数论—欧几里得算法、裴蜀定理、扩展欧几里得算法

引入

最大公约数

最大公约数即为 Greatest Common Divisor，常缩写为 gcd。

一组整数的公约数，是指同时是这组数中每一个数的约数的数。± 1 是任意一组整数的公约数；
一组整数的最大公约数，是指所有公约数里面最大的一个。

特殊的，我们定义 $\gcd(a,0)=a$。

最小公倍数

最小公倍数即为 Least Common Multiple，常缩写为 lcm。

一组整数的公倍数，是指同时是这组数中每一个数的倍数的数。0 是任意一组整数的公倍数；
一组整数的最小公倍数（Least Common Multiple, LCM），是指所有正的公倍数里面，最小的一个数。

互质

如果两个数 a 和 b 满足 $\gcd(a,b)=1$，我们称 a 和 b 互质。

欧几里得算法

欧几里得算法（Euclidean algorithm），是求解两个数最大公约数的最常用的算法。

算法思想

$\gcd(a,b)=\gcd(b,a \bmod b)$

具体证明见：OI-Wiki。

代码

```
int gcd(int a, int b) { return b == 0 ? a : gcd(b, a % b); }
```

因此也有递归写法：

```
int gcd(int a, int b) {
    int tmp;
    while (b != 0) tmp = a, a = b, b = tmp % b;
    return a;
}
```

对于 C++14，我们可以使用 中的 `__gcd(a,b)` 函数来求最大公约数。

时间复杂度

在输入为两个长为 n 的二进制整数时，欧几里得算法的时间复杂度为 $O(n)$；
换句话说，在默认 a, b 同阶的情况下，时间复杂度为 $O(\log\max(a,b))$。

欧几里得算法的最劣时间复杂度情况是 $\gcd(\mathrm{Fib}_{n+1}, \mathrm{Fib}_n)$，其时间复杂度为 $O(n)$；
但是，有 $\gcd(\mathrm{Fib}_{n+1}, \mathrm{Fib}_n) = \mathrm{Fib}_{\gcd(n+1,n)}$。

最小公倍数

计算

$\gcd(a,b) \times \operatorname{lcm}(a,b) = a \times b$。

要求两个数的最小公倍数，先求出最大公约数即可。

证明

设 $a = p_1^{k_{a_1}} p_2^{k_{a_2}} \dots p_s^{k_{a_s}}$，$b = p_1^{k_{b_1}} p_2^{k_{b_2}} \dots p_s^{k_{b_s}}$。

我们发现，对于 a 和 b 的情况，二者的最大公约数等于
$p_1^{\min(k_{a_1},k_{b_1})} p_2^{\min(k_{a_2},k_{b_2})} \dots p_s^{\min(k_{a_s},k_{b_s})}$。

最小公倍数等于 $p_1^{\max(k_{a_1},k_{b_1})} p_2^{\max(k_{a_2},k_{b_2})} \dots p_s^{\max(k_{a_s},k_{b_s})}$。

由于 $k_a + k_b = \max(k_a, k_b) + \min(k_a, k_b)$，
所以得到结论是 $\gcd(a, b) \times \operatorname{lcm}(a, b) = a \times b$。

裴蜀定理

定义

若 a、b 是不全为零的整数，则存在整数 x、y，使得 $ax + by = \gcd(a, b)$。

推广

若 $A[1 \sim n]$ 是非零整数序列，则整数序列 $X[1 \sim n]$ 一定满足：

$\sum\limits_{i=1}^{n} A_i X_i = k \times \gcd(A_1, A_2, \ldots, A_n)$，其中 k 为正整数。

扩展欧几里得算法

扩展欧几里得算法（Extended Euclidean algorithm，EXGCD），常用于求 $ax + by = \gcd(a, b)$ 的一组可行解。

算法思路

对于 $ax + by = \gcd(a, b)$，考虑与欧几里得算法相似的思路：

	结论：
求一组解 x'、y'，使得	$bx' + (a \bmod b)y' = \gcd(b, a \bmod b)$
（欧几里得定理）$\gcd(a, b) = \gcd(b, a \bmod b)$	$bx' + (a \bmod b)y' = \gcd(a, b)$
（模运算的定义）$a \bmod b = a - \lfloor \frac{a}{b} \rfloor \times b$	$bx' + (a - \lfloor \frac{a}{b} \rfloor \times b)y' = \gcd(a, b)$
整理，得	$ay' + b(x' - \lfloor \frac{a}{b} \rfloor \times y') = \gcd(a, b)$

我们要求一组解，使得 $ax + by = \gcd(a, b)$

因此有一组解为 $\begin{cases} x = y' \\ y = x' - \lfloor \frac{a}{b} \rfloor \times y' \end{cases}$

其边界值为 $b = 0$，这时有 $ax = \gcd(a, 0) = a$，既有 $x = 1$；为了方便起见，我们取 $y = 0$。

即：若 $b = 0$，则取 $\begin{cases} x = 1 \\ y = 0 \end{cases}$

代码

来自 `OI-Wiki`：

```cpp
int Exgcd(int a, int b, int &x, int &y) {
    if (!b) {
        x = 1;
        y = 0;
        return a;
    }
    int d = Exgcd(b, a % b, x, y);
    int t = x;
    x = y;
    y = t - (a / b) * y;
    return d;
}
```

简化后可以写作：

```cpp
int Exgcd(int a, int b, int &x, int &y) {
    if (!b) {
        x = 1, y = 0;
        return a;
    }
    int d = Exgcd(b, a % b, y, x);
    y -= a / b * x;
    return d;
}
```

特解到通解

假设我们现在求出了一组特解 x_0、y_0，使得 $ax_0 + by_0 = \gcd(a, b)$

接下来：

$$
\begin{aligned}
ax_0+by_0 &= \gcd(a,b) \\
(ax_0+H)+(by_0-H) &= \gcd(a,b) \\
a(x_0+H/a)+b(y_0-H/b) &= \gcd(a,b)
\end{aligned}
$$

可以看出 H 即是 a 的倍数，又是 b 的倍数，

所以 $H=k\times \operatorname{lcm}(a,b)$，其中 k 可以是任意整数。

即：$\begin{cases} x=x_0+k\times \dfrac{\operatorname{lcm}(a,b)}{a} \\ y=y_0+k\times \dfrac{\operatorname{lcm}(a,b)}{b} \end{cases}$ ，其中 $k\in\mathbb{Z}$。

Reference

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

[2] https://oi-wiki.org/math/number-theory/gcd/

合集：学习笔记

标签：算法 ， 学习笔记