Căn bậc hai modulo

Căn bậc 2 modulo

Người viết:

- Nguyễn Minh Hiển - Trường Đại học Công nghệ, ĐHQGHN

Reviewer:

- Phạm Công Minh - Trường Đại học Công nghệ, ĐHQGHN

Đôi khi, chúng ta sẽ gặp những bài tập như tính $\sqrt{x} \bmod p$ hay thậm chí như tính số Fibonacci $F_n \bmod p$. Mà chúng ta biết, công thức tổng quát:

$$F_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$$

Việc xuất hiện $\sqrt{5}$ đặt ra nhiều thách thức cho việc tính toán nhanh F_n, nhưng đồng thời cũng mở ra những phương pháp mới để chinh phục được bài toán $F_n \bmod p$

Một số định nghĩa

- Số nguyên dương a được gọi là **thặng dư bình phương** modulo p nếu: $\exists x : x^2 \equiv a \pmod p$Khi này, x được gọi là căn bậc hai của a modulo p.
- **Ký hiệu Legendre**: với p là số nguyên tố lẻ $\left(\frac{a}{p}\right) = \begin{cases} 0, & \text{nếu } a \equiv 0 \pmod p \\ 1, & \text{nếu } a \text{ là thặng dư bình phương mod } p \\ -1, & \text{nếu } a \text{ không là thặng dư bình phương mod } p \end{cases}$

Kiểm tra thặng dư bình phương

Ta sử dụng **tiêu chuẩn Euler** (Euler's criterion) như sau. Với p nguyên tố lẻ:

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

Đến đây, ta sử dụng lũy thừa nhanh để tính.

```
int pow_mod(int a, int n, int p); // hàm tính lũy thừa nhanh modul
int legendre_symbol(int a, int p) {
```

```
    return pow_mod(a, (p - 1) >> 1, p);
}
```

Độ phức tạp: $O(\log p)$

Thặng dư bình phương modulo nguyên tố

Bài toán

VNOJ - Số học 1

Tìm tất cả x thỏa mãn phương trình:

$$x^2 \equiv a \pmod p$$

- Với $p = 2$, phương trình có đúng 1 nghiệm: $x \equiv 1 \pmod p$
- Với p lẻ, theo *định lý Lagrange*, phương trình có đúng 2 nghiệm $x \equiv \pm x_0 \pmod p$

$\implies$ Như vậy, ta sẽ tìm nghiệm trong trường hợp p lẻ.

Tìm thặng dư không chính phương bất kỳ

- Trước hết, ta cần tìm thặng dư "*không chính phương*" để thực hiện hai thuật toán bên dưới.
- Vì một nửa số phần tử trong tập $\{1, 2, \cdots, p-1\}$ là thặng dư không chính phương, nên ta sẽ duyệt từng số từ 1 cho đến khi gặp được số thỏa mãn. Để kiểm tra một số thỏa mãn hay không, ta sử dụng cách **Tiêu chuẩn Euler** bên trên.
- Để thuật toán hiệu quả hơn, bạn nên sinh số ngẫu nhiên và kiểm tra đến khi tìm được. Xác suất 1 lần thử tìm được là $\frac{1}{2}$, nên xác suất sau 32 lần thử mà bạn chưa tìm ra là $\frac{1}{2^{32}}$.

Thuật toán Tonelli-Shanks

$$x^2 \equiv a \pmod p$$

- Thuật toán:
  - *Bài viết xin không đề cập phần chứng minh thuật toán. Bạn đọc tham khảo tại Wikipedia*.
  - **Bước 1:** ta phân tích $p = Q \cdot 2^S + 1$ với Q lẻ
  - **Bước 2:** Chọn z là một thặng dư không chính phương bất kỳ.
  - **Bước 3:** Gán $x \leftarrow a^{\frac{Q+1}{2}}$
    $b \leftarrow a^Q$
  - **Bước 4:** Lặp
    - Tìm m nhỏ nhất $(0 \le m < r)$ sao cho $b^{2^m} \equiv 1 \pmod p$
    - Nếu $m = 0 \iff b \equiv 1 \pmod p$ thì x chính là đáp án cần tìm.
      Nếu $m > 0$ thì đặt $e = \dfrac{p-1}{2^{m+1}} = Q \cdot 2^{S-m-1}$ gán:

$$x \leftarrow x \cdot z^e$$
$$b \leftarrow b \cdot z^{2e}$$

- Code C++ minh họa:

```cpp
int pow_mod(long long a, long long k, long long M) {
    long long ans = 1;
    for (; k > 0; a = a * a % M, k >>= 1) {
        if (k & 1) {
            ans = ans * a % M;
        }
    }
    return ans;
}

int Tonelli_Shanks(int a, int p) {
    if (p == 2) {
        return (a & 1);
    }
    int S = 0, Q = p - 1;
    while (Q % 2 == 0) {
        S++;
        Q /= 2;
    }
    int z = 2;
    while (legendre_symbol(z, p) != p - 1) {
        z++;
    }

    int x = pow_mod(a, (Q + 1) >> 1, p), b = pow_mod(a, Q, p);
    int m, v, e, u;

    while (b % p != 1) {
        m = 0, v = 1; // v = 2^m
        while (pow_mod(b, v, p) != 1) {
            m++;
            v <<= 1;
        }
        e = Q << (S - m - 1);
        u = pow_mod(z, e, p);
        x = (1LL * x * u) % p;
        b = (((1LL * u * u) % p) * b) % p;
    }
    return x;
}
```

- Độ phức tạp: $O(\log^2 p)$

Trường hữu hạn

Định nghĩa

- Như các bạn đã biết:

$$\left(a+b\sqrt{k}\right)^n = u+v\sqrt{k}$$

Trong đó $a, b, u, v, k \in \mathbb{Z}$ và $\sqrt{k} \notin \mathbb{Z}$.

> Bạn đọc có thể thấy nó khá giống số phức, chỉ thay $i=\sqrt{-1}$ bằng $\sqrt{k}$ mà thôi.

Mục đích của chúng ta là tính u, v theo $\mod p$. Như các bạn nghĩ đến, chúng ta sẽ sử dụng phép lũy thừa nhanh và có chút thay đổi cho phù hợp bài toán:
- Ký hiệu: $\langle a, b\rangle = a+b\sqrt{k}$

- Phần tử đơn vị: $\langle a, b\rangle \times \langle 1, 0\rangle = \langle a, b\rangle$
- Xét phép nhân 2 số $\begin{aligned}\langle a, b\rangle \times \langle u, v\rangle &= \Big\langle (au+bvk),\ (av+bu)\Big\rangle \\ &= \Big\langle (au+bvk) \bmod p,\ (av+bu) \bmod p\Big\rangle\end{aligned}$
- Phép lũy thừa: $\langle a, b\rangle^k = \underbrace{\langle a, b\rangle \times \cdots \times \langle a, b\rangle}_{k \text{ thừa số}}$

> Các phép toán trên chỉ là một số tính chất của trường hữu hạn $\mathbb{F}_{p^2} = \mathbb{F}_p\left(\sqrt{k}\right)$. Để có kiến thức đầy đủ hơn, bạn đọc tham khảo trên Wikipedia .

Thuật toán Cipolla

$$x^2 \equiv a \pmod p$$

- Thuật toán:
  - *Bài viết xin không đề cập phần chứng minh thuật toán. Bạn đọc tham khảo tại Wikipedia*.
  - **Bước 1:** Tìm b sao cho $b^2 - a$ là thặng dư không chính phương modulo p

- **Bước 2:** Ta tính $x + y\sqrt{b^2 - a} = \left(b + \sqrt{b^2 - a}\right)^{(p+1)/2}$.
  Khi đó, $x \bmod p$ tìm được chính là nghiệm của bài toán.
  Nói cách khác là $\langle x, y\rangle = \langle b, 1\rangle^{(p+1)/2}$ trên $\mathbb{F}_p\left(\sqrt{b^2 - a}\right)$

- Code C++ minh họa
  Về cài đặt, như đã nói ở trên, $\langle x, y\rangle$ khá giống số phức nên việc cài đặt cũng tương tự như vậy.

```cpp
int a, p;
int k; // thặng dư không chính phương mod p

struct Complex {
    int re, im;

    Complex(int a = 0, int b = 0) {
        re = a;
        im = b;
    }

    Complex operator*(const Complex &o) {
        Complex res;
        res.re = (1LL * re * o.re + (1LL * im * k % p) * o.im) % p
        res.im = (1LL * re * o.im + 1LL * im * o.re) % p;
        return res;
    }

    Complex pow(long long k) {
        Complex res = Complex(1, 0), A = *this;
        while (k) {
            if (k & 1) res = res * A;
            A = A * A;
            k >>= 1;
        }
        return res;
    }
};

int Cipolla(long long a, long long p) {
    if (p == 2) {
        return (a & 1);
    }
    // Tìm k = b^2 - a, sao cho k không chính phương
    int b = 2;
    while (true) {
        b %= p;
        k = (b * b - a) % p;
```

```
            if (k < 0) k += p;
            if (legendre_symbol(k, p) == p - 1) break;
            b++;
        }
        // Ta cần tìm <b, 1>^((p+1)/2)
        return Complex(b, 1).pow((p + 1) >> 1).re;
    }
```

- Độ phức tạp: $O(\log^2 p)$

Fibonacci modulo p

Ngoài các phương pháp như *Nhân ma trận* hay *Khử nhân ma trận*, còn có một phương pháp khác sử dụng
Công thức tổng quát của Fibonacci:

$$F_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$$

Xét modulo p nguyên tố.

- **Nếu 5 là thặng dư bình phương modulo p**
  Ví dụ: Bài Codeforces - DZY Loves Fibonacci Numbers với $p = 10^9 + 9$.
  Ta tính được: $\sqrt{5} = 383008016 \pmod{p}$
  Sử dụng nghịch đảo modulo, ta có:

$$\begin{aligned} \frac{1}{\sqrt{5}} &\equiv 276601605 \pmod{p} \\ \frac{1+\sqrt{5}}{2} &\equiv 691504013 \pmod{p} \\ \frac{1-\sqrt{5}}{2} &\equiv 308495997 \pmod{p} \end{aligned}$$

$$\implies F_n \equiv 276601605 \cdot (691504013^n - 308495997^n) \pmod{p}$$

> So với việc tính lũy thừa của ma trận, tính lũy thừa của 2 số vẫn nhanh hơn rất nhiều.

- **Nếu 5 không là thặng dư bình phương modulo p**

Ví dụ: Bài VNOI - Fibonacci với $p = 10^9 + 7$
Ở bài này, ta sử dụng trường hữu hạn như ở trên.
Ta sẽ viết $\left(\frac{1+\sqrt{5}}{2}\right)^n = \langle u_1, v_1\rangle$ và $\left(\frac{1-\sqrt{5}}{2}\right)^n = \langle u_2, v_2\rangle$
Trên thực tế, vì F_n nguyên nên $u_1 - u_2 = 0$. Từ đó suy ra $F_n \equiv v_1 - v_2 \pmod{p}$.

Do sử dụng công thức tổng quát, cách này có một ưu điểm mà không cách nào có được, thể hiện qua bài toán bên dưới đây.

Ví dụ: **Bài F - ICPC miền Nam 2023**
Tính S theo modulo $p = 998244353$ nguyên tố với:

$$S = \sum_{i=0}^{n} (F_n)^k$$

Giới hạn: $n \leq 10^{18}, k \leq 10^6$.

Lời giải

Xét:

$$F_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right] = \frac{1}{\sqrt{5}}\left[u^n - \left(-u^{-1}\right)^n\right]$$

với $u = \frac{1+\sqrt{5}}{2} = \left\langle \frac{1}{2}, \frac{1}{2} \right\rangle$

Ta viết lại S như sau:

$$\begin{aligned}
S &= \sum_{n=0}^{N} (F_n)^k \\
&= \sum_{n=0}^{N} \frac{1}{\sqrt{5}^k}\left[u^n - \left(-u^{-1}\right)^n\right]^k \\
&= \sum_{n=0}^{N} \frac{1}{\sqrt{5}^k} \sum_{i=0}^{k} \binom{k}{i} (u^n)^i \left(-\left(-u^{-1}\right)^n\right)^{k-i} \\
&= \sum_{n=0}^{N} \frac{1}{\sqrt{5}^k} \sum_{i=0}^{k} \binom{k}{i} (-1)^{k-i} \left((-1)^{k-i} u^{2i-k}\right)^n \\
&= \frac{1}{\sqrt{5}^k} \sum_{i=0}^{k} \binom{k}{i} (-1)^{k-i} \sum_{n=0}^{N} \left((-1)^{k-i} u^{2i-k}\right)^n
\end{aligned}$$

Đặt $v = (-1)^{k-i} u^{2i-k}$, ta có:

$$S = \frac{1}{\sqrt{5}^k} \sum_{i=0}^{k} \binom{k}{i} (-1)^{k-i} \sum_{n=0}^{N} v^n$$

Và

$$\sum_{n=0}^{N} v^n = \begin{cases} \dfrac{v^{N+1} - 1}{v - 1} & \text{nếu } v \neq 1 \\ N + 1 & \text{nếu } v = 1 \end{cases}$$

Bây giờ, chúng ta cần giải quyết bài toán tính $\frac{1}{\langle a, b\rangle} \mod p$ nếu $\langle a, b\rangle \neq 1$.
Chú ý rằng $v = (-1)^{k-i}u^{2i-k}$ và $2i - k$ có thể âm.

- **Cách 1: Trường hữu hạn $\mathbb{F}_{p^2} = \mathbb{F}_p\left(\sqrt{5}\right)$**
  Ta có $t^{p^2} \equiv t \pmod p$
  Nếu $t \neq \langle 0, 0\rangle$ thì $t^{p^2-1} \equiv \langle 1, 0\rangle \pmod p$
  Thay $t = \langle a, b\rangle$, ta có ngay:
  $$\frac{1}{\langle a, b\rangle} \equiv \langle a, b\rangle^{p^2-2} \pmod p$$
- **Cách 2: Nhân liên hợp**
  $$\frac{1}{a + b\sqrt{5}} = \frac{a - b\sqrt{5}}{a^2 - 5b^2} \equiv \left(a^2 - 5b^2\right)^{p-2} \cdot \langle a, -b\rangle \pmod p$$
  Rõ ràng cách thứ hai này cho hiệu suất tốt hơn với khoảng $2\log p$ trong khi cách thứ nhất sử dụng tới $8\log p^2 \sim 16\log p$ phép nhân.

Cả hai cách này đều có độ phức tạp tiệm cận $O(K\log \mathrm{MOD})$, đủ để đánh bại bài toán này.

Code C++ tham khảo:

```
#include <bits/stdc++.h>
using namespace std;

const int MOD = 998244353;
const int inv = (MOD + 1) >> 1;
const int MAX = 1e6;
int fact[MAX + 5], ifact[MAX + 5];
long long N, K;

int pow_mod(long long A, long long k) {
    long long res = 1;
    while (k) {
        if (k & 1) res = res * A % MOD;
        A = A * A % MOD;
        k >>= 1;
    }
    return res;
}

void add(int &x, const int &y) {
    x += y;
    if (x >= MOD) x -= MOD;
}

struct Complex {
    int re, im;
```

```
Complex(int a = 0, int b = 0) {
    re = a;
    im = b;
}

bool operator==(const Complex &o) { return re == o.re && im ==

Complex operator+(const Complex &o) {
    Complex res = *this;
    add(res.re, o.re);
    add(res.im, o.im);
    return res;
}

Complex operator-(const Complex &o) {
    Complex res = *this;
    add(res.re, MOD - o.re);
    add(res.im, MOD - o.im);
    return res;
}

Complex operator*(const Complex &o) {
    Complex res;
    res.re = (1LL * re * o.re + 5LL * im * o.im) % MOD;
    res.im = (1LL * re * o.im + 1LL * im * o.re) % MOD;
    return res;
}

Complex operator*(int k) {
    return Complex(1LL * re * k % MOD, 1LL * im * k % MOD);
}

Complex pow(long long k) {
    if (k < 0) return this->pow(-k).inv();
    Complex res = Complex(1, 0), A = *this;
    while (k) {
        if (k & 1) res = res * A;
        A = A * A;
        k >>= 1;
    }
    return res;
}

Complex inv() {
    return Complex(re, MOD - im) *
        pow_mod((1LL * re * re + 5LL * (MOD - im) * im) % M
}
```

```
    Complex operator-() { return Complex(MOD - re, MOD - im); }
};
const Complex unit = Complex(1, 0);

void factorial() {
    fact[0] = 1;
    for (int i = 1; i <= MAX; ++i) fact[i] = 1LL * fact[i - 1] * i
    ifact[MAX] = pow_mod(fact[MAX], MOD - 2);
    for (int i = MAX; i >= 1; --i) ifact[i - 1] = 1LL * ifact[i] *
}

int C(int n, int k, int p) {
    return (1LL * fact[n] * ifact[k] % p) * ifact[n - k] % p;
}

int main() {
    cin.tie(NULL)->sync_with_stdio(false);
    factorial();
    cin >> N >> K;
    Complex res;
    for (int i = 0; i <= K; ++i) {
        Complex V1 = Complex(inv, inv).pow(2 * i - K);
        if ((K - i) & 1) V1 = -V1;

        Complex V2;

        if (V1 == unit)
            V2 = Complex((N + 1) % MOD, 0);
        else {
            V2 = V1.pow(N + 1);
            V2 = (V2 - unit) * (V1 - unit).inv();
        }

        V2 = V2 * C(K, i, MOD);

        if ((K - i) & 1)
            res = res - V2;
        else
            res = res + V2;
    }
    if (K & 1)
        cout << 1LL * res.im * pow_mod(5, MOD - 1 - K / 2) % MOD;
    else
        cout << 1LL * res.re * pow_mod(5, MOD - 1 - K / 2) % MOD;
}
```

Bài tập luyện tập

- VNOI - Số học 1
- VNOI - Số học 2
- Codeforces - Div.1C - DZY Loves Fibonacci Numbers
- Codeforces - Mathematical Field of Experiments
- Codeforces - G - New Year and the Factorisation Collaboration
- Codechef - FN
- Codechef - LCASQRT
- Codechef - GUESSPRM
- Bài F - ICPC miền Nam 2023

Tài liệu tham khảo

- Wikipedia:
  - https://en.wikipedia.org/wiki/Quadratic_residue
  - https://en.wikipedia.org/wiki/Euler's_criterion
  - https://vi.wikipedia.org/wiki/Ký_hiệu_Legendre
  - https://vi.wikipedia.org/wiki/Ký_hiệu_Jacobi
  - https://en.wikipedia.org/wiki/Tonelli–Shanks_algorithm
  - https://vi.wikipedia.org/wiki/Trường_hữu_hạn
  - https://en.wikipedia.org/wiki/Cipolla's_algorithm
  - https://en.wikipedia.org/wiki/Hensel's_lemma
  - https://en.wikipedia.org/wiki/Lagrange's_theorem_(number_theory)