质数那些事

关键字：欧拉、费马、素数测试、线性筛、Miller-Robin、Pollard-Rho、线性求逆、快速乘

复习 求质数

质数的定义：若一个正整数无法被除了1和它本身之外的任何自然数整除，则称该数为质数。

质数的判定：“试除法”

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

质数的筛选——Eratosthenes筛法

2， 3， ~~4~~， 5， ~~6~~， 7， ~~8~~， 9， ~~10~~， 11， ~~12~~， ……

2， 3， ~~4~~， 5， ~~6~~， 7， ~~8~~， ~~9~~， ~~10~~， 11， ~~12~~， ……

2， 3， ~~4~~， 5， ~~6~~， 7， ~~8~~， ~~9~~， ~~10~~， 11， ~~12~~， ……

2， 3， ~~4~~， 5， ~~6~~， 7， ~~8~~， ~~9~~， ~~10~~， 11， ~~12~~， ……

2， 3， ~~4~~， 5， ~~6~~， 7， ~~8~~， ~~9~~， ~~10~~， 11， ~~12~~， ……

```
void primes(int n)
{
        memset(v,0,sizeof(v));//合数标记
        for(int i=2;i<=n;i++)
        {
                if(v[i]) continue;
                cout<<i<<endl;//i是质数
                for(int j = i;j<=n/i;j++)
                {
                        v[i*j] = 1;
                }
        }
}
```

算法思想：

我们从2开始，由小到大扫描每个数x，把它的倍数2x，3x，…，[N/x]*x标记为合数。当扫描到一个数时，若它尚未被标记，则说明，2——x-1之间没有能整除x的数，该数就是质数。

复习 算数基本定理

任何一个大于1的自然数N，都可以唯一分解成有限个质数的乘积

$$N=P_1^{a_1} * P_2^{a_2} * \cdot \cdot \cdot * P_n^{a_n}$$

这里$P_1<P_2<P_n$均为质数，其指数a_i是正整数。

这样的分解称为N的标准分解式

质因数分解

结合质数判定的“试除法”和质数筛选的“Eratosthenes 筛法”，我们可以扫描2—sqrt(n)之间的每个数d，若d能整除N，则从N中除掉所有的因子d，同时，累计除去d的个数。

```
void divide(int n)
{
        m = 0;
        for(int i=2;i<=sqrt(n);i++){
                if(n%i == 0){//i能整除n,则i一定是质数
                        p[++m] = i; c[m]=0;
                        while(n%i == 0){
                                n /= i;
                                c[m]++;
                        }
                }
        }
        if(n>1){
                p[++m]=n; c[m]=1;
                for(int i=1;i<=m;i++){
                cout<<p[i]<<'^'<<c[i]<<endl;
                }
        }
}
```

补充 关于同余

同余运算

如果两个数a和b之差能被m整除，那么我们就说a和b对模数m同余

我们记作$a \equiv b \pmod{m}$

把同余当作一种运算，是因为同余满足运算的诸多性质
它满足自反性（一个数永远和自己同余）
对称性（a和b同余，b和a也就同余）
传递性（a和b同余，b和c同余可以推出a和c同余）

如果$a \equiv b \pmod{m}$，$x \equiv y \pmod{m}$，则$a \pm x \equiv b \pm y \pmod{m}$

如果$a \equiv b \pmod{m}$，$x \equiv y \pmod{m}$，则$ax \equiv by \pmod{m}$

如果$ac \equiv bc \pmod{m}$，且c和m互质，则$a \equiv b \pmod{m}$

如果$a \equiv b \pmod{m}$，且$c \neq 0$，则$ac \equiv bc \pmod{mc}$

同余运算相关证明

- **性质：如果a≡b(mod m)，x≡y(mod m)，则a+x≡b+y(mod m)。**
- 证明：条件告诉我们，可以找到p和q使得a-mp = b-mq，也存在r和s使得x-mr = y-ms。于是a-mp + x-mr = b-mq + y-ms，即a+x-m(p+r) = b+y-m(q+s)，这就告诉我们a+x和b+y除以m的余数相同。
- **如果a≡b(mod m)，x≡y(mod m)，则ax≡by(mod m)。**
- 证明：条件告诉我们，a-mp = b-mq，x-mr = y-ms。于是(a-mp)(x-mr) = (b-mq)(y-ms)，等式两边分别展开后必然是ax-m(...) = by-m(...)的形式，这就说明ax≡by(mod m)。

- 如果ac≡bc(mod m)，且c和m互质，则a≡b(mod m)（就是说同余式两边可以同时除以一个和模数互质的数）。
证明：条件告诉我们，ac-mp = bc-mq，移项可得ac-bc = mp-mq，也就是说(a-b)c = m(p-q)。这表明，(a-b)c里需要含有因子m，但c和m互质，因此只有可能是a-b被m整除，也即a≡b(mod m)。

如果a≡b (mod m)，且c≠0，则ac≡bc (mod mc)

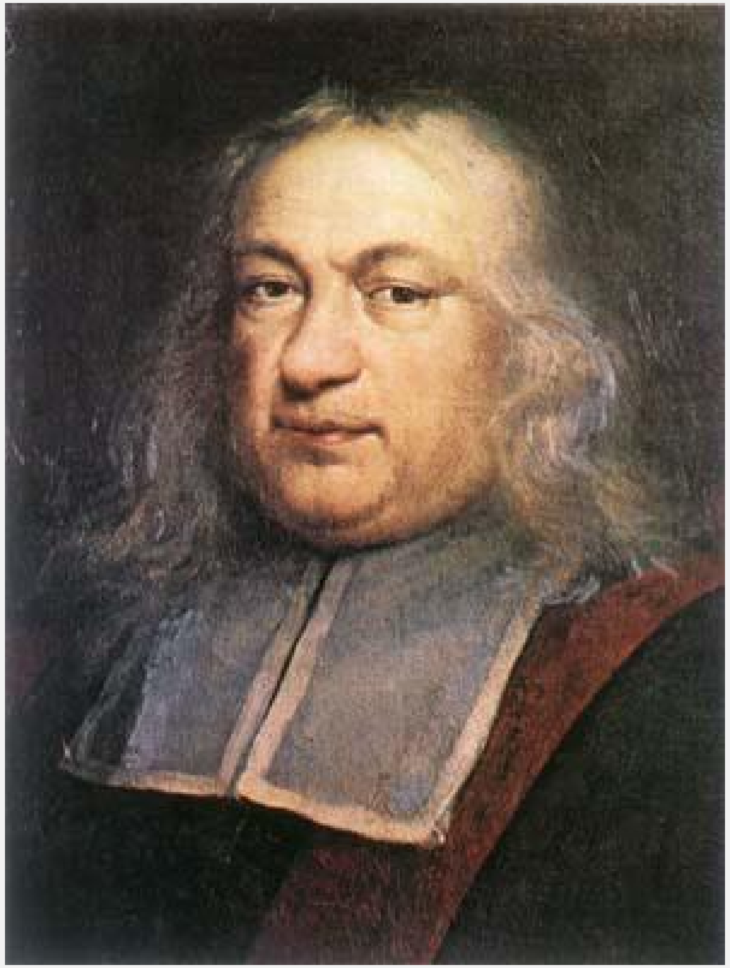
证明：m|(a-b)，则 mc|(a-b)c，证毕

费马小定理
Fermat Theory

费马小定理 Fermat Theory

如果p是素数，且a与p互质，即gcd(a,p)=1

那么$(a^{p-1}) \equiv 1 \pmod{p}$



费马小定理 证明

考虑 1a，2a，3a，...，(p-1)a 这些数

1.他们两两不同余（mod p）

2.他们均与p互质

因此可以得到 (1a)*(2a)*(3a)*...*((p-1)a)=1*2*3*...*(p-1) (mod p)

又因为 gcd(1*2*3*...*(p-1),p)=1，所以 $a^{p-1} \equiv 1 \pmod{p}$

费马小定理 应用

求乘法逆元 $(x*x')\equiv 1 \pmod p$ 称x'为x模p的乘法逆元

因为 $(a^{p-1}) \equiv 1 \pmod p$ **那么** $(a*a^{p-2}) \equiv 1 \pmod p$

所以 a^{p-2}**就是a关于模p的乘法逆元**

例：求解(a/b)mod p 其中p为素数

设b关于模p的乘法逆元为b'，我们有$(b*b')\equiv 1 \pmod p$

根据费马小定理，若b与p互质 $(b^{p-1}) \equiv 1 \pmod p$

有$(b*b^{p-2})\equiv 1 \pmod p$ 又因$(b*b')\equiv 1 \pmod p$

我们得到b的逆元$b'=b^{p-2}$,可通过快速幂求解。

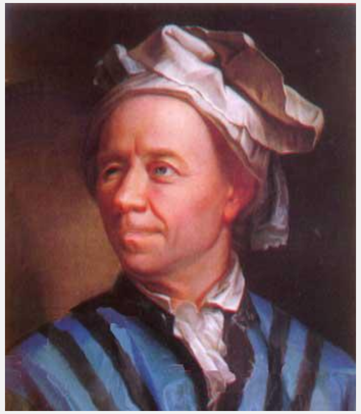
于是求(a/b)mod p 就转换为求(a*b')mod p

=((a mod p)*(b' mod p))mod p

欧拉函数

Euler's totient function

欧拉函数



对于一个正整数x，小于x且和x互质的正整数的个数，记做：φ(x)

其中φ(1)被定义为1，但是并没有任何实质的意义

通式1： $\varphi(x)=x*(1-1/p_1)*(1-1/p_2)*(1-1/p_3)*\ldots*(1-1/p_n)$

其中p_1，p_2，p_3……p_n为x的所有质因数

比如，12的质因数为2和3，那么$\varphi(12)=12*(1-1/2)*(1-1/3)=4$

通式2：若x是质数p的k次幂，即$x=p^k$，有$\varphi(x)=p^k-p^{k-1}=(p-1)*p^{k-1}$

因为除了p的倍数外，其他数都跟x互质。设$t*p=p^k$，那么$t=p^{k-1}$

比如，$27=3^3$，$\varphi(27)=27*(1-1/3)=3^3*(1-3^{-1})=3^3-3^2$

程序中我们一般将φ(x)写成phi(x)

欧拉函数的性质

性质1： 欧拉函数是积性函数

若x，y互质（即gcd(x，y)=1），那么$\varphi(x*y)=\varphi(x)*\varphi(y)$

性质2： 若x是质数$\varphi(x) = x-1$

欧拉定理

Euler Theorem

欧拉定理

若a，n为正整数，且a，n互质（即gcd(a,n)=1），则有：

$$a^{\varphi(n)} \equiv 1 \pmod n$$

求证费马小定理：如果n是素数，且a与n互质，那么$(a^{n-1}) \equiv 1 \pmod n$

证明：

1.因为n是质数，根据欧拉函数$\varphi(n)=n-1$

2.根据欧拉定理 $a^{\varphi(n)} \equiv 1 \pmod n$ 有 $a^{n-1} \equiv 1 \pmod n$ ，得证！

欧拉定理 证明

设小于n且与n互质的数为 $x_1, x_2, \ldots, x_{\varphi(n)}$

考虑这些数 a*x_1,a*x_2,...,a*x_{phi(n)}

1.他们两两不同余（mod n）

2.他们均与n互质

因此可以得到

(a*x_1)*(a*x_2)*(a*x_3)*...*(a*x_{phi(n)})=x_1*x_2*x_3*...*x_{phi(n)} (mod p)

又因为 gcd(x_1*x_2*x_3*...*x_{phi(n)},n)=1，所以 $a^{\varphi(n)} \equiv 1 \pmod{n}$

欧拉定理 应用1

计算$7^{2222222}$的个位数字。

即是求$7^{2222222} \bmod 10$

因为7与10互质，根据欧拉定理有$7^{\varphi(10)} \equiv 1 \pmod{10}$

因为$\varphi(10)=4$，我们有$7^4 \equiv 1 \pmod{10}$

$7^{2222222} \bmod 10 = ((7^4)^{555555} * 7^2) \bmod 10$

$= ((7^4 \bmod 10)^{555555} * 7^2) \bmod 10$

$= ((1)^{555555} * 7^2) \bmod 10$

$= 49 \bmod 10$

$= 9$

欧拉定理可以用来对幂的模运算进行降幂!

欧拉定理 降幂

a与n**互质**，当b很大时，可用欧拉定理降幂

$$a^b \bmod n = a^{b \bmod \varphi(n)} \bmod n$$

扩展(广义)欧拉定理

$$a^b \% c = a^{b \% \varphi(c) + \varphi(c)} \% c$$

条件$b >= \varphi(c)$ 不要求a，c互质

扩展欧拉定理 应用

计算$8^{22222222222222}$的末尾数字。

即是求$8^{22222222222222} \% 10$

因为8与10不互质，根据扩展欧拉定理有

$8^{22222222222222} \% 10 = 8^{22222222222222 \% Phi(10) + Phi(10)} \% 10$

因为Phi(10)=4

$8^{22222222222222} \% 10 = 8^{2+4} \% 10 = 4$

欧拉函数 代码实现

复习 算数基本定理

任何一个大于1的自然数N，都可以唯一分解成有限个质数的乘积

$N=P_1^{a_1} * P_2^{a_2} * \cdots * P_n^{a_n}$

这里$P_1<P_2<P_n$均为质数，其指数a_i是正整数。

这样的分解称为N的标准分解式

欧拉函数代码1：直接计算

$\varphi(x)=x*(1-1/p_1)*(1-1/p_2)*(1-1/p_3)*\ldots*(1-1/p_n)$ $p_1, p_2, p_3 \ldots p_n$为x的所有质因数

```
int euler(int x)                                 //计算φ(x)
{
     int i,ans= x, a = x;                        //找出所有x的质因数
     for(i = 2; i*i<= a; i++)                    //从2讨论到sqrt(n)即可
     {
            if(a%i == 0)                         //i为x的质因数
            {
                  ans = ans-ans/i; //φ(x) = x*(1 - 1/p1)*(1 - 1/p2)......
                  while(a%i == 0) a=a/i;         //算术基本定理，抹掉i^1,i^2,i^3....
            }
     }
     if(a > 1) ans =ans - ans/a;                 //存在大于sqrt(a)的质因子
     return ans;
}
```

NKOI3550 试试看！

时间复杂度为$O(\sqrt{n})$

复习 筛选法找质数

给出一个整数n，由小到大输出2到n之间所有的质数(n<=100000)。

```
for(i=1;i<=n;i++)a[i]=i;        //初始设每个数都是质数。a[i]=i表示i是素数，否则不是

t=int(sqrt(n)+0.5);             //只需讨论前根号n个数字
for(i=2;i<=t;i++)
    if(a[i]==i)                 //如果i是质数，那么i的倍数肯定是合数
        for(j=i+i;j<=n;j+=i)a[j]=0;    //把i的倍数全部标记为不是质数

for(i=2;i<=n;i++)
    if(a[i]==i)cout<<i<<" ";
```

欧拉函数代码2：筛选法(打表)

$\varphi(x)=x*(1-1/p_1)*(1-1/p_2)*(1-1/p_3)*\ldots*(1-1/p_n)$ p_1，p_2，p_3……p_n为x的所有质因数

```
void Selcet_Euler(int n) //求 φ(1) ,φ(2),...,φ(n)
{
        int i,j;
        for(i = 1;i <=n; i++)  phi[i]=i;
        for(i = 2;i <=n; i++)
        {
                if(phi[i] == i) //表示i未被筛到，i是素数
                {
                        for(j = i; j <=n; j += i) //然后更新含有它的数
                            phi[j] = phi[j] / i*(i - 1); // x*(1 - 1/p1)....*(1 - 1/pk).先除后乘
                 } // 此处注意先/i再*（i-1），否则范围较大时会溢出
          }
  }
```

时间复杂度O(NloglogN)

预先置所有数的欧拉函数值都为它本身，有定理可知，如果p是一个正整数且满足φ（p）=p-1;那么p是素数，在遍历过程中如果遇到欧拉函数与自身相等的情况。那么说明该数为素数，把这个数的欧拉函数值改变，同时也把能被素因子整除的数改变。

- 时间复杂度的计算

$$\sum_p \frac{n}{p} = n \sum_p \frac{1}{p} \approx n \sum_p \ln\left(1 + \frac{1}{p}\right) = n \ln\left(\prod_p \left(1 + \frac{1}{p}\right)\right)$$

$$< n \ln\left(\prod_p \left(1 + \frac{1}{p} + \frac{1}{p^2} + \cdots\right)\right) = n \ln\left(\sum_{k=1}^{\infty} \frac{1}{k}\right) \approx n \ln\ln n$$

欧拉函数 应用例题

例1：法雷数列 NKOJ3547

对任意给定的一个>=2的自然数n，将分母小于等于n的不可约的真分数按升序排列，这个序列称为n级法雷数列，以Fn表示。
也就是说数列中的任意元素a/b，有0 < a < b <= n 并且 gcd(a,b) = 1 。
下面是一些法雷数列的例子：

```
F2 = {1/2}
F3 = {1/3, 1/2, 2/3}
F4 = {1/4, 1/3, 1/2, 2/3, 3/4}
F5 = {1/5, 1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5}
```

给出一个n,你的任务是计算Fn有多少项。

$2<=N<=10^6$

例1：法雷数列 分析

观察：F5 = {1/5, 1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5}

= {1/2, 1/3, 2/3, 1/4, 3/4, 1/5, 2/5, 3/5, 4/5}

F5的项数=φ(2)+φ(3)+φ(4)+φ(5)

Fn的项数=φ(2)+φ(3)+......+φ(n)

例2：公约数求和 NKOJ3548

给你一个正整数n，求 $\sum gcd(i, n)$　$1<=i<=n$。

$1<n<=2^{31}-1$

$\sum gcd(i, n) = gcd(1, n)+gcd(2,n)+...+gcd(n,n)$

例2：公约数求和 分析

给你一个正整数n，求∑gcd(i, n)　　1<=i<=n。

ans = ∑i*φ(n/i)　(1<=i<=n 并且 i|n)。注意：b|a表示a%b==0

意思是从1到n的所有数字i，如果i是n的因数，那么计算i*φ(n/i)，加入答案中。

设1到n中有m个数字和n的最大公约数都是i，那么就需要把m*i加入答案中。问题是如何计算m的个数。

因为gcd(x,n) = i，可以得到gcd(x/i,n/i)=1，那么x/i就是和n/i互质的
有多少个这样的x/i呢，那么一共存在φ(n/i)个x/i，那么就可以推出m的个数就是φ(n/i)。

例3：GCD的个数 NKOJ3684

给定两个整数N,M, 求满足下列两个条件的X的个数：

条件1：1<=X<=N

条件2：gcd(X,N)>=M

2<=N<=1000000000, 1<=M<=N

例3：GCD的个数 分析

$ans = \Sigma\varphi(n/x)$ $(1 \le x \le n$ 并且 $x|n$ 并且 $GCD(x,n) \ge m)$

若x满足x|n 且 GCD(x,n)>=m

设y=n/x,则φ(y)表示小于y且与y互质的数的个数。

设小于y且与y互质的数有k个，分别是p1,p2,........,pk

那么有x*pi<=n ， 1<=i<=k　　　*(因为pi<y,而 y*x=n)*

同时有GCD(x*pi,n)>=m　　　*（因为GCD(x,n)>=m）*

也就是说，现在有k个数是满足要求的，而k=φ(y)

于是我们得到解法：先求n的所有大于等于m的因数Xi

$ans = \Sigma\varphi(n/X_i)$

质数筛

质数的普通筛法(Eratosthenes筛法)

```
const int MAXN = 100000;                                    //筛100000以内
的质数
void getPrime()
{
    int i, j,tot=0;
    for (i=1; i<=MAXN; i++) mark[i] = true;              //一开始假设所有数
都是质数
    mark[0] = mark[1] = false;                            //0和1不是质
数
    for (i=2; i<=MAXN; i++)
    {
        if (mark[i]==false) continue;                      //i不是质数,
跳过
        Prime[++tot]=i;
        //if( i*i > MAXN ) continue;
        for (j=i*2; j<=MAXN; j+=i) mark[ j ] = false;   //i是质数，那么把i的倍数
```

质数的线性筛法(Euler筛法)

```
const int MaxN = 100000;                                    //筛100
void getPrime()
{
        int i,j,Cnt=0;
        for(i=2;i<=MaxN;i++)
       {
                if(Mark[i]==false)Prime[++Cnt]=i;
                for(j=1;  j<=Cnt && i*Prime[j]<=MaxN;  j++)
               {
                        Mark[i*Prime[j]]=true;  //Prime[j]是合数i*Prime[j]的最小质因数
                        if(i%Prime[j]==0) break;
                        /* prime数组中的素数是递增的,当 i 能整除 prime[j],
                        那么 i*prime[j+1] 这个合数肯定会被 prime[j] 乘以某
                        个数筛掉。因为i中含有prime[j], prime[j] 比
                        prime[j+1] 小。接下去的素数同理。所以不用筛下去了。
                        */
               }
       }
}//时间复杂度O(n)
```

每个合数必有一个最小质因数，每个合数仅被它的最小质因数筛去正好一次。

原理:

1. 任何一个合数都可以表示成一个质数和一个整数的乘积
2. 假设A是一个合数,

   且A = x * y; (假设y是质数, x合数)

   x = a * b; (假设a是质数, 且a < x)

   有 A = a * b * y = a * Z (Z = b * y)

   即合数(x)与质数(y)的乘积可以表示成一个更大的合数(Z)与一个更小的质数(a)的乘积。

如果 i 是素数的话，一个大的素数 i 乘以不大于 i 的素数，这样筛除的数跟之前的是不会重复的。筛出的数都是 N=p1*p2的形式, p1，p2之间不相等

例如: 如果i = 8; 那么由于i%2 == 0; 因此筛8就只需要讨论质数2即可，因为对于大于2质数，像3，有:

8*3 = 2*4*3 = 12*2　也就是说24(8*3=24)并不需要在讨论8时去筛，在讨论12时才筛到。

线性筛求欧拉函数

先看看欧拉函数的性质（ p为质数 ）：

1. phi(p)=p-1
2. 如果i mod p = 0, 那么 phi(i * p)=p * phi(i)
3. 若i mod p ≠0, 那么 phi(i * p)=phi(i) * (p-1)

证明：

性质2

根据欧拉函数的定义有：

phi(p*i)=(p*i)*(1-1/p1)*(1-1/p2)*...*(1-1/pk)

=p*(i*(1-1/p1)*(1-1/p2)*...*(1-1/pk)))

=p*phi(i)

p1,p2,...,pk 是i的质因数，那么p一定在其中。

性质2得证。

若i mod p ≠0，p又是质数，说明i 与 p 互质。

根据欧拉函数的积性特征：若a与b互质，则phi(a*b)=phi(a)*phi(b)

我们有：phi(i*p)=phi(i)*phi(p)

又因为p是质数，phi(p)=p-1

所以有phi(i*p)=phi(i)*(p-1),得证

线性筛求欧拉函数

利用欧拉函数的性质（ p为质数 ）：

1. phi(p)=p-1
2. 如果i mod p = 0, 那么 phi(i * p)=p * phi(i)
3. 若i mod p ≠0, 那么 phi(i * p)=phi(i) * (p-1)

```
void getPhi() //求 φ(1) ,φ(2),...,φ(N)
{
        int i,j,tot=0;
        for(i=2;i<=N;i++)
              {
                if(Mark[i]==false){  Prime[++tot]=i;  phi[i]=i-1;  }//当 i 是素数时 phi[i]=i-1
                for(j=1;  j<=tot && i*Prime[j]<=N;  j++)
                        {
                        Mark[i*Prime[j]]=true;
                        if(i%Prime[j]==0)
                                {
                                phi[i*prime[j]]=phi[i]*Prime[j];   break;
                        }                                        //如果i mod p = 0, 那么 phi[i * p]=p * phi[i]
                        else   phi[i*Prime[j]]=phi[i]*(Prime[j]-1);
                }                                    //其实这里Prime[j]-1就是phi[Prime[j]], 利用了欧拉函数的积性
        }
} //时间复杂度O(n)
```

Miller-Rabin

素数测试

它是素数吗？ 18446744073709551613

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

对于一个整数n，朴素的素数判定方法是枚举2到√n之间的数字，讨论n能否整数它。时间复杂度为√n 。如果n是一个大整数，这种方法就可能会超时了。

回忆费马小定理：

如果p是素数，且a与p互质，即gcd(a,b)=1

那么 $(a^{p-1}) \equiv 1 \pmod p$

下面说法是否成立？

如果a与p互质，并且 $(a^{p-1}) \equiv 1 \pmod p$

那么p是素数

反例：$2^{340} \% 341 = 1$ 但341不是质数，341=11*31

二次探测定理

如果p是素数，x是小于p的正整数，且 $x^2 \equiv 1(mod\, p)$
那么要么x=1，要么x=p-1。

因为x^2 mod p = 1 相当于p能整除x^2 - 1，也即p能整除(x+1)(x-1)。
由于p是素数且x<p，那么只可能是x-1能被p整除(此时x=1)或x+1能被p整除(此时 x=p-1)。

我们可以用二次探测定理来进行素数测试，我们以2^{340} % 341 = 1 为例
2^{340} % 341 = $(2^{170})^2$ % 341 = 1 根据二次探测定理，如果341真的是素数的话：
2^{170} % 341 结果只可能是1或者340， 经计算确实是2^{170} % 341=1
继续讨论 2^{170} % 341 = $(2^{85})^2$ % 341 = 1
即2^{85} % 341 结果只可能是1或340，经计算得到2^{85} % 341 = 32 与定理相矛盾，说明341不是素数。

上面就是Miller-Rabin的素数测试的方法！

Miller-Rabin素数测试

如果p是素数，x是小于p的正整数，且 $x^2 \equiv 1(mod\, p)$ 那么要么x=1，要么x=p-1。

我们要测试整数n是否为素数。若n是素数，根据费马定理，有 a^{n-1} % n = 1 我们任选一个数做底数a

我们令 $n-1=d*2^R$ 其中d是一个奇数，于是 a^{n-1} % n = $(a^d)^{2^R}$ % n = $a^{d*2^{R-1}}$ $*a^{d*2^{R-1}}$ % n

根据二次探测定理， $a^{d*2^{R-1}}$ % n 要么等于1 要么等于 n-1 才行。

如果等于 $a^{d*2^{R-1}}$ % n 既不等于1也不等于n-1，n一定**不是素数**，结束讨论。

如果等于 $a^{d*2^{R-1}}$ % n = n-1，我们认为n**是素数**，结束讨论。

如果等于 $a^{d*2^{R-1}}$ % n = 1，继续讨论：

根据二次探测定理， $a^{d*2^{R-2}}$ % n 要么等于1 要么等于 n-1。……

上述讨论过程中，如果存在某个整数k，是的 a^{d*2^k} % n = n-1 （0<=k<R）我们就认为n是素数，结束讨论。

或者，若 a^d % n = 1，n是素数

这是一种不确定的算法，我们随机选一个底数a进行上述测试，准确率约为75% 。

我们可以找几个随机数进行上述测试，若都满足，我们可以放心认为n是素数，准确率极高。

一次测试的时间复杂度约为O($\log_3 n$)，若进行了m次测试，复杂度为O($m\log_3 n$)

Miller-Rabin素数测试

算法的过程：
对于要判断的数n
1.先判断是不是2，是的话就返回true。
2.判断是不是小于2的，或合数，是的话就返回false。
3.令$n-1=d*2^R$，求出d，R，其中d是奇数。
4.随机取一个a,且1<a<n
根据费马小定理，如果$a^{n-1}\equiv 1 \pmod p$那么n就极有可能是素数，如果等式不成立，那肯定不是素数了
5.因为$n-1=d*2^R$，所以$a^{n-1}=a^{d*2\wedge R}=(a^d)^{2\wedge R}$ 所以我们令$x=a^d$ % n
6.然后是R次循环，每次循环都让y=(x*x)%n，x=y,这样R次循环之后$x=a^{(d*2\wedge R)}=a^{n-1}$了
7.因为循环的时候y=(x*x)%n，且x肯定是小于n的，正好可以用二次探测定理，
如果x^2 % n==1，也就是y等于1的时候，假如n是素数，那么x==1||x==n-1，如果x!=1&&x!=n-1，那么n肯定不是素数了，返回false。
8.运行到这里的时候$x=a^{n-1}$，根据费马小定理，x!=1的话，肯定不是素数了，返回false
9.因为Miller-Rabin得到的结果的正确率为 75%，所以要多次循环步骤4~8来提高正确率
10.循环多次之后还没返回，那么n很大可能是素数了，返回true

Miller-Rabin素数测试

把n-1表示成$d*2^R$，尽可能提取因子2，

如果n是一个素数，那么或者$a^d \bmod n=1$，

或者存在某个k使得$a^{d*2^k} \bmod n=n-1$ （ $0<=k<R$ ）

Miller- Rabin算法的代码很简单：

计算d和R的值，

然后用快速幂计算$a^d \bmod n$的值，

最后把它平方R次。

```
bool Miller_Rabin(long long n)          //测试n是否是素数
{
      long long a,x,y,d,R=0;
      if(n==2)return true;
      if(n<2||n%2==0)return false;
      d=n-1;
      while(d%2==0){ d=d/2; R++; }       //去掉因子2
      for(int i=1;i<=10;i++)   //进行10次测试，测试次数越多越准确
      {
          a=rand()%(n-2)+2;           //随机产生[2,N-1]以内的底数a
          x=KSM(a,d,n);               //二分快速幂计算a^d % n
          for(int j=1; j<=R; j++)
          {
              y = x * x % n;
              if(y==1 && x!=1 && x!=n-1)return false;
              x=y;
          }
          if(x!=1)return false;
      }
      return true;
  }
```

玄学

- $n \leq 2^{32}$时，只需检测$a = 2,7,61$，即可确保正确，即$c = 3$；
- $n \leq 2^{64}$时，只需检测$a = 2,3,5,7,11$，即可确保正确，即$c = 5$。
- n特别大时，随机选一个a进行测试，合数通过测试的概率低于$\frac{1}{4}$，合数通过c轮测试的概率低于$\frac{1}{4^c}$，适当设定测试次数即可。

线性求逆元

线性求逆元

p为质数，a*x≡1(mod p)，则称x为a的逆元，记为a^{-1}

求出1-n中所有数的逆元(模p的逆元，n<=10,000,000)。也就是求1^{-1}，2^{-1}，3^{-1},……, n^{-1}

可用扩欧或者费马小定理来求，求n个数的逆元时间复杂度为O(nlogn)，太耗时。下面介绍线性求法：

$$1*1\equiv 1 \pmod p \Rightarrow 1^{-1}\equiv 1 \pmod p$$ //1的逆元就是1

$$a*a^{-1}\equiv 1 \pmod p \quad 1<a<p$$

令$k=\lfloor \frac{p}{a} \rfloor$，$r=p\ mod\ a$　　那么$p=k*a+r \quad 0<r<a$

则有$k*a+r\equiv 0 \pmod p$

两边同时乘上 $a^{-1}*r^{-1}$ 有$(k*a+r)*a^{-1}*r^{-1}\equiv 0 \pmod p$

那么 $k*r^{-1}+a^{-1}\equiv 0 \pmod p$

即 $a^{-1}\equiv -k*r^{-1} \pmod p$

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

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

我们把得到的递推式写成代码

B[i]=(p-p/i)*B[p%i]%P;

B[i]表示i的逆元

当求第i项时，B[p%i]项已经算好了。

线性求逆元 参考代码

```
inv[0]=inv[1]=1;
for(int i=2;i<=n;i++)
    inv[i]=(p-p/i)*inv[p%i]%p;
```

时间复杂度O(n)

快速（慢速）乘法

Miller-Rabin素数测试

k

当x很大时，很可能会溢出。
但结果的范围又在long long 以内。
用高精度乘法又太麻烦。
这时我们需要快速乘法。

```
bool Miller_Rabin(long long n)          //测试n是否是素数
{
      long long a,x,y,d,R=0;
      if(n==2)return true;
      if(n<2||n%2==0)return false;
      d=n-1;
      while(d%2==0){ d=d/2; R++; }       //去掉因子2
      for(int i=1;i<=10;i++)   //进行10次测试，测试次数越多越准确
      {
          a=rand()%(n-2)+2;           //随机产生[2,N-1]以内的底数a
          x=KSM(a,d,n);               //二分快速幂计算a^d % n
          for(int j=1; j<=R; j++)
          {
              y = x * x % n;
              if(y==1 && x!=1 && x!=n-1)return false;
              x=y;
          }
          if(x!=1)return false;
      }
      return true;
  }
```

快速乘法

```
long long quick_mul(long long a, long long b, long long mod) //快速计算 (a*b) % mod
{
    long long ans=0;                         // 初始化
    while(b)                                 //根据b的每一位看加不加当前a
    {
        if(b&1)                              //如果当前位为1
        {
            ans=(ans+a)%mod;                  //ans+=a
        }
        b=b>>1;                               //b向前移位
        a=(a<<1)%mod;                         //更新a
    }
    return ans;
}

int main()
{
        long long a,b,c;
        cin>>a>>b>>c;
        cout<<quick_mul(a,b,c)<<endl;
}
```

例如：

$3*19=3*(10011)=(3*2^4+3*2^1+3*2^0)$

Pollard-Rho

Pollard Rho算法：

- 先用Miller-Rabin算法确定n是合数；
- 随机整数x_0和a，生成一个序列$x_{i+1} = \left(x_i^2 + a\right) \bmod n$；
- 对$i = 1,2,3 \ldots$分别计算$p = \gcd(|x_{2i} - x_i|, n)$，当$x_{2i} = x_i$时停止并重新随机生成$x_0$和$a$；
- 如果$p \neq 1$则找到了n的一个约数p，递归处理p和$\frac{n}{p}$。
- 时间复杂度$O\left(n^{\frac{1}{4}}\right)$，空间复杂度$O(\log n)$。
- 时间复杂度的计算：
- 设$n = pq, p \leq q$，序列$y_i = x_i \bmod p$；
- 根据生日悖论，$\{y_i\}$周期期望长度$O\left(\sqrt{p}\right)$，$\{x_i\}$周期期望长度$O(\sqrt{n})$，不妨假设$\{y_i\}$周期短；
- 设$\{y_i\}$周期为t，则x_{2t}与x_t还没进入周期，计算$\gcd(|x_{2t} - x_t|, n)$可以得到p。

- 此处不细讲，具体情况自行百度

课后习题

NKOJ 3550，3547，3548，3549，3684，3685

HDU 2138

思维训练：NKOJ 3683

奋斗吧
少年

巨大的成功需要付出巨大的代价

no sacrifice, no success