结果对 x 取余

很多题目会有：结果对 100007 取余，结果对 10^9+7 取余，之类的要求。一般都是因为结果可能是一个很大的数，超过了 `long long` 的范围，要得到精确的值需要使用高精度计算的算法。为了避免高精度计算带来的时间复杂度和冗长的代码，出题人便要求结果对某个值取余，取余后的结果一致就认为算法正确。

例题 1

题面

输出 2^{100}，结果对 10^9+7 取余。

常见错解

```
...
  int ans = 1;
  for(int i = 1; i <= 100; i++)
    ans *= 2;
  cout << ans % 1000000007 << '\n';
...
```

这个做法虽然结果确实取余了，但是在中途计算时就会超过 int 的上限，做完循环后并不是准确的结果，最后再取余必然也不是正确的结果。

正解

```
...
  int ans = 1;
  for(int i = 1; i <= 100; i++)
    ans = (ans * 2) % 1000000007;
  cout << ans << '\n';
...
```

取余运算规则

- `(a * b * c) % x` 等价于 `(((a % x) * b) % x) * c) % x`
- `(a + b + c) % x` 等价于 `(((a % x) + b) % x) + c) % x`

- `(a - b) % x` 等价于 `(a % x - b % x + x) % x`
- `(a / b) % x` 等价于 `(a % x) * (b 的乘法逆元) % x`

简单来说，如果要求结果对 x 取余，只需要主要三点：

- 每次加法与乘法后都要立马做一次取余
- 减法的结果如果是负数，需要通过 `+x` 变成正数后取余。
- 除法运算需要使用乘法逆元进行处理。（一般只有提高组会遇到）

避免除法取余

普及组阶段常见的涉及到除法的一般只有排列组合数：

- $A_5^2 = \frac{5!}{2!} = \frac{5\times4\times3\times2\times1}{2\times1}$：提前考虑约分，$A_x^y = x \times (x-1) \times \cdots \times (y+1)$
- $C_5^2 = \frac{5!}{2!(5-2)!} = \frac{5\times4\times3\times2\times1}{(2\times1)\times(3\times2\times1)}$：使用杨辉三角优化，杨辉三角的计算只需要用加法。

杨辉三角对应的组合数：

$$
\begin{gathered}
C_0^0 \\
C_1^0, C_1^1 \\
C_2^0, C_2^1, C_2^2 \\
C_3^0, C_3^1, C_3^2, C_3^3 \\
C_4^0, C_4^1, C_4^2, C_4^3, C_4^4
\end{gathered}
$$