整除与同余

1 带余除法

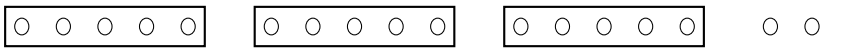
对于整数 a 和正整数 b，可以找到唯一的一组整数 q, r，使得

$$\boxed{a = bq + r, \qquad 0 \le r < b.}$$

其中，a 是被除数，b 是除数，q 是商，r 是余数。

例如，将 17 个物品每 5 个分成一组，可以分成 3 组，还剩 2 个。因此

$$17 = 5 \times 3 + 2.$$



一组 5 个　　一组 5 个　　一组 5 个　　剩 2 个

余数一定小于除数。如果还剩 5 个，就可以再分成一组，不能将 5 作为除以 5 的余数。

1.1 C++ 中的除法与取余

当 a 为非负整数、b 为正整数时：

```
int q=a/b; // 整数商
int r=a%b; // 余数
```

运算	商	余数	对应关系
17 除以 5	3	2	$17 = 5 \times 3 + 2$
20 除以 4	5	0	$20 = 4 \times 5$
3 除以 5	0	3	$3 = 5 \times 0 + 3$
0 除以 7	0	0	$0 = 7 \times 0$

整数相除时，`17/5` 的结果是 3，不是 3.4。除数不能为 0，`/0` 和 `%0` 都是非法运算。

2 整除

如果 a 除以 b 的余数为 0，就称 a 能被 b 整除，也称 b 整除 a，记作

$$b \mid a.$$

例如 $3 \mid 12$，因为 $12 = 3 \times 4$；但 5 不能整除 12。

```
if(a%b==0){
    // a能被b整除
}
```

这里 b 不为 0。特别地，0 能被任意非零整数整除，任意整数都能被 1 整除。

2.1 常见的整除特征

除数	判断方法
2	个位是 $0, 2, 4, 6, 8$。
3	各位数字之和能被 3 整除。
5	个位是 0 或 5。
9	各位数字之和能被 9 整除。
10	个位是 0。

例如 123 的各位数字之和是 $1 + 2 + 3 = 6$，所以 123 能被 3 整除。

2.2 同时满足多个整除条件

同时被 3 和 5 整除，需要两个条件都成立：

```
if(n%3==0 && n%5==0){
    // 两个条件都满足
}
```

由于 3 和 5 的最小公倍数为 15，也可以判断 `n%15==0`。

一般来说，同时被两个正整数整除，等价于被它们的最小公倍数整除。不能总把两个除数直接相乘：例如同时被 4 和 6 整除，只要求是 12 的倍数，不要求是 24 的倍数。

3 同余

两个整数除以同一个正整数 m，如果标准余数相同，就称它们模 m 同余，记作

$$a \equiv b \pmod{m}.$$

例如 17 和 7 除以 5 的余数都是 2，因此

$$17 \equiv 7 \pmod{5}.$$

同余的两个数不一定相等，但它们的差一定是 m 的倍数。上例中，$17 - 7 = 10$ 能被 5 整除。

3.1 按余数分类

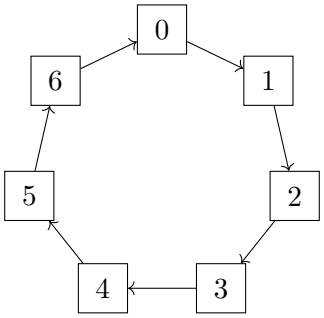
模 5 时，标准余数只有 $0, 1, 2, 3, 4$ 五种。

余数	对应的非负整数
0	$0, 5, 10, 15, 20, \ldots$
1	$1, 6, 11, 16, 21, \ldots$
2	$2, 7, 12, 17, 22, \ldots$
3	$3, 8, 13, 18, 23, \ldots$
4	$4, 9, 14, 19, 24, \ldots$

所以，对 m 取模后，只需要在 0 到 $m-1$ 之间记录结果。

3.2 周期与余数

每过 7 天，星期重复一次，因此经过 d 天与经过 $d \bmod 7$ 天，星期的变化相同。



周期为 m、编号从 0 开始时，向后移动 d 步，可以用 $(p+d) \bmod m$ 表示新位置。编号若从 1 开始，则先减 1，计算后再加 1。

4 负数的取余

C++ 的整数除法向 0 截断，% 的非零结果与被除数同号。

a	a/5	a%5	对应关系
7	1	2	$7 = 5 \times 1 + 2$
-7	-1	-2	$-7 = 5 \times (-1) - 2$
-10	-2	0	$-10 = 5 \times (-2)$

因此，C++ 的 -7%5 得到 -2。但要求结果在 0 到 4 之间时，-7 模 5 的标准余数应为 3：

$$-7 = 5 \times (-2) + 3.$$

4.1 将负余数调整为标准余数

先取余。如果结果小于 0，再加一次模数即可。

```
long long r=a%m;
if(r<0) r+=m;
```

例如 -7 对 5，先得到 -2，加 5 后得到 3；-10 对 5 已经得到 0，不再增加。

也常写成

$$\boxed{((a \bmod m) + m) \bmod m.}$$

```
long long r=(a%m+m)%m;
```

以上要求 $m > 0$。第二种写法还要保证中间的加法不会溢出；面对接近类型上限的模数时，前一种分支写法更稳妥。

4.2 例子

a	m	C++ 取余结果	标准余数
-1	7	-1	6
-8	7	-1	6
-14	7	0	0
0	7	0	0

5 取模的运算公式

以下均设 m 为正整数，$a \bmod m$ 表示非负的标准余数。

5.1 加法

$$\boxed{(a+b) \bmod m = ((a \bmod m) + (b \bmod m)) \bmod m.}$$

例如模 5 时：

$$(17+9) \bmod 5 = 26 \bmod 5 = 1,$$
$$(2+4) \bmod 5 = 6 \bmod 5 = 1.$$

也就是说，可以先把每个数取模，再相加，最后再取模。

5.2 减法

$$\boxed{(a-b) \bmod m = ((a \bmod m) - (b \bmod m)) \bmod m.}$$

C++ 中减法可能得到负数，因此还要调整结果：

```
long long r=(a%m-b%m)%m;
if(r<0) r+=m;
```

例如 $(7-9) \bmod 5 = 3$。如果只算 C++ 中的 `(7-9)%5`，得到的却是 -2。

5.3 乘法

$$\boxed{(ab) \bmod m = ((a \bmod m)(b \bmod m)) \bmod m.}$$

例如：

$$(17 \times 9) \bmod 5 = 153 \bmod 5 = 3,$$
$$(2 \times 4) \bmod 5 = 8 \bmod 5 = 3.$$

以上性质都来自同一个事实：去掉一个数中整倍数的 m，不会改变它对 m 的余数。

6 连续运算时取模

6.1 多个数相加

加法可以逐项取模，不必先算出完整总和。下面以非负整数为例：

```
long long sum=0;
for(int i=1;i<=n;i++){
    long long x;
    cin>>x;
    sum=(sum+x%mod)%mod;
}
```

取模后，sum 始终小于 mod。即使原来的总和超出整数类型范围，也可以只保存它的余数。

若输入可能为负数，可以先将每个数调整为标准余数：

```
long long r=x%mod;
if(r<0) r+=mod;
sum=(sum+r)%mod;
```

这里的加法仍要在所用类型的范围内。当前章节的求和题模数为 114514，两份余数相加不会超过 `long long` 的范围。

6.2 多个数相乘

```
long long ans=1;
for(int i=1;i<=n;i++){
    ans=ans*i%mod;
}
```

这段代码计算 $n!\bmod mod$。阶乘的初值是 1，因为 1 乘任何数都不改变它；如果初值为 0，后面就始终为 0。

6.3 幂的取模

对于非负整数 k：

$$a^k \bmod m = (a \bmod m)^k \bmod m.$$

按顺序乘 k 次，每次相乘后取模，就能得到幂的余数，不需要保存完整的 a^k。

```
long long ans=1%mod;
for(int i=1;i<=k;i++){
    ans=ans*(a%mod)%mod;
}
```

当 $k=0$ 时不进入循环，结果为 $1 \bmod m$。这里只介绍直接连乘，指数更大时可再学习快速幂。

7 取模时需要注意的地方

7.1 最后取模，不能挽回前面的溢出

```
int a=1000000000,b=1000000000;
long long r=1LL*a*b%1000000007;
```

1LL 让乘法按 long long 计算。如果先用两个 int 相乘，乘法就可能已经溢出，之后再赋给 long long 也没有用。

提前取模后，两个余数相乘仍然可能很大。使用乘法公式时，还要确保**余数的乘积**在计算类型的范围内，不能认为写了取模就一定安全。

7.2 除法不能直接照搬加减乘的公式

通常不能写成

$$(a/b) \bmod m \overset{\text{一般不成立}}{=} ((a \bmod m)/(b \bmod m)) \bmod m.$$

例如，$12/3 = 4$，对 5 取模仍为 4；但先取模后，C++ 的整数除法变成 $2/3 = 0$，结果不同。

因此，**加、减、乘可以按前面的公式处理，除法不能直接这样处理**。模意义下的除法需要额外条件，本节暂不展开。

7.3 模数为 1

任何整数模 1 的标准余数都是 0。所有取模模板都应允许 $m = 1$，尤其注意乘法、幂的初始值。

7.4 余数相同不代表原数相同

2、7、12 模 5 的余数都是 2。取模后，整倍数部分已经被省略，不能只靠余数恢复原数。

8 公式与写法对照

内容	关系或写法
带余除法	$a = bq + r$，$0 \le r < b$，$b > 0$
整除	$a \bmod b = 0$，表示 a 能被 b 整除
同余	$a \equiv b \pmod m$，表示标准余数相同
标准余数	先计算 `r=a%m`；若小于 0，再加 m
加法取模	先取余，再相加，最后取余
减法取模	先取余，再相减，取余后调整负数
乘法取模	先取余，再相乘，最后取余；注意乘法范围
连续运算	每一步都可以取模，不必保留完整结果

8.1 配套训练

算法竞赛入门指南：第三章“整除与同余”

题号	题目
P1210	带余除法
P1237	判断一个数能否同时被 3 和 5 整除
P1238	判断能否被 3、5、7 整除
P1890	计算星期几
P1579	三倍数
P1269	计算星期几
P2241	负数取模
P2204	求和取模 1
P2205	求和取模 2
P2206	计算阶乘