****1.1素数筛****

bool isprime[1000000] = {0};

long long prime[100000] = {0};

long long sieve(long long n)//筛[1,n]以内的素数，返回nprime为筛出的素数的个数。isprime[i]表示i是否是素数，prime[i]为素数表2，3，5...

{

/\*

如果不需要素数表，只用查询isprime可以只筛到sqrt(n)。

long long m = (long long) sqrt(n+0.5);

for(i=2;i<=m;i++)

注意这个时候返回的值不是素数的个数。

\*/

long long i,j;

long long m = (long long) sqrt(n+0.5);

long long nprime = 0;

memset(isprime,1,sizeof(isprime));

isprime[1]=0;

for(i = 2;i<=m;i++)

{

if(isprime[i])

{

prime[++nprime]=i;

for(j=i\*i;j<=n;j+=i)

isprime[j]=0;

}

}

return nprime;

}

****1.2 GCD与LCM****

long long gcd(long long a,long long b)

{

return b==0?a:gcd(b,a%b);

}

long long lcm(long long a,long long b)

{

return a/gcd(a,b)\*b;

}

****1.3.1整数方程ax+by=1****

//求同余方程f(x,y)=ax+by=1的一个一组解，满足abs(x)+abs(y)最小，d为gcd(a,b);

void gcd(long long a,long long b,long long&d ,long long& x,long long& y)

{

if(!b){d=a;x=1;y=0;}

else{gcd(b,a%b,d,y,x);y-=x\*(a/b);}

}

****1.3.2模n剩余类下的乘法逆元****

//一般地，也可以通过遍历[1,n-1]来求逆元

long long inversion(long long a,long long n)

{

long long d,x,y;

gcd(a,n,d,x,y);

return d==1?(x+n)%n:-1;

}

****1.3.3 中国剩余定理（x=a[i] mod m[i]）****

long long china(long long n, long long\* a,long long\* m)

{

int ifhave = 1;

long long a0,b,c,d,x0,y0;

long long a1= m[0];

long long r1= a[0];

for(int i =1;i<=n-1;i++)

{

a0=a1;b=m[i];c=a[i]-r1;

gcd(a0,b,d,x0,y0);

if(c%d!=0)

{

ifhave=0;

}

long long t = b/d;

x0 = (x0 \*(c/d)%t+t)%t;

r1=a1\*x0+r1;

a1=a1\*(m[i]/d);

}

if(!ifhave)r1=-1;

return r1;

}

//返回x==a[i] mod m[i]的解，若无解返回-1

**\*1.3.4 Pell方程**

long long pell(long long d,long long &x,long long& y)

{

long long m =(long long )sqrt(d);

if(m\*m==d)return -1;

y =1;

while(1)

{

x = (long long)sqrt(d\*y\*y+1);

if(x\*x - d\*y\*y ==1)

break;

y++;

}

return 1;

}

/\*

求解x^2-d\*y^2=1的最小正整数解。注意到d为平方数时无解，返回-1。

方程所有正整数解可以用矩阵快速幂得到。

QQ截图20160515102757.png

\*/

****1.4.1 Euler函数****

long long euler(long long n )

{

long long m = (long long )sqrt(n+0.5);

long long ans = n;

for(long long i =2;i<=m;i++)

if(n%i==0)

{

ans = ans / i \* (i-1);

while(n% i ==0) n/=i;

}

if(n>1) ans = ans / n \* (n-1);//处理到最后一步，如果n没有被除掉，则n必定是素数。

return ans;

}

****1.4.2 Euler函数表****

long long phi[10000];

void eulertable(long long n)

{

for(int i=2;i<=n;i++) phi[i]=0;

phi[1]=1;

for(int i =2;i<=n;i++)

if(!phi[i])

for(int j = i;j<=n;j+=i)

{

if(!phi[j])phi[j]=j;

phi[j]=phi[j] / i\*(i-1);

}

}

****1.5 快速幂****

long long fast (long long a,long long x)

{

if(x==0)return 1;

long long y = fast(a,x/2);

y=(y\*y);

if(x%2==1)

y=y\*a;

return y;

}

****1.6 矩阵类****

const int N = 2;

struct mat{long long a[N+1][N+1];};

mat operator + (mat a,mat b)

{

mat c;

memset(c.a,0,sizeof(c.a));

for(int i=1;i<=N;i++)

for(int j =1;j<=N;j++)

c.a[i][j]=a.a[i][j]+b.a[i][j];

return c;

}

mat operator \* (mat a,mat b)

{

mat c;

memset(c.a,0,sizeof(c.a));

int i,j,k;

for(int i =1;i<=N;i++)

for(int j =1;j<=N;j++)

for(int k =1;k<=N;k++)

c.a[i][j]+=a.a[i][k]\*b.a[k][j];

return c;

}

mat matinit(void)

{

mat c;

for(int i =1;i<=N;i++)

for(int j =1;j<=N;j++)

{

c.a[i][j]=0;

if(i==j)c.a[i][j]=1;

}

return c;

}

mat fastmat(mat a,long long k)

{

mat c = matinit();

while(k>0)

{

if(k%2==1) c = c\*a;

k=k/2;

a=a\*a;

}

return c;

}

****1.7 Miller-Rabin大数判素****

//基于伪素判别方法：若n是素数，则对任意（a，n）=1，成立a^n-1==1 mod n

LL multi(LL a,LL b,LL m)//快速乘

{

LL ans=0;

while(b)

{

if(b&1)

{

ans=(ans+a)%m;

b--;

}

b>>=1;

a=(a+a)%m;

}

return ans;

}

LL quick\_mod(LL a,LL b,LL m)//快速mod

{

LL ans=1;

a%=m;

while(b)

{

if(b&1)

{

ans=multi(ans,a,m);

b--;

}

b>>=1;

a=multi(a,a,m);

}

return ans;

}

bool Miller\_Rabin(LL n)

{

if(n==2) return true;

if(n<2||!(n&1)) return false;

LL a,m=n-1,x,y;

int k=0;

while((m&1)==0)

{

k++;

m>>=1;

}

int Times = 10;

for(int i=0;i<Times;i++)

{

a=rand()%(n-1)+1;//采取[1,n-1]的数为底

x=quick\_mod(a,m,n);

for(int j=0;j<k;j++)//二次试探，避免伪素数

{

y=multi(x,x,n);

if(y==1&&x!=1&&x!=n-1) return false;

x=y;

}

if(y!=1) return false;

}

return true;

}

//在main函数中加入srand(time(null)); 初始化随机数种子

****1.8容斥原理（DFS）****

//例：求小于等于m的与n互素的个数

// p[0]为素因子的个数，p[i]存放第i个素因子。e.g. n=6,p={3,2,2,3}

//depth为已走过的素因子个数，mul为当前的乘积，op表示当前取值的奇偶性，ans计数

void IncludeExclude(int depth, LL m, LL mul, int op, int\* p, LL &ans) {  
        if(m < mul) return ;   
        if(depth == p[0]) {   
            ans += (op ? -1 : 1) \* (m / mul);  
            return ;  
        }  
        for(int i = 0; i < 2; i++) {   
            // 0 表示不取, 1表示取  
            IncludeExclude( depth+1, m, mul \* (i?p[depth+1]:1), op^i, p, ans );//这里的^是异或  
        }  
    }

//调用参数为(0,m,1,1,p,ans)