内容：

- 算法
- 实现
- 正确性证明
- 时间复杂度
- 最小公倍数
- 练习题

给定两个非负整数 a 和 b，我们要求它们的 **GCD**（最大公约数），也就是所有能整除 a 和 b 的数中最大的一个，通常记作 $gcd(a, b)$，公式定义如下：

$$gcd(a, b) = \max_{k=1...\infty:\ k|a \wedge k|b} k$$

（这里的 `|` 符号代表整除）

当 a, b 中一个数为0且另一个不为0的时候，它们的最大公约数根据定义就是不为0的那个数。当两个数都是0的时候，求最大公约数问题不是良定义的（可以是任意大的数），我们规定结果是0。因此我们有结论，当其中有一个数为0时，两个数的最大公约数是另一个数

下面讨论的欧几里得算法（Euclidean algorithm）能够在 $O(log(min(a, b)))$ 的时间复杂度内求出两个数 a 和 b 的最大公约数。该算法最早在欧几里得的 `Elements` 中被描述（大约公元前300年），实际的发明时间可能更早

算法

算法内容非常简洁：

$$gcd(a, b) = \begin{cases} a & if\ \ b == 0 \\ gcd(b, a\ mod\ b) & otherwise. \end{cases}$$

实现

```
int gcd (int a, int b) {
    if (b == 0)
        return a;
    else
        return gcd (b, a % b);
}
```

用c++的三元运算符，可以写成：

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

非递归实现如下：

```
int gcd (int a, int b) {
    while (b) {
        a %= b;
        swap(a, b);
    }
    return a;
}
```

译者注：在 c++17 中头文件 `<numberic>` 已经自带了 `std::gcd` 函数，接受两个整形参数，返回它们绝对值的最大公约数

正确性证明

首先注意到在欧几里得算法的每次迭代中第二个参数都会严格递减（并且第二个参数是非负的），因此该算法一定会终止。

证明正确性需要确认对所有的 $a > 0, b \geq 0$ 都有 $gcd(a, b) = gcd(b, a \; mod \; b)$

证明方法是证明左式整除右式，同时右式整除左式，这显然要求左右两式相等。我们先定义 $d = gcd(a, b)$，那么根据最大公约数的定义有，$d \mid a$，$d \mid b$

我们按下式展开取模操作：

$$a \; mod \; b = a - b \cdot \lfloor \frac{a}{b} \rfloor$$

显然取模的结果也能被 d 整除，也就是说我们有：

$$d \mid b$$
$$d \mid (a \; mod \; b)$$

对于任意三个数 p, q, r，如果 $p \mid q$，$p \mid r$，那么 $p \mid gcd(q, r)$。因此我们得到：

$$d = gcd(a, b) \mid gcd(b, a \; mod \; b)$$

也就是左式整除右式，同理我们可以得到右式整除左式，从而得证

时间复杂度

欧几里得算法的时间复杂度可以有 $Lamé$ 定理得到，它展示了欧几里得算法和斐波那契数之间的联系：

如果 $a > b \geq 1$，$b < F_n$，那么欧几里得算法最多运行 $n - 2$ 步

不仅如此，这个上界还是严格的，当 $a = F_n$，$b = F_{n-1}$ 的时候，算法就需要运行 $n-2$ 次。换言之，对于欧几里得算法，最耗时的输入就是相邻的两个斐波那契数

因为斐波那契数是指数上涨的，因此欧几里得算法的复杂度是 $O(log\, min(a, b))$

最小公倍数

求两个数的最小公倍 **LCM**（least common multiple）可以由以下公式化归为求最大公约数问题：

$$lcm(a, b) = \frac{a \cdot b}{gcd(a, b)}$$

因此，最大公约数可以在相同的时间复杂度内求出，下面的实现比较仔细，预防了 $a \cot b$ 时的溢出情况：

```
int lcm (int a, int b) {
    return a / gcd(a, b) * b;
}
```

练习题

- ✅ [Codechef - GCD and LCM]()