二次剩余

一个数 a 如果不是 p 的倍数且模 p 同余于某个数的平方，则称 a 为模 p 的 **二次剩余**。而一个不是 p 的倍数的数 b 不同余于任何数的平方，则称 b 为模 p 的 **二次非剩余**。

对二次剩余求解，也就是对常数 a 解下面的这个方程： $$ x^2\equiv a\pmod p $$ 通俗一些，可以认为是求模意义下的开方。这里只讨论 **p 为奇素数** 的求解方法，将会使用 Cipolla 算法。

解的数量

对于 $x^2\equiv n\pmod p$ 能满足 “n 是模 p 的二次剩余 ” 的 n 一共有 $\frac{p-1}{2}$ 个（0 不包括在内），二次非剩余有 $\frac{p-1}{2}$ 个。

对于给定的 n 和 p 若 n 是 p 的二次剩余，则关于 x 的方程 $x^2\equiv n\pmod p$ 有且仅有两个解 $\pm x_0$ 满足 $x_0^2\equiv n\pmod p$。

勒让德符号

$$ \left(\frac n p\right)=\begin{cases}1,\,&p\nmid n\text{且}n\text{是}p\text{的二次剩余}\\ -1,\,&p\nmid n \text{且}n\text{不是}p\text{的二次剩余}\\ 0,\,&p\mid n\end{cases} $$

欧拉判别准则

$$ \left(\frac n p\right)\equiv n^{\frac{p-1}{2}}\pmod p $$

n 是二次剩余，当且仅当 $n^{\frac{p-1}{2}}\equiv 1\pmod p$。

n 是二次非剩余，当且仅当 $n^{\frac{p-1}{2}}\equiv -1\pmod p$。

Cipolla 算法

找到一个数 a 满足 a^2-n 是 **二次非剩余**，至于为什么要找满足二次非剩余的数，在下文会给出解释。这里通过生成随机数再检验的方法来实现，由于二次非剩余的数量为 $\frac{p-1}{2}$ 接近 $\frac p 2$，所以期望约 2 次就可以找到这个数。

建立一个 “ 复数域 ”，并不是实际意义上的复数域，而是根据复数域的概念建立的一个类似的域。在复数中 $i^2=-1$ 这里定义 $i^2=a^2-n$ ，于是就可以将所有的数表达为 $A+Bi$ 的形式，这里的 A 和 B 都是模意义下的数，类似复数中的实部和虚部。

在有了 i 和 a 后可以直接得到答案： $x^2\equiv n\pmod p$ 的解为 $(a+i)^{\frac{p+1}2}$。

参考实现：

```
#include <bits/stdc++.h>
```

```cpp
using namespace std;

typedef long long ll;
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) {
  n %= p;
  if (p == 2) return n;
  if (binpow_real(n, (p - 1) / 2, p) == p - 1) return -1;
  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);
```

```
}
```

例题

{https://www.luogu.com.cn/problem/P5491|【模板】二次剩余}

题意：求解方程 $x^2\equiv N\pmod p$

题解：模板题

代码：

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
ll w;
struct num{
    ll x,y;
};
num mul(num a,num b,ll p){
    num ans={0,0};
    ans.x=((a.x*b.x%p+a.y*b.y%p*w%p)%p+p)%p;
    ans.y=((a.y*b.x%p+a.x*b.y%p)%p+p)%p;
    return ans;
}
ll binpow_real(ll a,ll b,ll p){
    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){
    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){
    n%=p;
    if(p==2) return n;
    if(binpow_real(n,(p-1)/2,p)==p-1) return -1;
    ll a;
    while(1){
```

```
        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);
}
int main(){
    srand(time(0));
    int t;
    scanf("%d",&t);
    while(t--){
        ll n,p;
        scanf("%lld %lld",&n,&p);
        if(!n){
            puts("0");continue;
        }
        ll ans1=cipolla(n,p),ans2;
        if(ans1==-1){
            puts("Hola!");continue;
        }
        ans2=p-ans1;
        if(ans1==ans2) printf("%lld\n",ans1);
        else if(ans1<ans2) printf("%lld %lld\n",ans1,ans2);
        else printf("%lld %lld\n",ans2,ans1);
    }
    return 0;
}
```

参考链接

OI Wiki