0x70 二次剩余

0x71 二次剩余与二次非剩余

- **定义**

对于二次同余方程 $x^2 \equiv n \pmod p$ 有解，则称 n 为 p 的二次剩余，x 为该二次同余方程的解。

二次剩余 n，就是一个二次项 $\%p$ 后的剩余。

n 当对任意 x，$x^2 \equiv n \pmod p$ 不成立时，称 d 是模 p 的二次非剩余。

- **应用**

求 $\sqrt{n}\%p$，若 n 为 p 的二次剩余，则 $\sqrt{n}\%p = x\%p$。

即：若该二次同余方程有解，则 n 可以在模 p 的意义下开根。

0x72 Cipolla 算法解算法二次同余方程

需要保证模 p 是**奇素数**。

引入勒让德符号：$(\dfrac{n}{p})$

表示 n 是否为 p 的二次剩余，1 和-1 表示是与否，0 表示 $n=0$的情况。

- **定理**

定理72.1： $(\frac{n}{p} = n^{\frac{P-1}{2}} \pmod p)$

定理72.2： 若找到一个 a 使得 $a^2-n=w$，且$(\dfrac{w}{p})=-1$则$x=(n+\sqrt{w})^{\frac{p+1}{2}}$ 为 $x^2\equiv n(\mod p)$的解

定理72.3： 对于二次同余方程 $x^2 \equiv n(\mod p)$有 $\frac{p-1}{2}+1$个 n 使此方程有解

- **实现**

> 求解方程 $x^2 \equiv N \pmod p$。保证 p 是奇素数。输入一个 T 代表数据组数，接下来 T 行，每行一个 N 和一个 p。对于每一行输出：若有解，则按 $\mod p$ 后递增的顺序输出在 $\mod p$ 意义下的全部解. 若两解相同，只输出其中一个 若无解，则输出 `No Solution`

有了上面的三个定理，现在我们的唯一问题就是如何找到合适的 a 使得 ω 满足条件，我们考虑**随机**，由**定理72.3**可知，a^2-n 在模p意义下为非二次剩余的概率为 $\frac{p-1}{2p}$，那么这样随机的期望次数就接近于 2，可以看做是 $O(1)$ 求。因为在整个过程中使用过快速幂，所以时间复杂度为 $O(logp)$。

```cpp
int mod;
ll I_mul_I; // 虚数单位的平方
struct Complex {//建议自己实现复数
    ll real, imag;
    Complex(ll real = 0, ll imag = 0): real(real), imag(imag) { }
```

```
};
inline bool operator == (Complex x, Complex y) {
    return x.real == y.real and x.imag == y.imag;
}
inline Complex operator * (Complex x, Complex y) {
    return Complex((x.real * y.real + I_mul_I * x.imag % mod * y.imag) % mod,
            (x.imag * y.real + x.real * y.imag) % mod);
}
Complex qpow(Complex x, int k) {
    Complex res = 1;
    while(k) {
        if(k & 1) res = res * x;
        x = x * x;
        k >>= 1;
    }
    return res;
}
bool check_if_residue(int x) {
    return qpow(x, (mod - 1) >> 1) == 1;
}
int solve(int n, int p) {
    n %= p, mod = p;
    if(p == 2) return n;
    ll a = rand() % mod;
    if(qpow(n,(mod - 1) / 2) == p - 1) return -1;//不存在
    while(!a || check_if_residue((a * a + mod - n) % mod))
        a = rand() % mod;
    I_mul_I = (a * a + mod - n) % mod;

    return int(qpow(Complex(a, 1), (mod + 1) >> 1).real);
}
int n, m, p, t;
int main()
{
    //srand(time(0));
    scanf("%d", &t);
    while(t -- ) {
        scanf("%d%d", &n, &p);
        int ans1 = 0, ans2 = 0;
        if(n == 0) {
            puts("0");
            continue;
        }
        ans1 = solve(n, p);
        if(ans1 == -1) puts("No Solution");
        else {
            ans2 = p - ans1;
```

```
            if(ans1 > ans2) swap(ans1, ans2);
            if(ans1 == ans2) printf("%d\n", ans1);
            else printf("%d %d\n", ans1, ans2);
        }
    }
    return 0;
}
```

0x80 某些非线性丢番图方程

0x81 毕达哥斯拉三元组（勾股数）

满足 $x^2+y^2=z^2$ 的 (x,y,z) 三元组称为毕达哥拉斯三元组，当 $gcd(x,y,z)=1$ 时，称其为本原的毕达哥斯拉三元组。

> 毕达哥拉斯三元组，也称为勾股数。

- **基本性质定理**

引理81.1： 如果 x,y,z 为一个本原毕达哥拉斯三元组，则 $gcd(x,y)=gcd(y,z)=gcd(z,x)=1$。

引理81.2： 设 x,y,z 为一个本原毕达哥拉斯三元组，则 x 为偶数且 y 为奇数或者 x 为奇数， y 为偶数。

引理81.3： 若 r,s 和 t 为正整数，且 $gcd(r,s)=1$，$rs=t^2$ ，则存在整数 m,n ，使得 $r=m^2$，$s=n^2$。（两个平方数的乘积还是一个平方数）

由此，我们可以证明得到所有的毕达哥拉斯三元组的解了。

定理81.4： 由正整数 x、y、z 构成的三元组 (x,y,z) ，其中 y 为偶数，那么由他们构成的本原的毕达哥拉斯三

元组当且仅当存在

互素的**一奇一偶**的正整数 m、n，且 $m>n$，满足

$$\begin{cases}x=m^2-n^2\\ y=2mn\\ z=m^2+n^2\end{cases}$$

我们可以看出，本原的毕达哥拉斯三元组中，**最大的数一定是奇数。**

特别地，由 f_nf_{n+3}，$2f_{n+1}f_{n+2}$，$f_{n+1}^2+f_{n+2}^2$ 构成毕达哥拉斯三元组，将 $f_n=f_{n+2}-f_{n+1}$，$f_{n+3}+f_{n+1}$即得

- **求解 n 以内本原的毕达哥拉斯三元组个数**

根据 $\begin{cases}x=m^2-n^2\\ y=2mn\\ z=m^2+n^2\end{cases}$ ，只要枚举一下 m、n（$m,n\le\sqrt{n}$），然后将三元组乘以 i （保证 $i\times z$ 在范围内），即可求出所有的毕达哥拉斯三元组。

```
int x[N], y[N], z[N];
int pythagoras(int n) {
    int num = 0; //数组下标
    int res = 0; //本原三元组的个数
    int m = sqrt(n * 1.0);
    for (int i = 1; i <= m; i += 2) { //从 1 开始，每次 + 2，保证为奇数
        for (int j = 2; j <= m; j += 2) { //从 2 开始，每次 + 2 ，保证为偶数
            a = max(i, j); //大的为 m
            b = min(i, j); //小的为 n
            if (gcd(i, j) != 1) //要求 m,n 互质
                continue;
            x[num] = a * a - b * b;
            y[num] = 2 * a * b;
            z[num] = a * a + b * b;
            num++;
            if ((a * a + b * b) <= n) //保证在范围内
                res++;
        }
    }
    return res;
}
```

Problem A Find Integer （18年CCPC网络赛）

给你两个整数 n, a，找到整数 b, c，使$a^n + b^n = c^n$，其中 t 组数据，$1 \le t \le 10^6, 1 \le a \le 10^4, 1 \le n \le 10^9$。

Solution

费马大定理可知，$n > 2$或$n \le 0$ 时，不存在正整数解。

当 $n = 1$ 时我们直接构造即可。当 $n = 2$ 时，既是一个毕达哥斯拉三元组，按照上述定理求解即可。

```
int main() {
    int t, a, n;
    ll b, c;
    scanf("%d", &t);
    while (t--) {
        scanf("%d%d", &n, &a);
        if (n > 2 || n == 0)
            puts("-1 -1");
        else if (n == 1) printf("1 %d", a + 1);
        else {
            if (a & 1) {
                c = (a * a + 1) / 2;
                b = c - 1;
            } else {
```

```
                c = (a * a / 2 + 2) / 2;
                b = c - 2;
            }
            printf("%lld %lld\n", b, c);
        }
    }
}
```

0x82 费马大定理

费马大定理：

- $m > 2$时， $x^m + y^m = z^m$ 无正整数解
- 当$m = 2$，对于式子$a^2 + b^2 = c^2$（n为任意正整数）：
  - 当 a 为奇数时： $a = 2n + 1, c = n^2 + (n+1)^2, b = c - 1$
  - 当 a 为偶数时：$a = 2n + 2, c = 1 + (n-1)^2, b = c - 2$

0x83 平方和

- **费马平方和定理**

奇质数能表示为两个平方数之和的充分必要条件是该质数被 4 除余 1 。

1.如果两个整数都能表示为两个平方数之和的形式，则他们的积也能表示为两个平方数之和的形式。

$$\begin{aligned}(a^2+b^2)(c^2+d^2) &= a^2c^2 + a^2d^2 + b^2c^2 + b^2d^2 \\ &= (a^2c^2 + b^2d^2 - 2abcd) + (a^2d^2 + b^2c^2 + 2abcd) \\ &= (ac - bd)^2 + (ad + bc)^2\end{aligned}$$

2.如果一个能表示为两个平方数之和的整数，能被另一个能表示为两个平方数之和的素数整除，则他们的商也能表示为两个平方数之和。

即$\frac{a^2+b^2}{p^2+q^2} = (\frac{qp+bq}{p^2+q^2})^2 + (\frac{aq-bp}{p^2+q^2})^2$

3.如果 a, b 互质，则$a^2 + b^2$ 的所有因子都可以表示为两个平方数的和

4.任何形如 $4n + 1$ 的素数都能表示为两个平方数之和的形式

- **拉格朗日四平方和定理**

每个正整数都能表示为四个整数的平方和形式。

- **欧拉恒等式**

$$\begin{aligned}&(a^2+b^2+c^2+d^2)(x^2+y^2+z^2+w^2) \\ =\ &(ax+by+cz+dw)^2 + (ay-bx+cw-dz)^2 + (az-bw-cx+dy)^2 + (aw+bz-cy-dx)^2\end{aligned}$$

0x84 佩尔方程与连分数

0x84.1 佩尔方程与连分数

- **基本性质定理**

连分数是一种特殊的繁分数，其形式为：$a_1+\frac{1}{a_2+\frac{1}{a_3+\frac{1}{\dots}}}$ ，通常记为：$[a_1,a_2,\dots,a_n]=\frac{p_n}{q_n}$，其中 p_n 和 q_n 称为连分数多项式，对于任意的 a 均为一次式，它们的比值称为第 n 个渐进值渐进分数。

佩尔（`Pell`）方程是一种不定二次方程，其与连分数，二次型，代数论等有着重要的联系。

其形式为：$x^2-dy^2=1,(d\in N^+)$，其中 d 不为非平方数

- **佩尔方程迭代公式**

定义：设 p,q 为整数，且满足 $p^2-Dq^2=T$，则称 $a=p-q\sqrt{D}$ 给出该方程的解

- **推论**

设 $\begin{cases}a=x_1-y_1\sqrt{D}\\b=x_2-y_2\sqrt{D}\end{cases}$ 给出方程 $x^2-Dy^2=T$ 的解

则：$ab=A-B\sqrt{D}$ 给出方程 $x^2-Dy^2=T^2$ 的解，其中 $\begin{cases}A=x_1x_2+Dy_1y_1\\B=x_1y_2+x_2y_1\end{cases}$

取 $d=D,T=1$，则有佩尔方程 $x^2-dy^2=1,(d\in N^+)$

若佩尔方程的最小特解为 (x_1,y_1)，故有迭代公式：$\begin{cases}x_n=x_{n-1}x_1+dy_{n-1}y_1\\y_n=x_{n-1}y_1+y_{n-1}x_1\end{cases}$

将迭代公式写为矩阵的形式，有：$\begin{pmatrix}x_k\\y_k\end{pmatrix}=\begin{pmatrix}x_1&dy_1\\y_1&x_1\end{pmatrix}^{k-1}\begin{pmatrix}x_1\\y_1\end{pmatrix}$，可以通过矩阵快速幂找出第 k 大的解

- **连分数求解佩尔方程**

对于连分数 $[a_1,a_2,\dots,a_n]=\frac{p_n}{q_n}$ 的渐进值来讲，有着递归关系式：

$$\begin{cases}p_1=a_1,p_2=a_2a_1+1,\dots,p_{n+1}=a_{n+1}p_n+p_{n-1}\\q_1=1,\quad q_2=a_2,\quad \dots\dots,\quad q_{n+1}=a_{n+1}q_n+q_{n-1}\end{cases},n\geqslant 2$$

通过数学归纳法，可得到关系式：$p_{n+1}q_n-p_nq_{n+1}=(-1)^{n-1},n\geqslant 1$

定义：对于正整数 p、q，若有 $|p^2-a^2q^2|<a$，则比值 $\frac{p}{q}$ 必为 a 的一个渐近值

由此可得：佩尔方程的全部根的集合为 $x^2-dy^2=(p^2-dq^2)^n=1$

由佩尔方程的迭代公式 $\begin{cases}x_n=x_{n-1}x_1+dy_{n-1}y_1\\y_n=x_{n-1}y_1+y_{n-1}x_1\end{cases}$ 可得出佩尔方程的最小解

即：$\begin{cases}x=\frac{(p+\sqrt{d}\,q)^n+(p-\sqrt{d}\,q)^n}{2}\\y=\frac{(p+\sqrt{d}\,q)^n-(p-\sqrt{d}\,q)^n}{2\sqrt{d}}\end{cases}$

0x84.2 解佩尔方程

- **暴力寻找最小解**

```cpp
int x[N], y[N];
void pell(int &a, int &b, int d) { //暴力寻找pell方程最小解
    b = 1;
    while (true) {
        a = (LL)sqrt(d * b * b + 1);
        if (a * a - d * b * b == 1)
            break;
        b++;
    }
}
int main() {
    int d;
    while (scanf("%d", &d) != EOF) {
        int m = (int)sqrt((double)d);
        if (m * m == d) { //d不能为完全平方数
            cout << "No Solution" << endl;
            continue;
        }
        int a = 0, b = 0;
        pell(a, b, d); //暴力找到最小解
        cout << a << " " << b << endl;
    }
    return 0;
}
```

- **迭代公式求前 n 个解**
  使用迭代公式求解 $pell$ 方程的前 n 个解时，应先用暴力寻找到最小解，然后再套用迭代公式求出前 n 个解，由于 $pell$ 方程相邻两解之间的差值较大，n 一般很小

```cpp
int x[N], y[N];
void pell(int &a, int &b, int d) { //暴力寻找pell方程最小解
    b = 1;
    while (true) {
        a = (LL)sqrt(d * b * b + 1);
        if (a * a - d * b * b == 1)
            break;
        b ++ ;
    }
}
int main() {
    int d;
    while (scanf("%d", &d) != EOF) {
        int m = (int)sqrt((double)d);
        if (m * m == d) { //d不能为完全平方数
```

```cpp
            cout << "No Solution" << endl;
            continue;
        }
        int a = 0, b = 0;
        pell(a, b, d); //暴力找到最小解
        x[1] = a, y[1] = b; //第一组解
        for (int i = 2; i <= 10; i ++ ) { //递推公式
            x[i] = x[i - 1] * x[1] + 2 * y[i - 1] * y[1];
            y[i] = x[i - 1] * y[1] + y[i - 1] * x[1];
        }
        for (int i = 1; i <= 10; i ++ )
            cout << x[i] << " " << y[i] << endl;
    }
    return 0;
}
```

- **连分数法**
  当要求 $pell$ 方程的最小解时，暴力可能会 TLE，此时可以使用连分数法，其关键是计算连分数的展开

```cpp
int a[20000];
bool pell(int &x, int &y, int d) {
    int m = (int)sqrt((double)d);
    if (m * m == d) //d不能为完全平方数
        return false;
    //将d以连分数形式存储
    int num = 0; //连分数数位
    double sq = sqrt(d); //d的高精度根，相当于r0
    a[num++] = m; //存储整数部分
    int b = m; //当前整数部分
    int c = 1; //连分数最终展开时的分母
    double temp;//连分数展开时的每一项
    do {
        c = (d - b * b) / c;
        temp = (sq + b) / c;
        a[num++] = (int)(floor(temp));
        b = a[num - 1] * c - b;
    } while (a[num - 1] != 2 * a[0]); //当有一位等于整数两倍时结束
    //将连分数形式化为分子分母形式，即求p、q两个值
    int p = 1, q = 0;
    for (int i = num - 2; i >= 0; i -- ) {
        int temp = p;
        p = q + p * a[i];
        q = temp;
    }
    if ((num - 1) % 2) { //连分数长度为奇数时
        x = 2 * p * p + 1;
```

```cpp
        y = 2 * p * q;
    } else { //连分数长度为偶数时
        x = p;
        y = q;
    }
    return true;
}
int main() {
    int d;
    while (scanf("%d", &d) != EOF) {
        int x, y;
        if (pell(x, y, d))
            cout << x << " " << y << endl;
        else
            cout << "No Solution" << endl;
    }
    return 0;
}
```

0x84.3 竞赛例题选讲

Problem A Square Number（hdu 2281)

输入一个数 N，找到满足下式的 n, x，且 $n \le N$。

$x^2 = \frac{1^2+2^2+\cdots+n^2}{n}$

Solution

将右边式子的分子求和化简，有：$x^2 = \frac{(n+1)(2*n+1)}{6}$

移项化简，有：$(4n+3)^2 - 48 * x^2 = 1$

我们发现满足佩尔方程的形式，直接带入佩尔方程公式求解即可。

Problem B Street Numbers（poj 1320)

有 m 个编号从 1 到 m 的房子，问是否存在 $1+2+3+...+(N-1) = (N+1)+(N+2)+...+(M)$，求出前 10 个 n, m。

Solution

将左右两端的等差数列求和，有：$(2M+1)^2 - 8N^2 = 1$

满足佩尔方程的形式：$x = 2m+1, y = n$

可以得到最小的一组解为 $x = 3, y = 1$，直接求前 10 个佩尔方程的解即可。

0x90 高斯整数

0x90.0 复数

复数分为实部和虚部，可以描述为一个二元组 (x, y)，表示这个数等于$x + y\sqrt{-1}$。一般用 i 表示 $\sqrt{-1}$。

由于是个二元组，所以它在理解的时候可以抽象为一个二维向量，分布在平面直角坐标系上。

事实上，它确实也有不少性质和向量相同。

- **复数的模**

定义复数的模，为复平面原点到 (a, b)的距离，即$|z| = \sqrt{x^2 + y^2}$

定义 z 的**范数** $N(z) = |z|^2 = x^2 + y^2$。

- **共轭复数**

复数 $z = a + bi$ 的共轭是 $z' = a - bi$，记作 $\bar{z}$。

性质：$z \times \bar{z} = a^2 + b^2$，$|z| = |\bar{z}|$。

- **复数的大小**

复数可以看做和向量一样，无法比较大小。

- **复数的加减**

对应部 的加减，如 $(a, c) + (b, d) = (a + b, c + d)$

- **复数乘法**

两个复数 $z_1 = a_1 + b_1 i,\ z_2 = a_2 + b_2 i$ 相乘的结果为

$z_0 \ = \ z_1 \times z_2 \ = \ (a_1 + b_1 i) \times (a_2 + b_2 i) = a_1 a_2 + a_1 b_2 i + a_2 b_1 i + b_1 b_2 i^2$

又因为 $i^2 = -1$，所以

$z_0 \ = \ (a_1 a_2 - b_1 b_2) + (a_1 b_2 + a_2 b_1) i$

在复平面上，z_1，z_2，z_0 所对应的幅角 θ_1，θ_2，θ_0

有关系：$\theta_0 = \theta_1 + \theta_2$

- **复数的除法**

对于两个复数 $z_1 = a_1 + b_1 i,\ z_2 = a_2 + b_2 i$，他们相除的结果为

$z_0 = \frac{z_1}{z_2}$

考虑分数上下同时乘 $\overline{z_2}$ ，有

$z_0 \ = \ \frac{z_1 \overline{z_2}}{a_2^2 + b_2^2}$

分母是一个实数，可以直接将分子的实部虚部除以分母。

或者也可以展开：

$\frac{a + c \times i}{b + d \times i} = \frac{(a + c \times i)(b - d \times i)}{(b + d \times i)(b - d \times i)} = \frac{(ab + cd) + (bc - ad) \times i}{b^2 + d^2}$

即：

$\frac{(a,b)}{(c,d)} = (\frac{ab+cd}{b^2+d^2}, \frac{bc-ad}{b^2+d^2})$

- **复数指数幂：**

有**欧拉公式**$e^{i\theta} = \cos\theta + i\sin\theta$其中 e 是自然对数的底数

当取 $\theta = \pi$ 时，有$e^{i\pi} = \cos\pi + i\sin\pi$又因为$\cos\pi = 1, \ sin\pi = 0$

所以$e^{i\pi} = -1$

也就是：$e^{i\pi} + 1 = 0$

0x91 高斯整数

- **定义：**

形如 $a + bi$ （其中a, b 是整数）的复数被称为高斯整数，高斯整数全体记作集合 $Z[i] = \{a + bi | a, b \in Z\}, \ where \ i^2 = -1$，换言之，高斯整数是实部和虚部都为整数的复数。由于高斯整数在乘法和加法下交换，它们形成了一个**交换环**。也就是高斯整数是加、减、乘运算下封闭。

> 满足**加法**、**乘法**以及**交换律**、**结合律**、**分配率**并有 0 **元**和**单位元**的集合称为**交换环**。

在复平面上，高斯整数是二维复平面上的**整点**(a, b)。

高斯整数的模是它和自己共轭复数的乘积，即范数 $N(a + bi) = (a + bi)(a - bi) = a^2 + b^2$，它的模可以表示为两个数字的平方和，所以不能表示为 $4k + 1, \ where \ k \in Z$。

- **整除**

设 α, β 是高斯整数，我们称 α 整除 β ，是指存在一个高斯整数 γ，使得 $\beta = \alpha\gamma$。同整数集，若 α 整除 β ，记作 $\alpha \mid \beta$反之记作 ，$\alpha \nmid \beta$ 。

- **带余除法**

也称欧几里得除法，高斯整环 $Z[i]$ 是欧几里得环，所以它具有很多在整数域和多项式域上成立的特性，比如辗转相除，裴蜀定理，主理想，欧几里德引理，唯一分解定理，中国剩余定理。

定理91.1： 设 α, β 为高斯整数且 $\beta \neq 0$，则存在高斯整数 γ 和 ρ ，使得 $\alpha = \beta\gamma + \rho$。

而且 $0 \leq N(\rho) \leq N(\beta)$。这里的 γ 被成为商， ρ 被称为 余数 。

欧几里德引理

$$\begin{aligned} &\forall a, \ b, \ p \quad where \ p \ is \ a \ prime \\ &\qquad p|ab \Rightarrow p|a \ or \ p|b \end{aligned}$$

如果将高斯整数 a 写为 $a = bq + r$，有$N(r) \leq \frac{N(b)}{2}$。

证明：

令 $\frac{a}{b} = x + yi$，令$-\frac{1}{2} \leq x - m \leq \frac{1}{2}$， $-\frac{1}{2} \leq y - n \leq \frac{1}{2}$，令$q = m + ni$，由$a = bq + r$，$r = b(x - m + (y - n)i)$，故$N(r) \leq \frac{N(b)}{2}$。

0x92 高斯素数

定义92.1： 若 $\epsilon \mid 1$，则称高斯整数 ϵ 是 **单位** ，若 ϵ 是单位，则称 $\epsilon\alpha$ 是高斯整数 α 的一个相伴，高斯整数的单位为 $1,-1,i,-i$。

定义92.2： 若非零高斯整数 π 不是单位，而且只能够被单位和它的相伴整除，则称之为高斯素数。

定理92.3： 若 π 是高斯整数，而且 $N(\pi)=p$，其中 p 是有理整数，则 π 和 $\bar{\pi}$ 是高斯素数，而 p 不是高斯素数。

高斯整数形成了一个唯一分解域。高斯整数只有当且仅当它是一个素数时才不可约分，高斯整数中 $Z[i]$ 的素数称为高斯素数。

1. 高斯素数的共轭复数依然为高斯素数。
2. 高斯素数的相伴复数依然为高斯素数。

   $\forall a+bi \in Z[i], -a+bi \in Z[i]$

3. 作为高斯素数的整素数 p 是 $4k+3$ 型素数，其余的素数可以写为 2 个共轭高斯素数的乘积
4. 一个高斯整数 $a+bi$ 是高斯素数当且仅当以下两个条件之一成立：

- a,b 中一个为 0，且另一个数字为 $4k+3$ 型素数

  若 p 为 $4k+3$ 型素数，存在 d，d 不等于单位元，也不等于 p ，满足 $d \mid p$ ，则 $d\bar{d} \mid p$ ，由于 $d\bar{d}$ 为整数，有 $d\bar{d}=p$，得出 $p=a^2+b^2$，$a^2+b^2 \equiv 0,1,2(mod\ 4)$ ，与 p 为 $4k+3$ 型素数矛盾。

- a,b 都非 0 ，且 a^2+b^2 是素数

  令 $u=a+bi$ ，则 $u\bar{u}=p$，p 为一个素数，若存在存在 d，d 不等于单位元，也不等于 u，$d|u$，则 $d\bar{d} \mid p$，由于 $d\bar{d}$ 为整数，有 $d\bar{d}=p$，得出 $d=u$，与假设矛盾。

0x93 唯一分解

由于高斯整环是一个唯一分解域，所以每一个高斯整数都可以写为一个单位元和若干高斯素数的乘积，这种分解是唯一的（忽略共轭和相伴）。

$\forall a \in Z[i],\ a=u\cdot(1+i)^{e_0}\prod p_m^{e_i},\ where\ u\ \in \{1,-1,i,-i\},\ 0 \le e_i$

0x94 最大公约数

两个高斯整数的最大公约数并不唯一，加入 d 是 a 与 b 的最大公约数，则 a,b 的最大公约数为 $d,-d,id,-id$。

若$a=i^k\prod p_m^{\nu_m},\ b=i^n\prod p_m^{\mu_m}$，则其中一个最大公约数为

$d=\prod p_m^{\lambda_m},\ where\ \lambda_m=min(\nu_i,\ mu_i)$

可以根据带余除法里的结论，进行辗转相除，时间复杂度为 $O(log(n))$。

```
Complex div(Complex a, Complex b) {
    long double mo = b.norm();
    Complex c = a * b.conj();
    long double r = 1. * c.r / mo, i = 1. * c.i / mo;
    return Complex(round(r), round(i));
}
Complex gcd(Complex a, Complex b) {
    if (b.r == 0 && b.i == 0) return a;
    Complex c = div(a, b);
    return gcd(b, a - b * c);
}
```

0x95 同余和剩余系

给定一个高斯整数 z_0 ，对于高斯整数 z_1, z_2 ，如果它们的差是 z_0 的整数倍，即 $z_1 - z_2 = kz_0, k \in Z$，那么称 z_1 与 z_2 模 z_0 同余，写作 $z1 \equiv z2 \pmod{z0}$。

模 z_0 同余是一种等价关系，它将高斯整数分成若干个等价类，称为剩余类，剩余类写作 Z/z_0Z ，剩余类形成了一个交换环。

0x96 费马二平方和定理

p 是一个素数，p 可以写成两个平方数的和，当且仅当 $p = 2$ 或 $p \equiv 1 \pmod{4}$。

充分性：令 $p = a^2 + b^2$ ，对任意一个整数 x ，有 $x2 \equiv 0, 1, 2 \pmod{4}$，故 $a^2 + b^2 \equiv 0, 1, 2 \pmod{4}$，由于 p 是素数，所以 $p = 2$ 或 $p \equiv 1 (mod 4)$。

必要性：当 $p = 2$，$2 = 12 + 12$，当 p 为 $4k + 1$ 型素数，由于 p 不是高斯素数，所以可以进行分解，即 $p = u\bar{u}, u = a + bi$，故 $p = a^2 + b^2$。

0x97 分解4k+1型素数

p 为 $4k + 1$ 型素数，如果存在 $k2 \equiv -1 \pmod{p}$ ，则 $p|(k + i)(k - i)$，由于 $p = u\bar{u}$，有 $u|(k + i)$，故 $u = (k + i, p)$ ，即可将 p 分解为两个平方数的和。

由于有一半的数在模p下存在平方剩余，随机一个 t ，检验 $t^{\frac{p-1}{2}} \equiv -1 (mod\, p)$，令 $k = t^{\frac{p-1}{4}}$，时间复杂度为 $O(Tlog(p)), T$ 为测试次数。

0x98 构造 $a^2 + b^2 = n$的方案

1. 首先将 n 分解为 $n=\prod p_m^{e_m}$ 的形式
2. 将 2 分解为 $(1+i)(1-i)$，将 $4k+1$ 型素数分解为 $u\bar{u}$，其中 u 为高斯素数，$4k+3$ 型素数不可分解
3. $n=\prod p_m^{e_i}$，$0\le e_i$，由$n=u\bar{u}$，故需要将n中的高斯素数分成两部分，使得两部分共轭。考虑素数 p_m，对于 $4k+1$ 型素因子，分解为 $u^{e_m}\bar{u}^{e_m}$，左边可以放 k 个 u，e_m-k 个 $\bar{u}$；对于 $4k+3$ 型素因子，因为不能进行分解，所以左右两边的数量应该一样多，即 $4k+3$ 型的素数 p_m 出现的次数必须为偶数。

令 $f(n)=\frac{1}{4}\sum_{x,y\in Z}[x^2+y^2=n]$，除以 4 是因为由 4 个单位元。由于素因子可以分开考虑，所以该函数为积性函数。

1. $f(2^k)=1$
2. $f(p^e)=e+1$，当 p 为 $4k+1$ 型素数
3. $f(p^{2e+1})=0$，$f(p^{2e})=1$，当 p 为 $4k+3$ 型素数

0x99 竞赛例题选讲

Problem A A math problem （HDU 2650)

给出 $a+bj$，其中 $j=\sqrt{-2}$，判断 $a+bj$ 是否为高斯素数。

Solution

我们按照上面证明的判断高斯素数的方法直接模拟即可。

注意这里不是正常的复数，这里设 $j=\sqrt{-2}$。

$|j|=\sqrt{a^2+(\sqrt{-2}b)^2}$

很明显原来的判定中 a^2+b^2 就变成了 a^2+2b^2。同理若 $j=\sqrt{-k}$，就是 a^2+kb^2。

由于这里的数据很大，所以我们判断素数的时候需要用到Miller$_R$abin算法

```
typedef long long ll;
ll mul(ll a, ll b, ll m) {
    ll ans = 0;
    while (b) {
        if (b & 1) {
            ans = (ans + a) % m;
            b--;
        }
        b >>= 1;
        a = (a + a) % m;
    }
    return ans;
}
ll qpow(ll a, ll b, ll m) {
    ll ans = 1;
    a %= m;
```

```cpp
    while (b) {
        if (b & 1) {
            ans = mul(ans, a, m);
            b--;
        }
        b >>= 1;
        a = mul(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;
    }
    for (int i = 0; i < Times; i++) {
        a = rand() % (n - 1) + 1;
        x = qpow(a, m, n);
        for (int j = 0; j < k; j++) {
            y = mul(x, x, n);
            if (y == 1 && x != 1 && x != n - 1)
                return false;
            x = y;
        }
        if (y != 1)
            return false;
    }
    return true;
}

int main() {
    ll a, b;
    while (~scanf("%lld%lld", &a, &b)) {
        if (a == 0) {
            if (b % 4 == 3 && Miller_Rabin(b))
                puts("Yes");
            else
                puts("No");
        } else {
            ll t = a * a + 2 * b * b;
```

```
            if (Miller_Rabin(t))
                puts("Yes");
            else
                puts("No");
        }
    }
    return 0;
}
```

Problem B Gaussian Prime Factors （POJ 3361)

求一个数的高斯素因子。

Solution

我们根据高斯素数的判定原理，我们只需要将输入的数进行质因数分解，如果得到的质数是 $4x+3$ 的形式就保存，不是就枚举,找到 a，求得 b 看 b 是否是整数，找到 a 和 b，保存 $a+bi$ 和 $a-bi$，最后排序输出即可。

```
typedef long long ll;
struct complex
{
    ll a, b;
    char op;
} c[1100];
ll cp;
bool cmp(complex m, complex n)
{
    bool ret = 0;
    if (m.a < n.a)
        ret = 1;
    if (m.a == n.a && m.b < n.b)
        ret = 1;
    if (m.a == n.a && m.b == n.b && m.op == '+' && n.op == '-')
        ret = 1;
    return ret;
}
void getResult(ll p)
{
    ll i, j;
    if ((p - 3) % 4) {
        for (i = 1;; i++) {
            j = ll(sqrt(double(p - i * i)));
            if (i * i + j * j == p) {
                c[cp].a = i, c[cp].b = j, c[cp++].op = '+';
                c[cp].a = i;
                c[cp].b = j, c[cp++].op = '-';
```

```
                break;
            }
        }
    }
    else
        c[cp].a = p, c[cp].b = 0, c[cp++].op = '*';
}
int main()
{
    ll in, num = 0, i, j, k, tmp, isFir;
    while (scanf("%lld", &in) != EOF) {
        tmp = in;
        isFir = 1;
        cp = 0;
        printf("Case #%lld: ", ++ num);

        for (i = 2; i * i < tmp; i ++ ) {
            if (tmp % i == 0) {
                getResult(i);
                while (tmp % i == 0)
                    tmp /= i;
            }
        }
        if (tmp != 1)
            getResult(tmp);
        sort(c, c + cp, cmp);
        for (i = 0; i < cp; i++) {
            if (i)
                printf(", ");
            if (!c[i].b)
                printf("%lld", c[i].a);
            else if (c[i].b == 1)
                printf("%lld%cj", c[i].a, c[i].op);
            else
                printf("%lld%c%lldj", c[i].a, c[i].op, c[i].b);
        }
        printf("\n");
    }
    return 0;
}
```

0xA0 杂项

0xA1 哥德巴赫猜想及其拓展

哥德巴赫猜想

哥德巴赫猜想认为任一大于2的偶数（$n \geq 4$），都可表示成两个素数之和。

> 尽管如今仍未得到证明，但是根据那群科学家的验证，如果有一个偶数不能拆分，那么至少是一个几百位的偶数，因此我们在竞赛的小数据范围内，所有的拆分一定是可以实现的

拓展

- 任何一个大于5的奇数都可以表示为三个素数之和
- 任何一个大于8的整数都可以表示为四个素数之和

0xA2 幂级数展开式常用公式

$$e^x=\sum_{n=0}^{\infty}\frac{1}{n!}x^n=1+x+\frac{1}{2!}x^2+\ldots+\frac{1}{n!}x^n+\ldots,x\in(-\infty,+\infty)$$

$$\sin x=\sum_{n=0}^{\infty}\frac{(-1)^n}{(2n+1)!}x^{2n+1}=x-\frac{1}{3!}x^3+\frac{1}{5!}x^5-\ldots+\frac{(-1)^n}{(2n+1)!}x^{2n+1}+\ldots,x\in(-\infty,+\infty)$$

$$\cos x=\sum_{n=0}^{\infty}\frac{(-1)^n}{(2n)!}x^{2n}=1-\frac{1}{2!}x^2+\frac{1}{4!}x^4-\ldots+\frac{(-1)^n}{(2n)!}x^{2n}+\ldots,x\in(-\infty,+\infty)$$

$$\ln(1+x)=\sum_{n=0}^{\infty}\frac{(-)^n}{n+1}x^{n+1}=x-\frac{1}{2}x^2+\frac{1}{3}x^3-\ldots+\frac{(-)^n}{n+1}x^{n+1}+\ldots,x\in(-1,1]$$

$$\frac{1}{1-x}=\sum_{n=0}^{\infty}x^n=1+x+x^2+x^3+\ldots+x^n+\ldots,x\in(-1,1)$$

$$\frac{1}{1+x}=\sum_{n=0}^{\infty}(-1)^n x^n=1-x+x^2-x^3+\ldots+(-1)^n x^n+\ldots,x\in(-1,1)$$

$$(1+x)^\alpha=1+\sum_{n=1}^{\infty}\frac{\alpha(\alpha-1)\ldots(\alpha-n+1)}{n!}x^n=1+\alpha x+\frac{\alpha(\alpha-1)}{2!}x^2+\ldots+\frac{\alpha(\alpha-1)\ldots(\alpha-n+1)}{n!}x^n+\ldots,x\in(-1,1)$$

$$\arctan x=\sum_{n=0}^{\infty}\frac{(-)^n}{2n+1}x^{2n+1}=x-\frac{1}{3}x^3+\frac{1}{5}x^5+\ldots+\frac{(-)^n}{2n+1}x^{2n+1}+\ldots,x\in(-1,1)$$

$$\arcsin x=x+\frac{1}{6}x^3+\frac{3}{40}x^5+\ldots,x\in(-1,1)$$

$$\tan x=x+\frac{1}{3}x^3+\frac{2}{15}x^5+\frac{17}{315}x^7+\frac{62}{2835}x^9+\frac{1382}{155925}x^{11}+\ldots.x\in(-1,1)$$

$$\sec x=1+\frac{1}{2}x^2+\frac{5}{24}x^4+\frac{61}{720}x^6+\ldots,x\in(-\frac{\pi}{2},\frac{\pi}{2})$$

$$\csc x=\frac{1}{x}+\frac{1}{6}x+\frac{7}{360}x^3+\frac{31}{15120}x^5+\ldots,x\in(0,\pi)$$

0xA3 特征根法求数列通项公式

特征根法是解常系数线性微分方程的一种通用方法。

特征根法也可用于通过数列的递推公式求通项公式，原理与微分方程相同。

对形如 $a_{n+2}=pa_{n+1}+qa_n$ 的递推式中，显然可以把原式转化为$a_{n+2}-x_1a_{n+1}=x_2(a_{n+1}-x_1a_n)$

那显然x1，x2为方程x²-px-q=0的两根,由此我们可以简单推出特征根数列的通项公式。

对形如 $a_{n+2}=pa_{n+1}+qa_n$ 的递推式中（p，q为常数）有 $a_n=C_1\alpha^n+C_2\beta^n$ （α，β为特征解，C1，C2为常数）

- **竞赛例题选讲**

Problem A text1

在数列an中，a1=-1,a2=2，当n∈N+时，存在 $a_{n+2}=5a_{n+1}-6a_n$ ，求数列的通项?

$\because a_{n+2}=5a_{n+1}-6a_n$ (将 a_{n+2} 设为 x^2 ，将 a_{n+1} 设为x，将 a_n 设为1)

特征方程为 $x^2-5x+6=0$ ，易解得特征解为x1=2，x2 = 3

则α=2，β=3， $a_n=C_1 2^n+C_2 3^n$,代入a1=-1，a2 = 2，

解出 $C_1=-\frac{5}{2}$ $C_2=\frac{4}{3}$

得：$a_n=-\frac{5}{2}\cdot 2^n+\frac{4}{3}\cdot 3^n$

Problem B text2

设数列的前n项和为 S_n ，已知 $a_n=\frac{S_{n+1}-2}{4}$,a1= 1,试求 a_n 的通项公式?

有个Sn，不过没关系，消掉依然特征根。

$\because a_1=2$, $S_{n+1}=4a_n+2$, $S_2=a_1+a_2=4a_1+2$

易得 $a_2=5$ ，又有 $S_{n+1}=4a_n+2, S_{n+2}=4a_{n+1}+2$ 两式相减

$a_{n+2}=4a_{n+1}-4a_n$ 则其特征方程为 $x^2-4x+4=0$ 解得x1=x2=2

两根相等时，我们这样设，设 $a_n=(C_1+nC_2)2^n$,代入a1=2，a2=5

我们就解得 $C_1=-\frac{1}{4}$， $C_2=\frac{3}{4}$，

故 $a_n=(-\frac{1}{4}+\frac{3n}{4})*2^n$ $=(3n-1)2^{n-2}$

- **竞赛例题选讲**

Problem A The Nth Item （2019-ACM-ICPC-南昌区网络赛 H)

已知：$f(n)=3\cdot f(n-1)+2\cdot f(n-2),(n\geq 2)$，求$f(n) \pmod{998244353}$。$T$组询问，其中$T\leq 10^7, n\leq 10^{18}$。

Solution

我们利用上面的特征根法求得通项公式为

$$a_n=\frac{\sqrt{17}}{17}\cdot\left(\left(\frac{3+\sqrt{17}}{2}\right)^n-\left(\frac{3-\sqrt{17}}{2}\right)^n\right)$$

$\sqrt{17} \pmod{998244353}$我们可以用二次剩余求得。然后就可以用快速幂在 $O(log(n))$ 的时间复杂度内求得 a_n 。但是因为$T\leq 10^7$，所以还需优化。

$n\leq 10^{18}$，我们通过欧拉降幂之后得到 $n\leq 10^9$

我们令 $k=\lfloor\sqrt{n}\rfloor$ 则：

$$n=k\cdot t+r$$

$$x^n=x^{k\cdot t+r}\Leftrightarrow x^n=x^{k\cdot t}+x^r(t,r\leq k)$$

然后我们只需要预处理出 $x^r, r \le k$ 以及 $x^{k \cdot t}, t \le k$，这样我们的x^n 就可以 $O(1)$ 查询了，而不用借助快速幂。

Code

```cpp
#define int long long
const int maxn = 100005, INF = 0x3f3f3f3f;
const int mod = 998244353;
int inv17;
//求解x^2=n(mod p),即x=sqrt(n)(mod p) O(sqrt(p))
/*类似复数 单位元为w(复数的单位元为-1)*/
struct Complex {
    int x, y, w;
    Complex() {}
    Complex(int x, int y, int w) : x(x), y(y), w(w) {}
};
/*类复数乘法 */
Complex mul(Complex a, Complex b, int p) {
    Complex ans;
    ans.x = (a.x * b.x % p + a.y * b.y % p * a.w % p) % p;
    ans.y = (a.x * b.y % p + a.y * b.x % p) % p;
    ans.w = a.w;
    return ans;
}
/*类复数快速幂 */
Complex Complexfpow(Complex a, int b, int mod) {
    Complex ans = Complex(1, 0, a.w);

    while (b) {
        if (b & 1)
            ans = mul(ans, a, mod);

        a = mul(a, a, mod);
        b >>= 1;
    }

    return ans;
}
int qpow(int a, int b, int mod) {
    int ans = 1;
    a %= mod;

    while (b) {
        if (b & 1)
            (ans *= a) %= mod;

        (a *= a) %= mod;
```

```cpp
        b >>= 1;
    }

    return ans;
}
/*求解x^2=n(mod p) */
int solve(int n, int p) {
    n %= p;

    if (n == 0)
        return 0;

    if (p == 2)
        return n;

    if (qpow(n, (p - 1) / 2, p) == p - 1)
        return -1;  /*勒让德定理判断n不是p的二次剩余 */
    mt19937 rnd(time(0));//更加高效的STL自带随机数
    int a, t, w;
    do {
        a = rnd() % p;
        t = a * a - n;
        w = (t % p + p) % p;                    /*构造w=a^2-n */
    } while (qpow(w, (p - 1) / 2, p) != p - 1); /*找到一个w不是p的二次剩余 */

    Complex ans = Complex(a, 1, w);
    ans = Complexfpow(ans, (p + 1) / 2, p); /*答案为(a+w)^{(p+1)/2} */
    return ans.x;
}
pair<int, int> bit1[maxn], bit2[maxn];
signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    inv17 = qpow(17, mod - 2, mod);
    int x = solve(17, mod); //二次剩余
    int lim = ceil(sqrt(1e9));
    int s1 = (3 + x) % mod * qpow(2, mod - 2, mod) % mod,
        s2 = (3 - x) % mod * qpow(2, mod - 2, mod) % mod;
    bit1[0].first = bit1[0].second = bit2[0].first = bit2[0].second = 1;
    for (int i = 1; i <= lim; i++) { //预处理
        bit1[i].first = bit1[i - 1].first * s1 % mod;
        bit1[i].second = bit1[i - 1].second * s2 % mod;
        bit2[i].first = qpow(s1, i * lim, mod);
        bit2[i].second = qpow(s2, i * lim, mod);
    }
    int q, n;
    cin >> q >> n;
```

```cpp
    int ans = 0;
    while (q--) {
        int tmp = 0;
        if (n == 0)
            tmp = 0;
        else if (n == 1)
            tmp = 1;
        else {
            int t = n % (mod - 1);
            int t2 = t / lim, t1 = t % lim;
            tmp = (bit1[t1].first * bit2[t2].first % mod - bit1[t1].second * bit2[t2].second % mod) % mod * x % mod *
                  inv17 % mod;
        }
        tmp = (tmp + mod) % mod;
        n = tmp * tmp ^ n;
        ans ^= tmp;
        // cout<<n<<endl;
    }
    cout << (ans + mod) % mod << endl;
    return 0;
}
```