数论—中国剩余定理、扩展中国剩余定理

中国剩余定理

定义

中国剩余定理（Chinese Remainder Theorem，CRT）

求解如下形式的一元线性同余方程组（其中 m 两两互质）：

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \qquad \cdots \\ x \equiv a_k \pmod{m_k} \end{cases}$$

过程

1. 计算所有模数的积 $M = \prod m_i$；
2. 对于第 i 个方程：
    1. 计算：$M_i = \dfrac{M}{m_i}$；
    2. 计算：$v_i = M_i^{-1} \pmod{m_i}$（乘法逆元）；
    3. 计算：$c_i = M_i v_i$。
3. 方程组在 $0 \sim M-1$ 范围内的唯一解为：$x = \sum_{i=1}^{k} a_i c_i \pmod{M}$。

证明

证明对于任意 $i \in [1,k]$，有 $x \equiv a_i \pmod{m_i}$。

当 $i \neq j$ 时，M_j 中乘进去了 m_i，所以有 $M_j \equiv 0 \pmod{m_i}$，
所以 $c_j \equiv M_j \equiv 0 \pmod{m_i}$。

又有 $c_i \equiv M_i \cdot M_i^{-1} \pmod{m_i} \equiv 1 \pmod{m_i}$，所以我们有：

$$\begin{aligned} x &\equiv \sum_{j=1}^{k} a_j c_j && \pmod{m_i} \\ &\equiv a_i c_i && \pmod{m_i} \\ &\equiv a_i && \pmod{m_i} \end{aligned}$$

即证明了解同余方程组的算法的正确性。

性质

1. 系数列表 $\{a_i\}$ 与解 x 之间是一一映射关系，方程组总是有唯一解。
   证明见：https://oi-wiki.org/math/number-theory/crt/
2. 设模 M 意义下的一个特解是 x_0，则通解为：$x = x_0 + kM$，其中 $k \in \mathbb{N}$.

代码

题目：P1495 中国剩余定理

▼ 点击查看代码

```
const int N = 10;

ll exgcd(ll a, ll b, ll &x, ll &y, ll d = 0)
{
    if (b == 0)
        x = 1, y = 0, d = a;
    else
        d = exgcd(b, a % b, y, x), y -= a / b * x;
    return d;
}

ll inv(ll a, const ll m, ll x = 0, ll y = 0)
{
    exgcd(a, m, x, y);
    return (x % m + m) % m;
}

int a[N], m[N];

int main()
{
    int n = rr;

    ll mul = 1;
    for (int i = 1; i <= n; ++i)
        m[i] = rr, a[i] = rr, mul *= m[i];

    ll x = 0;
    for (int i = 1; i <= n; ++i)

```

```
    {
        ll t = mul / m[i], c = inv(t, m[i]);
        x = (x + a[i] * t % mul * c % mul) % mul;
    }

    printf("%lld\n", x);
    return 0;
}
```

应用

CRT 合并

若要求一个大数 $r \bmod m$ 的结果 x，即求解关于 x 的线性同余方程 $x \equiv r \pmod m$；

则可以将模数分解为 $m = \sum\limits_{i=1}^{k} p_i$（即质因数分解，$p$ 两两互质）；

然后去求解 x 在模各个 p_i 意义下的结果，最后用 CRT 合并；则求出来的答案一定是一一对应的。

即将 $x \equiv r \pmod m$ 转换为一个线性同余方程组：$\begin{cases} x \equiv r \pmod {m_1} \\ x \equiv r \pmod {m_2} \\ \qquad \cdots \\ x \equiv r \pmod {m_k} \end{cases}$

CRT 合并的举例

题目：P2480 古代猪文。题面略...

求 $\dbinom{n}{m} \bmod 999911658$，即求 $x \equiv \dbinom{n}{m} \pmod {999911658}$.

根据上方的描述，因为 $999911658 = 2 \times 3 \times 4679 \times 35617$，原方程转化为：

$$\begin{cases} x \equiv \dbinom{n}{m} & \pmod 2 \quad (1) \\ x \equiv \dbinom{n}{m} & \pmod 3 \quad (2) \\ x \equiv \dbinom{n}{m} & \pmod {4679} \quad (3) \\ x \equiv \dbinom{n}{m} & \pmod {35617} (4) \end{cases}$$

使用 CRT 合并即可.

▼ 点击查看核心代码

```
// ...
const int N = 35620;

const ll MOD1 = 999911659;
const ll MOD2 = 999911658;

const ll m[4] = {2, 3, 4679, 35617};
const ll r[4] = {499955829, 333303886, 289138806, 877424796};   // 即 c[i]

// ...
int main()
{
    int n = rr, g = rr;
    if (g % MOD1 == 0)
        printf("0\n"), exit(0);

    // 分解质因数至 dv 数组...
    ll x = 0;
    for (int i = 0; i < 4; ++i)
    {
        MOD = m[i];

        // 预处理模 MOD 意义下的逆元...
        for (int j : dv)
            x = (x + lucas(n, j) * r[i] % MOD2) % MOD2;
    }

    ll r = qpow(g, x, MOD1);
    printf("%lld\n", r);
    return 0;
}
```

扩展中国剩余定理

定义

求解线性同余方程组$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \quad\quad\cdots \\ x \equiv a_k \pmod{m_k} \end{cases}$

但是模数 m_i 不一定两两互质。

此时因为 m_i 不一定与 m_j 互质，故不一定存在乘法逆元，即无法使用中国剩余定理。

做法

公式变形

先考虑前两个方程：$x \equiv a_1 \pmod{m_1}$、$x \equiv a_2 \pmod{m_2}$.
将它们转化为不定方程：$x = m_1 p + a_1 = m_2 q + a_2$，$p, q \in \mathbb{Z}$.
则有 $m_1 p - m_2 q = a_2 - a_1$.

解的情况

由裴蜀定理：
当 $\gcd(m_1, m_2) \nmid a_2 - a_1$ 时，无解；
当 $\gcd(m_1, m_2) \mid a_2 - a_1$ 时，有解。

求解不定方程

现在考虑如何使用扩展欧几里得算法求出一组可行解：

考虑方程：$m_1 p - m_2 q = a_2 - a_1$.
因为 $\gcd(m_1, m_2) \mid a_2 - a_1$，所以方程两边可以同时除去 $\gcd(m_1, m_2)$，同时设：

$$\begin{cases} k_1 & = \dfrac{m_1}{\gcd(m_1, m_2)} \\ \\ k_2 & = \dfrac{m_2}{\gcd(m_1, m_2)} \\ \\ z & = \dfrac{a_2 - a_1}{\gcd(m_1, m_2)} \end{cases}$$

得 $k_1 p - k_2 q = z$，且 $k_1 \perp k_2$；所以可以用扩展欧几里得算出：

方程 $k_1 s + k_2 t = 1$ 的一组解 (s, t)；因此有 $\begin{cases} p = zs \\ q = -zs \end{cases}$

回看刚开始的方程 $x \equiv a_1 \pmod{m_1}$，即可得出一个特解：

$$
\begin{aligned}
x_0 &= m_1 p + a_1 \\
&= m_1 \cdot zs + a_1 \\
&= \frac{m_1 s \times (a_2 - a_1)}{\gcd(m_1, m_2)} + a_1
\end{aligned}
$$

手模一下可知新的方程是模 $\operatorname{lcm}(m_1, m_2)$ 意义下的。

然后再考虑将特解转为通解，这一点很简单，在此引用 rxj 的一句话：从线性代数的角度讲，这个通解的构造方式是十分平凡的。对 $\operatorname{lcm}(m_1, m_2)$ 取模的结果，将整个整数集划分成了 $\operatorname{lcm}(m_1, m_2)$ 个等价类，哪个等价类里面有特解，那整个等价类肯定全都是解。

也就是通解 $x' = x_0 + k \times \operatorname{lcm}(m_1, m_2)$，其中 $k \in \mathbb{Z}$.

然后就可以得出合并后的方程：$x \equiv x' \pmod{\operatorname{lcm}(m_1, m_2)}$.

> 如果你没看懂，可以再看看 rxj 的 https://www.luogu.com.cn/blog/blue/kuo-zhan-zhong-guo-sheng-yu-ding-li

代码（此处的乘法比较容易溢出，一般开大一点，`long long` 不行就 `int128`）：

```cpp
void merge(ll &a1, ll &m1, ll a2, ll m2)
{
    ll g = gcd(m1, m2), m = m1 / g * m2;

    ll p, q;
    exgcd(m1 / g, m2 / g, p, q);

    p = p * m1 % m;
    p = p * ((a2 - a1) / g) % m;

    a1 = (a1 + p + m) % m;
    m1 = m;
}
```

例题

题目：P4777 扩展中国剩余定理

▼ 点击查看代码

这道题很坑，数很大，我开到了 `int128` ...

```cpp
typedef __int128_t vl;

const int N = 1e5 + 10;

ll gcd(ll a, ll b) { return b ? gcd(b, a % b) : a; }

ll exgcd(ll a, ll b, vl &x, vl &y)
{
    if (b == 0)
    {
        x = 1, y = 0;
        return a;
    }
    ll d = exgcd(b, a % b, y, x);
    y -= a / b * x;
    return d;
}

void merge(ll &a1, ll &m1, ll a2, ll m2)
{
    ll g = gcd(m1, m2), m = m1 / g * m2;

    vl p, q;
    exgcd(m1 / g, m2 / g, p, q);

    p = p * m1 % m;
    p = p * ((a2 - a1) / g) % m;

    a1 = (a1 + p + m) % m;
    m1 = m;
}

int main()
{
    int n = rr;

    ll mm = rr, aa = rr;
    for (int i = 1; i < n; ++i)
    {
        ll m = rr, a = rr;
```

```
        merge(aa, mm, a, m);
    }

    printf("%lld\n", aa % mm);
    return 0;
}
```

Reference

[1] https://oi-wiki.org/math/number-theory/crt/

[2] https://www.bilibili.com/video/BV1AN4y1N7Su/

[3] https://www.bilibili.com/video/BV1Ut4y1F7HG/

[4] https://numbermatics.com/n/999911658/

[5] https://www.luogu.com.cn/blog/blue/kuo-zhan-zhong-guo-sheng-yu-ding-li

---

合集：学习笔记

标签：学习笔记 ，算法