线性同余方程

本文讨论线性同余方程的求解。

基本概念

设 a, b, n 为整数，x 为未知数，那么，形如

$$ax \equiv b \pmod n$$

的方程称为 **线性同余方程**（linear congruence equation）。

求解线性同余方程，需要找到区间 $[0, n-1]$ 中 x 的全部解。当然，将它们加减 n 的任意倍数，依然是方程的解。在模 n 的意义下，这些就是该方程的全部解。

本文接下来介绍了两种求解线性同余方程的思路，分别利用了逆元和不定方程。对于一般的情形，逆元和不定方程的求解都需要用到 扩展欧几里得算法，因此，这两种思路其实是一致的。

用逆元求解

首先，考虑 a 和 n 互素的情形，即 $\gcd(a, n) = 1$ 的情形。此时，可以计算 a 的 逆元 a^{-1}，并将方程两边同乘以 a^{-1}，这就得到方程的唯一解：

$$x \equiv ba^{-1} \pmod n.$$

紧接着，考虑 a 和 n 不互素的情形，即 $\gcd(a, n) = d > 1$ 的情形。此时，原方程不一定有解。例如，$2x \equiv 1 \pmod 4$ 就没有解。因此，需要考虑两种情形：

- 当 d 不能整除 b 时，方程无解。对于任意的 x，方程左侧 ax 都是 d 的倍数，但是方程右侧 b 不是 d 的倍数。因此，它们不可能相差 n 的倍数，因为 n 的倍数也一定是 d 的倍数。因此，方程无解。
- 当 d 可以整除 b 时，可以将方程的参数 a, b, n 都同除以 d，得到一个新的方程：

  $$a'x \equiv b' \pmod{n'}.$$

  其中，$\gcd(a', n') = 1$，也就是说，a' 和 n' 互素。这种情形已经在前文解决，所以，可以通过求解逆元得到方程的一个解 x'.

  显然，x' 也是原方程的一个解。但这并非原方程唯一的解。由于转化后的方程的全体解为

  $$\{x' + kn' : k \in \mathbf{Z}\}.$$

  这些解中落在区间 $[0, n-1]$ 的那些，就是原方程在区间 $[0, n-1]$ 中的全部解：

  $$x \equiv (x' + kn') \pmod n, \quad k = 0, 1, \cdots, d-1.$$

总结这两种情形，线性同余方程的 **解的数量** 等于 $d = \gcd(a, n)$ 或 0。

用不定方程求解

线性同余方程等价于关于 x, y 的 二元一次不定方程：

$$ax + ny = b.$$

利用所引页面的讨论，方程有解当且仅当 $\gcd(a, n) \mid b$，而且该方程的一组通解是

$$\begin{aligned} x &= x_0 + t\frac{n}{d}, \\ y &= y_0 - t\frac{a}{d}, \end{aligned}$$

其中，$d = \gcd(a, n)$ 是它们的最大公约数，t 是任意整数。

进而，线性同余方程的通解就是

$$x \equiv \left(x_0 + t\frac{n}{d}\right) \pmod{n}, \quad t \in \mathbf{Z}.$$

将 x_0 对 n/d 取模就得到同余方程的最小（非负）整数解，也就是上文的 x'.

参考实现

本节提供的参考实现可以得到同余方程的最小非负整数解。如果解不存在，则输出 -1。

参考实现

C++

```cpp
// Extended Euclidean Algorithm.
// Finds integers x, y such that a*x + b*y = gcd(a, b),
// and returns gcd(a, b).
int ex_gcd(int a, int b, int& x, int& y) {
  if (!b) {
    x = 1;
    y = 0;
    return a;
  } else {
    int d = ex_gcd(b, a % b, y, x);
    y -= a / b * x;
    return d;
  }
}

// Solves the linear congruence equation:
//     a * x ≡ b (mod n), where n > 0.
// Returns the smallest non-negative solution x,
// or -1 if there is no solution.
int solve_linear_congruence_equation(int a, int b, int n) {
  int x, y;
  int d = ex_gcd(a, n, x, y);
  if (b % d) return -1;
  n /= d;
  x *= b / d;
  return (x % n + n) % n;
}
```

Python

```python
def ex_gcd(a, b):
    """
    Extended Euclidean Algorithm.
    Finds integers x, y such that a*x + b*y = gcd(a, b),
    and returns (gcd, x, y).
    """
    if b == 0:
        return a, 1, 0
    d, x1, y1 = ex_gcd(b, a % b)
    x = y1
    y = x1 - (a // b) * y1
    return d, x, y


def solve_linear_congruence_equation(a, b, n):
    """
    Solves the linear congruence equation:
```

```
        a * x ≡ b (mod n), where n > 0.
    Returns the smallest non-negative solution x,
    or -1 if there is no solution.
    """
    d, x, y = ex_gcd(a, n)
    if b % d != 0:
        return -1
    n //= d
    x *= b // d
    return (x % n + n) % n
```

习题

- 「NOIP2012」同余方程

本页面主要译自博文 Модульное линейное уравнение первого порядка 与其英文翻译版 Linear Congruence Equation。其中俄文版版权协议为 Public Domain + Leave a Link；英文版版权协议为 CC-BY-SA 4.0。内容有改动。

本页面最近更新：2025/8/11 22:43:39，更新历史

发现错误？想一起完善？ 在 GitHub 上编辑此页！

本页面贡献者：Ir1d, Enter-tainer, MegaOwIer, Tiphereth-A, Xeonacid, Great-designer, Haohu Shen, iamtwz, ksyx, kZime, ouuan, stevebraveman, aofall, c-forrest, CoelacanthusHex, leoleoasd, Marcythm, Menci, Persdre, Phemon, shawlleyw, shuzhouliu, sshwy, StudyingFather, tsentau