
    if(!notprime[i])
    {
        prime[++tot]=i;
        for(int j=1;j<=tot&&i*prime[j]<=n;j++)
        {
            notprime[i*prime[j]]=1;
            if(i%prime[j]==0)break;
        }
    }
}
```

P1835 素数密度

- 计算区间$[l,r]$中素数个数。
- $1 \le l \le r < 2^{31}, r - l \le 10^6$。

Solution

- 直接暴力一个一个判断显然不可行。
- 考虑使用筛法。
- 我们知道，一个合数x的最小质因子p一定满足$p^2 \le x$。
- 那么，我们只要用$\le \sqrt{2^{31}}$的所有质数，去筛$[l, r]$区间内的合数即可。复杂度大约是$O((r-l)\log(r-l))$。

如何实现

```cpp
for(int i=2;i<=lim;i++)
{
    if(!notprm[i])
    {
        prm[++cnt]=i;
        for(int j=i+i;j<=lim;j+=i)notprm[j]=1;
    }
}
for(int i=1,j=prm[1];i<=cnt;j=prm[++i])
{
    ll k=max(1ll*(l+j-1)/j*j,2ll*j);
    while(k<=r)vis[k-l]=1,k+=j;
}
if(l==1)vis[0]=1;
int ans=0;
for(int i=l;i<=r;i++)ans+=!vis[i-l];
```

分解质因数

- 如何将一个数x分解质因数？
- $x \leq 10^{14}$

- 考虑x的素因子p，一定可分为两类：
- $p \leq \sqrt{n}$
- $p > \sqrt{n}$
- 第二类的p至多只有一个

如何实现

```cpp
vector<int>calc(int n)
{
    vector<int>result;
    for(int i=2;i*i<=n;i++)
    {
        if(n%i==0)
        {
            while(n%i==0)n/=i;
            result.push_back(i);
        }
    }
    if(n>1)result.push_back(n);
    return result;
}
```

最大公约数（GCD）

- 如何求a, b的最大公约数?

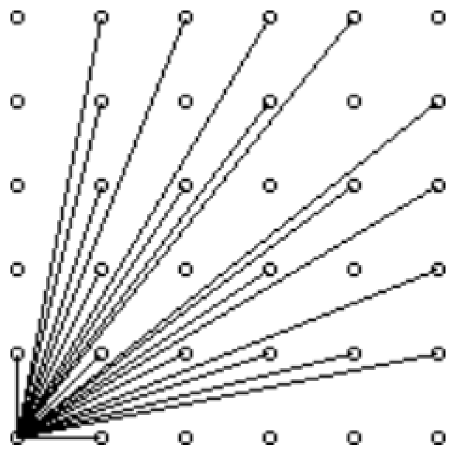
欧几里得算法（辗转相除法）

- $\gcd(a,b) = \gcd(b, a \bmod b)$
- 证明：
- 设$a = bk + c, c = a \bmod \mathrm{b}$。
- 令$d|a, d|b$，则$\frac{c}{d} = \frac{a}{d} - \frac{b}{d}k$为整数，则$d|c$。
- 反过来，令$d|b, d|c$，则$\frac{a}{d} = \frac{c}{d}k + \frac{b}{d}$为整数，则$d|a$。
- 所以公约数相同，则最大公约数相同。
- 复杂度$O(\log\max(a,b))$
- 当然，也可以使用g++编译器内建函数 __gcd

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

[SDOI2008] 仪仗队

作为体育委员，C 君负责这次运动会仪仗队的训练。仪仗队是由学生组成的 $N \times N$ 的方阵，为了保证队伍在行进中整齐划一，C 君会跟在仪仗队的左后方，根据其视线所及的学生人数来判断队伍是否整齐（如下图）。



现在，C 君希望你告诉他队伍整齐时能看到的学生人数。

- $1 \leq N \leq 40000$
- 样例：输入4 输出9

Solution

- 我们不妨将左下角的下标看成(0,0)，这样一个位置的下标就表示这个位置与左下角两个坐标方向的距离。
- 在第0行和第0列，一定只能看到第一个。
- 考虑$[1,n-1]\times[1,n-1]$这些区域的(x,y)能否被看到取决于什么。
- (x,y)看不到当且仅当存在(x_1,y_1)使得$\frac{x_1}{x}=\frac{y_1}{y}<1$。
- 这等价于$\gcd(x,y)\neq 1$。
- 那么，我们的最终答案就是$2+\sum_{i=1}^{n-1}\sum_{j=1}^{n-1}[\gcd(i,j)=1]$。

Solution

- 这个式子怎么求：

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

- 我们令$f(x)=\sum_{i=1}^{n-1}\sum_{j=1}^{n-1}[\gcd(i,j)=x]$，$g(x)=\sum_{i=1}^{n-1}\sum_{j=1}^{n-1}[x|\gcd(i,j)]$。
- 如果$x|i,x|j$，那么$x|\gcd(i,j)$，这样$g(x)=\left\lfloor\frac{n-1}{x}\right\rfloor^2$是显然的。
- 考虑一个容斥，$f(x)=g(x)-f(2x)-f(3x)-\cdots$。
- 对x从大到小依次计算即可。

如何实现

```
cin>>n;
if(n==1)
{
    cout<<0<<'\n';
    return;
}
n--;
for(int i=n;i;i--)
{
    f[i]=(n/i)*(n/i);
    for(int j=2*i;j<=n;j+=i)f[i]-=f[j];
}
cout<<f[1]+2<<'\n';
```

扩展欧几里得算法（EXGCD）

- 求解方程$ax+by=\gcd(a,b)$ 的一组解。
- 设$ax_1+by_1=\gcd(a,b)$，$bx_2+(a \bmod b)y_2=\gcd(b,a \bmod b)$。
- 可得$ax_1+by_1=bx_2+(a \bmod b)y_2$
- $a \bmod b=a-\left\lfloor\frac{a}{b}\right\rfloor b$
- $ax_1+by_1=bx_2+\left(a-\left\lfloor\frac{a}{b}\right\rfloor b\right)y_2=ay_2+b\left(x_2-\left\lfloor\frac{a}{b}\right\rfloor y_2\right)$
- 故$x_1=y_2, y_1=x_2-\left\lfloor\frac{a}{b}\right\rfloor y_2$
- $a=1, b=0$时，显然存在一组解$x=1, y=0$
- 复杂度$O(\log\max(a,b))$

如何实现

- 建议记住。

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

[NOIP2012 提高组] 同余方程

- 求关于 x 的同余方程 $ax \equiv 1 \pmod b$ 的最小正整数解。保证有解。
- $a, b \leq 2 \times 10^9$

Solution

- 我们容易发现，$ax \equiv 1 \pmod b$等价于$ax + by = 1$。
- 如果$\gcd(a, b) \neq 1$，不妨设其为d，则$d|a, d|b$，所以$d|ax + by$，显然a不是d的倍数，方程无解，矛盾。所以$\gcd(a, b) = 1$
- 使用exgcd求出方程的一组可行解x, y，得到$ax \equiv 1 (\bmod b)$
- 容易发现，$a(x + b) \equiv 1 (\bmod b)$。
- 将x对b取模可以得到最小正整数解。
- 为什么不会错过更小的解？
- 可以参考
- P5656 【模板】二元一次不定方程 (exgcd) 题解。

费马小定理

- 若p为素数，$\gcd(a, p) = 1$，则$a^{p-1} \equiv 1 \pmod p$。
- 另一种形式：对任意整数a，有$a^p \equiv a \pmod p$。
- 证明：
- 显然$1^p \equiv 1 \pmod p$。假设$a^p \equiv a \pmod p$成立
- $(a+1)^p = a^p + \binom{p}{1}a^{p-1} + \binom{p}{2}a^{p-2} + \cdots + \binom{p}{p-1}a + 1$
- 模p意义下，$\binom{p}{1} \equiv \binom{p}{2} \equiv \cdots \equiv \binom{p}{p-1} \equiv 0 \pmod p$
- 则$(a+1)^p \equiv a^p + 1 \equiv a + 1 \pmod p$
- 得证。
- 非常重要，建议记住。

乘法逆元

- 定义：如果一个线性同余方程$ax \equiv 1 \pmod b$，则x是$a \bmod b$的逆元，记为a^{-1}。
- 是不是非常熟悉？
- 你已经学会了如何求一个数的逆元了！
- 下面再介绍一种方法。
- 由于$ax \equiv 1 \pmod b$，则$ax \equiv a^{b-1} \pmod b$
- 所以$x \equiv a^{b-2} \pmod b$，使用快速幂求逆元即可。
- 注意：这种方法要求b是素数。
- 对于b不是素数的情况，依然可以用快速幂求逆元，有兴趣可以阅读OI Wiki欧拉定理
- https://oiwiki.org/math/number-theory/fermat/#%E6%AC%A7%E6%8B%89%E5%AE%9A%E7%90%86

线性求任意n个数的逆元

- 如何在线性时间内求任意n个数$a_1, a_2, \ldots, a_n$的逆元?
- 如果逐个求，复杂度$O(n \log p)$，不能满足需求。
- 首先计算n个数的前缀积$s_i = \prod_{j=1}^{i} a_j$。
- 使用前面的方法计算s_n的逆元，记为sv_n。
- 对s_i的逆元sv_i，只需要将sv_{i+1}乘上a_{i+1}，即可求得。
- $a^{-1} = s_{i-1} \times sv_i$
- 还有一种线性求$1-n$的逆元的方法，这里不做介绍，复杂度相同，但是递推式略复杂，有兴趣可以阅读OI Wiki
- https://oiwiki.org/math/number-theory/inverse/#%E7%BA%BF%E6%80%A7%E6%B1%82%E9%80%86%E5%85%83

如何实现

- 建议记住。

```
s[0]=1;
for(int i=1;i<=n;i++)s[i]=1ll*s[i-1]*a[i]%p;
sv[n]=pow(s[n],p-2,p);
for(int i=n-1;i>=0;i--)sv[i]=1ll*sv[i+1]*a[i+1]%p;
for(int i=1;i<=n;i++)inv[i]=1ll*sv[i]*s[i-1]%p;
```

裴蜀定理

- $ax + by = c$存在整数解当且仅当$\gcd(a,b)|c$
- 通过EXGCD，我们可以得出 $ax + by = \gcd(a,b)$ 的一组整数解。
- 那么我们只需要将这组整数解扩大$\frac{c}{\gcd(a,b)}$倍，那么就可以得到$ax + by = c$的整数解。
- 如果c不是$\gcd(a,b)$的倍数，又由$\gcd(a,b)\,|(ax + by)$，那么一定无解。

扩展中国剩余定理（EXCRT）

- 求解如下形式的一元线性同余方程组

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \quad\quad\cdots \\ x \equiv a_n \pmod{m_n} \end{cases}$$

扩展中国剩余定理（EXCRT）

- 考虑只有两个方程如何求解

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \end{cases}$$

- 转化：$x = m_1 p + a_1 = m_2 q + a_2 \Rightarrow m_1 p - m_2 q = a_2 - a_1$
- 由裴蜀定理，当且仅当$\gcd(m_1, m_2) | a_2 - a_1$时有解，使用EXGCD求出一组可行解$(p, q)$
- 合并成以下的方程：

$$x \equiv m_1 p + a_1 \pmod{\operatorname{lcm}(m_1, m_2)}$$

- 方程在$\bmod \operatorname{lcm}(m_1, m_2)$意义下有唯一解。
- 多个方程两两合并即可。
- 这个唯一解证明可以参考阮行止的博客 https://www.luogu.com.cn/article/lr8vtpzl

如何实现

```
void merge(ll&a1,ll&m1,ll a2,ll m2)
{
    ll g=gcd(m1,m2),l=m1/g*m2;//gcd and lcm
    ll d=(a2-a1)/g;
    ll p,q;
    EXGCD(m1,m2,p,q);
    a1=(((__int128_t)m1*p*d+a1)%l+l)%l;
    m1=l;
}
```

[CQOI2007] 余数求和

- 计算$\sum_{i=1}^{n} k \bmod i$。
- $n, k \le 10^9$

数论分块

$$\Sigma_{i=1}^{n} k \bmod i$$
$$= \Sigma_{i=1}^{n}\left(k - \left\lfloor\frac{k}{i}\right\rfloor \times i\right)$$
$$= nk - \Sigma_{i=1}^{n} i \times \lfloor\frac{k}{i}\rfloor$$

- 考虑后面那个式子怎么算。
- 不难发现，对于所有的i，$\lfloor\frac{k}{i}\rfloor$至多有$2\sqrt{k}+1$种取值。
- 证明：$1 \le i \le \sqrt{k}$，$\lfloor\frac{k}{i}\rfloor$有$\sqrt{k}$种取值。
- $\sqrt{k} \le i \le n$，$0 \le \left\lfloor\frac{k}{i}\right\rfloor \le \sqrt{k}$，所以有$\sqrt{k}+1$种取值。

数论分块

- 显然，对每一种$\left\lfloor \frac{k}{i} \right\rfloor$的不同取值，都有一个连续区间$[l,r] \subset [1,n]$。
- 我们只要对每个连续区间求出答案加起来即可。
- 一个结论：
- 使得$\left\lfloor \frac{k}{i} \right\rfloor = \left\lfloor \frac{k}{j} \right\rfloor$，最大且满足$i \le j \le k$的$j$为$\left\lfloor \frac{k}{\left\lfloor \frac{k}{i} \right\rfloor} \right\rfloor$。
- 证明参考OI Wiki
- https://oiwiki.org/math/number-theory/sqrt-decomposition/#%E6%95%B0%E8%AE%BA%E5%88%86%E5%9D%97%E7%BB%93%E8%AE%BA

如何实现

```
int n,k;
cin>>n>>k;
ll ans=1ll*n*k;
for(int l=1,r;l<=min(n,k);l=r+1)
{
    r=min(n,k/(k/l));
    ans-=1ll*(k/l)*(r-l+1)*(l+r)/2;
}
cout<<ans<<'\n';
```

P2424 约数和

对于一个数 X，函数 $f(X)$ 表示 X 所有约数的和。例如：$f(6)=1+2+3+6=12$。对于一个 X，Smart 可以很快的算出 $f(X)$。现在的问题是，给定两个正整数 $X,Y(X<Y)$，Smart 希望尽快地算出 $f(X)+f(X+1)+\ldots\ldots+f(Y)$的值，你能帮助 Smart 算出这个值吗？

- $1 \le x < y \le 2 \times 10^9$。

Solution

- 首先转换成计算$\Sigma_{i=1}^{y} f(i) - \Sigma_{i=1}^{x-1} f(i)$。我们只需要知道怎么计算$f(x)$的前缀和即可。

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

- 这个式子我们刚算过。

组合数

- 组合数$\binom{n}{m}$如何快速求解？
- $\binom{n}{m}=\binom{n-1}{m-1}+\binom{n-1}{m}$，边界条件$\binom{n}{0}=\binom{n}{n}=1$。
- 复杂度$O(n^2)$，可以求出$0-n$所有的组合数。
- 如果我们对于n,m比较大的情形，需要快速计算$\binom{n}{m}$？
- $\binom{n}{m}=\frac{n!}{m!(n-m)!}$。需要用到乘法逆元，预处理复杂度$O(n)$，单次查询复杂度$O(1)$。

如何实现

```
for(int i=0;i<=n;i++)C[i][0]=C[i][i]=1;
for(int i=1;i<=n;i++)
    for(int j=1;j<i;j++)
        C[i][j]=(C[i-1][j]+C[i-1][j-1])%P;
```

```
fc[0]=1;
for(int i=1;i<=n;i++)fc[i]=1ll*fc[i-1]*i%P;
inv[n]=pow(fc[n],P-2);
for(int i=n-1;i>=0;i--)inv[i]=1ll*inv[i+1]*(i+1)%P;
int C(int n,int m)
{
    return 1ll*fc[n]*inv[m]%P*inv[n-m]%P;
}
```

卢卡斯定理

- 我们之前讨论的情形是n不太大$(n \le 10^7)$，并且模数p比n大且是素数。
- Lucas 定理在另一种情形下很有用：
- n, m比较大，p是比较小的素数$(p \le 10^6)$。

$$\binom{n}{m} \bmod p = \binom{\left\lfloor \frac{n}{p} \right\rfloor}{\left\lfloor \frac{m}{p} \right\rfloor} \times \binom{n \bmod p}{m \bmod p} \bmod p$$

- 预处理复杂度$O(p)$，单次查询复杂度$O(\log n)$。
- 证明详见OI Wiki
- https://oiwiki.org/math/number-theory/lucas/#%E8%AF%81%E6%98%8E

如何实现

```cpp
int Lucas(int n,int m)
{
    if(!m)return 1;
    return 1ll*C(n%P,m%P)*Lucas(n/P,m/P)%P;
}
```

P1962 斐波那契数列

- 斐波那契数列是满足下面递推式的一个数列：

$$F_n = \begin{cases} 1(n \le 2) \\ F_{n-1} + F_{n-2}(n \ge 3) \end{cases}$$

- 求出$F_n \bmod 10^9 + 7$。
- $n < 2^{63}$

矩阵加速递推

- 直接递推显然是不行的。
- 我们考虑如何用矩阵乘法来描述这个递推过程。

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \times \begin{pmatrix} F_{n-1} \\ F_{n-2} \end{pmatrix} = \begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix}$$

- 那么将向量$\begin{pmatrix} F_{n-1} \\ F_{n-2} \end{pmatrix}$也写成矩阵乘法，就是

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \times \begin{pmatrix} F_{n-2} \\ F_{n-3} \end{pmatrix} = \begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix}$$

- 由此递归下去，可以得到

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-2} \times \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix}$$

- 对矩阵进行快速幂，可以求出答案。

如何实现

```
struct matrix
{
    int a[2][2];
    matrix(){memset(a,0,sizeof(a));}
    matrix operator*(const matrix &b)const
    {
        matrix c;
        For(i,0,1)For(j,0,1)For(k,0,1)c.a[i][j]=(c.a[i][j]+1ll*a[i][k]*b.a[k][j])%P;
        return c;
    }
};
```

如何实现

```
matrix pow(matrix a,ll b)
{
    matrix ans;
    ans.a[0][0]=ans.a[1][1]=1;
    while(b)
    {
        if(b&1)ans=ans*a;
        a=a*a;
        b>>=1;
    }
    return ans;
}
```

```
int main()
{
    ll n;
    cin>>n;
    if(n==1)
    {
        cout<<1<<'\n';
        return 0;
    }
    matrix a;
    a.a[0][0]=a.a[0][1]=a.a[1][0]=1;
    a=pow(a,n-2);
    cout<<(a.a[0][0]+a.a[0][1])%P<<'\n';
}
```

[SDOI2010] 古代猪文

- 一句话题意：求 $g^x \bmod 999911659$, 其中 $x = \Sigma_{d|n}\binom{n}{d}$
- $n, g \leq 10^9$

Solution

- 不难想到，使用费马小定理，先计算 $x \bmod (999911659-1)$，然后计算快速幂。
- 如何计算上面这个结果？
- 现在问题变为下面这样：
- 计算$\sum_{d|n}\binom{n}{d} \bmod 999911658$的结果。
- n的约数可以通过$O(\sqrt{n})$方式枚举，因此我们只需要快速计算$\binom{n}{d}$取模的结果。

Solution

- 考虑到n,d都是10^9级别，直接计算组合数是不现实的。
- 让我们看看999911658这个数有什么性质。
- 把这个数质因数分解，得到$999911658 = 2 \times 3 \times 4679 \times 35617$
- 计算组合数对这四个质数取模的结果是简单的。
- 令$\binom{n}{d} = x$，上面四个质数分别为p_1, p_2, p_3, p_4，$999911658 = m$

Solution

$$\begin{cases} x \equiv a_1 \pmod{p_1} \\ x \equiv a_2 \pmod{p_2} \\ x \equiv a_3 \pmod{p_3} \\ x \equiv a_4 \pmod{p_4} \end{cases}$$

- 我们要求出$x \bmod m$的结果。
- 看看这个式子是不是和前面的EXCRT长得很像？
- 使用EXCRT解这个方程，得到$x \equiv a \pmod{m}$。这个a就是我们要求的东西。
- 代码实现留作作业。