I —— 解题报告

题意:给出一个正整数 N(2 <= N < 2^{54}),判断 N 是不是素数,若不是素数,求出其最小的素因子.

思路:模板题.首先用 Miller-Rabin 进行素性测试,若不是素数,则使用 Pollard-rho 方法不断对 N 进行分解,求出最小素因子.

注意:java 的 BigInteger 中的 isProbablePrime()方法就是 Miller-Rabin,但是使用该方法提交无论如何都是 RE..而同样的代码,用 G++提交也会 RE,C++却不会..实在不能理解 POJ 的编译器…

```
#include <cstdio>
#include <cstdlib>
#include <ctime>
#include <cmath>
#include <algorithm>

using namespace std;

typedef __int64 LL;

LL big_rand(LL m)
{
    LL x=rand();
    x*=rand();
    if(x<0) x=-x;
    return x%m;
}
LL mod_mul(LL x,LL y,LL n)
{
    if(!x || !y)   return 0;
    return (((x&1)*y)%n+(mod_mul(x>>1,y,n)<<1)%n)%n;
}
LL mod_exp(LL x,LL y,LL n)
{
    LL res=1;
    while(y)
    {
        if(y&1) res=mod_mul(res,x,n);
        x=mod_mul(x,x,n);
        y>>=1;
    }
    return res;
}
bool Miller_Rabbin(LL n)
{
```

```
    LL i,j,x,m,k;
    if(n==2)        return 1;
    if(n<2 || !(n&1))       return false;
    m=n-1;k=0;
    while(!(m&1))   m>>=1,k++;
    for(i=0;i<4;i++)
    {
        x=big_rand(n-2)+2;
        x=mod_exp(x,m,n);
        if(x==1)        continue;
        for(j=0;j<k;j++)
        {
            if(x==n-1)      break;
            x=mod_mul(x,x,n);
        }
        if(j>=k)        return false;
    }
    return true;
}
LL gcd(LL x,LL y)
{
    if(x>y) swap(x,y);
    while(x)
    {
        LL t=y%x;
        y=x;
        x=t;
    }
    return y;
}
LL func(LL x,LL m)
{
    return (mod_mul(x,x,m)+1)%m;
}
LL Pollard_rho(LL n)
{
    if(Miller_Rabbin(n))    return n;
    if(!(n&1))      return 2;
    LL i,x,y,res;
    i=1;
    while(true)
    {
        x=i++;
        y=func(x,n);
```

```
            res=gcd(y-x,n);
            while(res==1)
            {
                x=func(x,n);
                y=func(func(y,n),n);
                res=gcd((y-x+n)%n,n)%n;
            }
            if(res>0 && res<n)     return res;
        }
}
LL factor[100],nfac,minfac;
void cal_factor(LL n)
{
        LL x=Pollard_rho(n);
        if(x==n)
        {
            minfac=min(minfac,x);
            return;
        }
        cal_factor(x);
        cal_factor(n/x);
}
const LL lim=100000;
int main(void)
{
        LL n,t;
        srand((unsigned)time(NULL));
        scanf("%I64d",&t);
        while(t--)
        {
            scanf("%I64d",&n);
            if(Miller_Rabbin(n))    puts("Prime");
            else
            {
                if(!(n&1))      puts("2");
                else
                {
                    for(minfac=3;minfac<lim && n%minfac;minfac+=2);
                    if(minfac>=lim)
                    {
                        LL rn=sqrt(1.0*n);
                        if(rn*rn == n)
                        {
                            minfac=rn;
```

```
                    cal_factor(rn);
                }
                else
                {
                    minfac=n;
                    cal_factor(n);
                }
            }
            printf("%I64d\n",minfac);
        }
    }
    return 0;
}
```