# 数学基础

## 2025 暑期留校集训

**99_wood**

东北大学
计算机科学与工程学院

2025 年 8 月 14 日

# 开始之前

# 开始之前

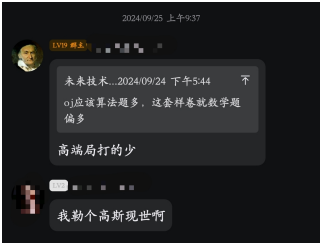
- 算法竞赛 $\neq$ 数学竞赛.

# 开始之前

- 算法竞赛 $\neq$ 数学竞赛.
- 但是随着你不断深入算法竞赛, 数学知识会变得越来越重要. 没有良好的数学素养, 很难在算法竞赛中取得好成绩.

# 开始之前

- 算法竞赛 $\neq$ 数学竞赛.
- 但是随着你不断深入算法竞赛, 数学知识会变得越来越重要. 没有良好的数学素养, 很难在算法竞赛中取得好成绩.
- 虽然在比赛当中你可以猜猜猜, 但是在你学习数学知识时, 还是要怀着严谨的心态. 多思考, 多感悟.



# 目录

1. 数学思想
2. 整除
3. 质数
4. 公因数与公倍数
5. 模意义下运算及扩展欧几里得算法
6. 中国剩余定理
7. 组合数学

# 数学思想总览

在算法竞赛中, 数学知识不仅仅是公式和定理, 更重要的是思想与方法. 常见的数学型思路包括:

1. 拆项与化简
2. 交换枚举顺序
3. 算贡献
4. 容斥原理
5. 转化思想
6. 分块思想
7. 分类讨论
8. 利用单调性
9. 前缀和与差分
10. 二进制分解

这些是构建数学解题思维的常用工具箱.

# 拆项与化简

## 核心思想

将复杂的表达式拆解为简单的可处理部分, 通过代数变形化简计算量.

# 拆项与化简

## 核心思想

将复杂的表达式拆解为简单的可处理部分, 通过代数变形化简计算量.

## 例

求和式 $\sum_{k=1}^{n} k(k+1)$.

# 拆项与化简

## 核心思想

将复杂的表达式拆解为简单的可处理部分, 通过代数变形化简计算量.

## 例

求和式 $\sum_{k=1}^{n} k(k+1)$.

$$\sum_{k=1}^{n} k^2 + k = \left(\sum_{k=1}^{n} k^2\right) + \left(\sum_{k=1}^{n} k\right) = \frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2}$$

利用公式直接求得结果, 避免逐项计算.

# 交换枚举顺序

**核心思想**

在双重或多重循环（或求和）中, 改变枚举顺序, 让复杂问题转化为易计算的形式.

# 交换枚举顺序

## 核心思想

在双重或多重循环（或求和）中, 改变枚举顺序, 让复杂问题转化为易计算的形式.

## 例

统计所有 $(i,j)$ 满足 $1 \le j \le i \le n$

$$\sum_{i=1}^{n}\sum_{j=1}^{i}\lfloor\frac{i}{j}\rfloor$$

# 交换枚举顺序

## 解

令 $l = \lfloor \frac{n-j+1}{j} \rfloor = \lfloor \frac{n+1}{j} \rfloor - 1$

$$\begin{aligned}
\sum_{i=1}^{n}\sum_{j=1}^{i}\lfloor\frac{i}{j}\rfloor &= \sum_{j=1}^{n}\sum_{i=j}^{n}\lfloor\frac{i}{j}\rfloor \\
&= \sum_{j=1}^{n}\left(\sum_{k=1}^{l} kj + (n-(l+1)j+1)(l+1)\right) \\
&= \sum_{j=1}^{n}\left(j\sum_{k=1}^{l} k + (n-(l+1)j+1)(l+1)\right) \\
&= \sum_{j=1}^{n}\left(j\frac{l(l+1)}{2} + (n-(l+1)j+1)(l+1)\right)
\end{aligned}$$

# 交换枚举顺序

## 解

$$\sum_{i=1}^{n}\sum_{j=1}^{i}\lfloor\frac{i}{j}\rfloor=\sum_{j=1}^{n}\left(j\frac{l(l+1)}{2}+(n-(l+1)j+1)(l+1)\right)$$

$$=\sum_{j=1}^{n}j\frac{l(l+1)}{2}+\sum_{j=1}^{n}(n-(l+1)j+1)(l+1)$$

这里就可以在 $O(n)$ 的时间复杂度内计算出结果.

如果你会数论分块, 还可以在 $O(\sqrt{n})$ 的时间复杂度内计算出结果.

# 算贡献

**核心思想**

不直接遍历所有组合, 而是固定一个元素, 计算它对整体答案的贡献.

# 算贡献

## 核心思想

不直接遍历所有组合, 而是固定一个元素, 计算它对整体答案的贡献.

## 例

求数组所有子段的和.

# 算贡献

## 核心思想

不直接遍历所有组合, 而是固定一个元素, 计算它对整体答案的贡献.

## 例

求数组所有子段的和.

固定 $a_k$, 它出现在多少个子段里? 左边可选 $k$ 种起点, 右边可选 $n-k+1$ 种终点, 总贡献:

$$a_k \times k \times (n-k+1)$$

总和为 $\sum_{k=1}^{n} a_k \cdot k \cdot (n-k+1)$.

# 快速幂

## 思路

利用二进制拆分指数, 将幂运算的时间复杂度从 $O(n)$ 降为 $O(\log n)$:

$$a^b = \begin{cases} (a^{b/2})^2, & b \text{ 为偶数} \\ a \times (a^{b-1}), & b \text{ 为奇数} \end{cases}$$

## 例

计算 $3^{13}$:

$$3^{13} = 3 \times (3^6)^2 = 3 \times (3^3)^4 = 3 \times (3 \times 3^2)^4$$

# 快速幂

## 代码

```
1 int qpow(int a, int b, int mod) {
2     int res = 1;
3     while (b) {
4         if (b & 1) res = 1ll * res * a % mod;
5         a = 1ll * a * a % mod;
6         b >>= 1;
7     }
8     return res;
9 }
```

# 整除

**定义**

对于 $d, a \in \mathbb{Z}, d \neq 0$, 如果存在 $k \in \mathbb{Z}$, 使得 $a = k \cdot d$, 则称 $d \mid a$ ($d$ 整除 $a$).

- $\mid$ 是整除符号.
- $d$ 是 $a$ 的约数, $a$ 是 $d$ 的倍数.

# 性质

整除关系有以下性质:

# 性质

整除关系有以下性质:

- 自反性: 对于任意整数 $n \neq 0$, 有 $n \mid n$.

# 性质

整除关系有以下性质:

- 自反性: 对于任意整数 $n \neq 0$, 有 $n \mid n$.
- 反对称性: 若有 $a \mid b$ 且 $|a| \neq |b|$, 则 $b \nmid a$.

# 性质

整除关系有以下性质:

- 自反性: 对于任意整数 $n \neq 0$, 有 $n \mid n$.
- 反对称性: 若有 $a \mid b$ 且 $|a| \neq |b|$, 则 $b \nmid a$.
- 传递性: 若有 $a \mid b, b \mid c$, 则 $a \mid c$.

# 性质

整除关系有以下性质:

- 自反性: 对于任意整数 $n \neq 0$, 有 $n \mid n$.
- 反对称性: 若有 $a \mid b$ 且 $|a| \neq |b|$, 则 $b \nmid a$.
- 传递性: 若有 $a \mid b, b \mid c$, 则 $a \mid c$.
- $\forall a \in \mathbb{Z} \wedge a \neq 0, a \mid 0$.

# 性质

整除关系有以下性质:

- 自反性: 对于任意整数 $n \neq 0$, 有 $n \mid n$.
- 反对称性: 若有 $a \mid b$ 且 $|a| \neq |b|$, 则 $b \nmid a$.
- 传递性: 若有 $a \mid b, b \mid c$, 则 $a \mid c$.
- $\forall a \in \mathbb{Z} \wedge a \neq 0, a \mid 0$.
- $\forall a \in \mathbb{Z}, 1 \mid a$.

# 性质

整除关系有以下性质:

- 自反性: 对于任意整数 $n \neq 0$, 有 $n \mid n$.
- 反对称性: 若有 $a \mid b$ 且 $|a| \neq |b|$, 则 $b \nmid a$.
- 传递性: 若有 $a \mid b, b \mid c$, 则 $a \mid c$.
- $\forall a \in \mathbb{Z} \wedge a \neq 0, a \mid 0$.
- $\forall a \in \mathbb{Z}, 1 \mid a$.
- $a \mid b, a \mid c \Leftrightarrow \forall x, y \in \mathbb{Z}, a \mid (bx + cy)$.

# 性质

整除关系有以下性质:

- 自反性: 对于任意整数 $n \neq 0$, 有 $n \mid n$.
- 反对称性: 若有 $a \mid b$ 且 $|a| \neq |b|$, 则 $b \nmid a$.
- 传递性: 若有 $a \mid b, b \mid c$, 则 $a \mid c$.
- $\forall a \in \mathbb{Z} \wedge a \neq 0, a \mid 0$.
- $\forall a \in \mathbb{Z}, 1 \mid a$.
- $a \mid b, a \mid c \Leftrightarrow \forall x, y \in \mathbb{Z}, a \mid (bx + cy)$.
- $m \neq 0, a \mid b \Leftrightarrow ma \mid mb$.

# 性质

整除关系有以下性质:

- 自反性: 对于任意整数 $n \neq 0$, 有 $n \mid n$.
- 反对称性: 若有 $a \mid b$ 且 $|a| \neq |b|$, 则 $b \nmid a$.
- 传递性: 若有 $a \mid b, b \mid c$, 则 $a \mid c$.
- $\forall a \in \mathbb{Z} \wedge a \neq 0, a \mid 0$.
- $\forall a \in \mathbb{Z}, 1 \mid a$.
- $a \mid b, a \mid c \Leftrightarrow \forall x, y \in \mathbb{Z}, a \mid (bx + cy)$.
- $m \neq 0, a \mid b \Leftrightarrow ma \mid mb$.

这说明自然数集上的整除关系是一个偏序关系.

# 约数

**定义**

如果 $a \mid b$, 那么 $a$ 是 $b$ 的约数, $b$ 是 $a$ 的倍数. 也称 $a$ 为 $b$ 的因子.

**推论**

任何数 $n$ 至少有两个因子: $1$ 和 $n$ 自身. 我们将它们称为 $n$ 的平凡因子.

# Quiz

**Quiz 1**

$[1, n]$ 的整数中, $k$ 的倍数有多少个?

# Quiz

**Quiz 1**

$[1, n]$ 的整数中, $k$ 的倍数有多少个?

**解**

$[1, n]$ 中 $k$ 的倍数一次为 $k, 2k, \ldots, mk$. 其中 $m = \lfloor \frac{n}{k} \rfloor$. 因此答案为 $\lfloor \frac{n}{k} \rfloor$.

# Quiz

## Quiz 2

如何计算 $[1, n]$ 中每个数的约数个数 $d(n)$？要求复杂度为 $\mathcal{O}(n \log n)$ 级别.

# Quiz

## Quiz 2

如何计算 $[1,n]$ 中每个数的约数个数 $d(n)$? 要求复杂度为 $\mathcal{O}(n\log n)$ 级别.

## 解

我们可以使用筛法来计算 $[1,n]$ 中每个数的约数个数 $d(n)$.
具体步骤如下:

- 初始化一个大小为 $n+1$ 的数组 $d$, 全部赋值为 $0$.
- 对于每个整数 $i\in[1,n]$, 将 $i$ 的所有倍数 $j$ 的约数个数 $d(j)$ 加 $1$.

这个复杂度为 $\sum_{i=1}^{n}\frac{n}{i}=n\sum_{i=1}^{n}\frac{1}{i}=\mathcal{O}(n\ln n)$ 的. (注意这个调和级数的求和技巧经常用到)
这样我们就可以在 $\mathcal{O}(n\ln n)$ 的时间复杂度内计算出每个数的约数个数.

# 质数

## 定义

大于 $1$ 的自然数中, 只有 $1$ 和它本身两个因子的数叫做质数.
反之, 如果一个数有超过两个因子, 则称它为合数.
$1$ **既不是质数也不是合数.**

## 例

$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, \ldots$ 等都是质数.

# Quiz

## Quiz 3

质数有无数个. 如何证明?

# Quiz

**Quiz 3**

质数有无数个. 如何证明?

**证明.**

考虑反证法:
假设质数是有限的, 则可以列出所有质数 $p_1, p_2, \ldots, p_n$.
考虑数 $N = p_1 p_2 \cdots p_n + 1$.
显然 $N$ 不是任何 $p_i$ 的倍数, 因此 $N$ 是质数. 这与假设矛盾, 所以质数有无数个. □

# 质数的分布

虽然我们目前没有摸清楚质数的具体分布情况, 但是我们能给出近似的:

设 $\pi(n)$ 为不超过 $n$ 的质数个数. 那么有:

$$\pi(n) \sim \frac{n}{\ln n}$$

# 判断质数

我们可以在 $\mathcal{O}(\sqrt{n})$ 的时间复杂度内判断 $n$ 是不是质数.

## 代码

```
bool isPrime(const int x) {
    if(x == 1) return false;
    for(int i = 2; i * i <= x; ++i){
        if(x % i == 0) return false;
    }
    return true;
}
```

# 找质数

**问题**

如何求出 $[1, n]$ 中的所有质数?

# 找质数

**问题**

如何求出 $[1, n]$ 中的所有质数?

朴素想法是逐个判断, 然而很不幸它的复杂度是

$$\sum_{i=1}^{n} \sqrt{i} = O(n\sqrt{n})$$

.

# 埃氏筛

# 埃氏筛

我们可以采用前面提到的筛法.

# 埃氏筛

我们可以采用前面提到的筛法.

筛法是一种高效的找出质数的方法, 最著名的就是**埃拉托斯特尼筛法**.

# 埃氏筛

我们可以采用前面提到的筛法.

筛法是一种高效的找出质数的方法, 最著名的就是**埃拉托斯特尼筛法**.

首先, 我们筛掉 $2$ 的倍数, 然后筛掉 $3$ 的倍数, 然后筛掉 $5$ 的倍数...

# 埃氏筛

我们可以采用前面提到的筛法.

筛法是一种高效的找出质数的方法, 最著名的就是**埃拉托斯特尼筛法**.

首先, 我们筛掉 $2$ 的倍数, 然后筛掉 $3$ 的倍数, 然后筛掉 $5$ 的倍数…

剩下来的数就是质数.

# 埃氏筛

## 代码

```
bool isPrime[MAXN];
std::vector<int> prime;
void sieve(const int n) {
    for(int i = 1; i <= n; ++i) isPrime[i] = true;
    isPrime[1] = false;
    for(int i = 2; i <= n; ++i){
        if(isPrime[i]){
            prime.push_back(i);
            for(int j = i; 1ll * j * i <= n; ++j){
                isPrime[j * i] = false;
            }
        }
    }
}
```

# 埃氏筛

埃氏筛的复杂度是 $\mathcal{O}(n \log \log n)$.

# 埃氏筛

埃氏筛的复杂度是 $\mathcal{O}(n \log \log n)$.

一个不太严谨的证明是我们之前提到的根据质数的分布规律,

$$T = \sum_{p \text{ 是质数}} \frac{n}{p} \sim \ln \ln n.$$

# 埃氏筛

埃氏筛的复杂度是 $\mathcal{O}(n \log \log n)$.

一个不太严谨的证明是我们之前提到的根据质数的分布规律,

$$T = \sum_{p \text{ 是质数}} \frac{n}{p} \sim \ln \ln n.$$

因此, 埃氏筛的总时间复杂度为

$$\mathcal{O}(n \ln \ln n).$$

# 埃氏筛

埃氏筛的复杂度是 $\mathcal{O}(n \log \log n)$.

一个不太严谨的证明是我们之前提到的根据质数的分布规律,

$$T = \sum_{p \text{ 是质数}} \frac{n}{p} \sim \ln \ln n.$$

因此, 埃氏筛的总时间复杂度为

$$\mathcal{O}(n \ln \ln n).$$

严谨的证明可以参考 OI-wiki.

# 欧拉筛

但是我们注意到, 一些数会被多次筛去. 比如 $2 \mid 6$ 并且 $3 \mid 6$, 所以 $6$ 会被 $2$ 和 $3$ 两次筛去.

# 欧拉筛

但是我们注意到, 一些数会被多次筛去. 比如 $2 \mid 6$ 并且 $3 \mid 6$, 所以 $6$ 会被 $2$ 和 $3$ 两次筛去.

为了避免这种情况, 我们可以在筛选时只考虑每个质数的最小倍数. 这就是**欧拉筛**.

# 欧拉筛

但是我们注意到, 一些数会被多次筛去. 比如 $2 \mid 6$ 并且 $3 \mid 6$, 所以 $6$ 会被 $2$ 和 $3$ 两次筛去.

为了避免这种情况, 我们可以在筛选时只考虑每个质数的最小倍数. 这就是**欧拉筛**.

显然每个数只会被它的最小质因子筛去一次. 这样, 我们就可以在 $\mathcal{O}(n)$ 的时间复杂度内找出所有质数.

# 欧拉筛

## 代码

```
std::vector<int> prime;
int d[MAXN];
void sieve(int n) {
    for(int i = 2; i <= n; ++i){
        if(d[i] == 0){
            prime.push_back(i);
            d[i] = i;
        }
        for(const int p : prime){
            if(p > d[i] || 1ll * p * i > n) break;
            d[p * i] = p;
        }
    }
}
```

# 唯一分解定理

每个数都可以拆成质数乘积的方式. 这个过程叫做质因数分解.

# 唯一分解定理

每个数都可以拆成质数乘积的方式. 这个过程叫做质因数分解.

**例**

$$5 = 5 = 5^1$$
$$15 = 3 \times 5 = 3^1 \times 5^1$$
$$36 = 2 \times 2 \times 3 \times 3 = 2^2 \times 3^2$$

# 唯一分解定理

每个数都可以拆成质数乘积的方式. 这个过程叫做质因数分解.

**例**

$$5 = 5 = 5^1$$
$$15 = 3 \times 5 = 3^1 \times 5^1$$
$$36 = 2 \times 2 \times 3 \times 3 = 2^2 \times 3^2$$

这样的分解方式是唯一的! 这就是**唯一分解定理**, 也叫**算术基本定理**.

# 唯一分解定理

每个数都可以拆成质数乘积的方式. 这个过程叫做质因数分解.

**例**

$$5 = 5 = 5^1$$
$$15 = 3 \times 5 = 3^1 \times 5^1$$
$$36 = 2 \times 2 \times 3 \times 3 = 2^2 \times 3^2$$

这样的分解方式是唯一的! 这就是**唯一分解定理**, 也叫**算术基本定理**.

分解某个数的质因数可以通过枚举所有小于等于 $\sqrt{n}$ 的数来实现, 复杂度为 $\mathcal{O}(\sqrt{n})$.

# Quiz

## Quiz 4

如果我们把 $A$ 分解成了 $2^{a_1}3^{a_2}5^{a_3}\cdots$，把 $B$ 分解成了 $2^{b_1}3^{b_2}5^{b_3}\cdots$.
那么 $A \times B$ 如何表示？如何判断 $A \mid B$ 是否成立？

# Quiz

## Quiz 4

如果我们把 $A$ 分解成了 $2^{a_1}3^{a_2}5^{a_3}\cdots$, 把 $B$ 分解成了 $2^{b_1}3^{b_2}5^{b_3}\cdots$.
那么 $A \times B$ 如何表示? 如何判断 $A \mid B$ 是否成立?

## 解

- $A \times B = 2^{a_1+b_1}3^{a_2+b_2}5^{a_3+b_3}\cdots$
- $A \mid B$ 当且仅当 $\forall i, a_i \leq b_i$.

# Quiz

**Quiz 5**

应用 Quiz 4 的结论. 给定 $n$ 的质因数分解形式 $2^{a_1}3^{a_2}5^{a_3}\cdots$ 如何快速求出 $d(n)$?

# Quiz

## Quiz 5

应用 Quiz 4 的结论. 给定 $n$ 的质因数分解形式 $2^{a_1}3^{a_2}5^{a_3}\cdots$ 如何快速求出 $d(n)$?

## 解

根据质因数分解的性质, 我们有

$$d(n) = (a_1 + 1)(a_2 + 1)(a_3 + 1)\cdots$$

因此, 我们只需要在分解时记录每个质因子的指数即可.

# 公因数与公倍数

**定义**

对于两个整数 $a, b$, 如果存在一个整数 $d$ 使得 $d \mid a$ 且 $d \mid b$, 则称 $d$ 是 $a$ 和 $b$ 的公因数. 最大的公因数称为 $\gcd(a, b)$, 也记为 $(a, b)$.

**定义**

对于两个整数 $a, b$, 如果存在一个整数 $m$ 使得 $a \mid m$ 且 $b \mid m$, 则称 $m$ 是 $a$ 和 $b$ 的公倍数. 最小的公倍数称为 $\mathrm{lcm}(a, b)$, 也记为 $[a, b]$.

# Quiz

## Quiz 6

如果我们把 $A$ 分解成了 $\prod p_i^{a_i}$, 把 $B$ 分解成了 $\prod p_i^{b_i}$.
$\gcd(A,B)$ 和 $\mathrm{lcm}(A,B)$ 的表达式是什么?

# Quiz

## Quiz 6

如果我们把 $A$ 分解成了 $\prod p_i^{a_i}$，把 $B$ 分解成了 $\prod p_i^{b_i}$.
$\gcd(A,B)$ 和 $\mathrm{lcm}(A,B)$ 的表达式是什么?

## 解

$$\gcd(A,B) = \prod p_i^{\min(a_i,b_i)}$$

$$\mathrm{lcm}(A,B) = \prod p_i^{\max(a_i,b_i)}$$

# 最大公因数与最小公倍数的性质

最大公因数与最小公倍数有以下性质:

# 最大公因数与最小公倍数的性质

最大公因数与最小公倍数有以下性质:

- $\forall d, d \mid a \wedge d \mid b \Leftrightarrow d \mid \gcd(a, b)$.

# 最大公因数与最小公倍数的性质

最大公因数与最小公倍数有以下性质:

- $\forall d, d \mid a \wedge d \mid b \Leftrightarrow d \mid \gcd(a, b)$.
- $\forall m, a \mid m \wedge b \mid m \Leftrightarrow \operatorname{lcm}(a, b) \mid m$.

# 最大公因数与最小公倍数的性质

最大公因数与最小公倍数有以下性质:

- $\forall d, d \mid a \wedge d \mid b \Leftrightarrow d \mid \gcd(a, b)$.
- $\forall m, a \mid m \wedge b \mid m \Leftrightarrow \operatorname{lcm}(a, b) \mid m$.
- $\gcd(a, b) \times \operatorname{lcm}(a, b) = a \times b$.

# 最大公因数与最小公倍数的性质

最大公因数与最小公倍数有以下性质:

- $\forall d, d \mid a \wedge d \mid b \Leftrightarrow d \mid \gcd(a, b)$.
- $\forall m, a \mid m \wedge b \mid m \Leftrightarrow \operatorname{lcm}(a, b) \mid m$.
- $\gcd(a, b) \times \operatorname{lcm}(a, b) = a \times b$.
- $\gcd(a, 0) = |a|$, $\operatorname{lcm}(a, 0) = 0$.

# 最大公因数与最小公倍数的性质

最大公因数与最小公倍数有以下性质:

- $\forall d, d \mid a \wedge d \mid b \Leftrightarrow d \mid \gcd(a, b)$.
- $\forall m, a \mid m \wedge b \mid m \Leftrightarrow \operatorname{lcm}(a, b) \mid m$.
- $\gcd(a, b) \times \operatorname{lcm}(a, b) = a \times b$.
- $\gcd(a, 0) = |a|$, $\operatorname{lcm}(a, 0) = 0$.
- $\gcd(a, 1) = 1$, $\operatorname{lcm}(a, 1) = |a|$.

# 最大公因数与最小公倍数的性质

最大公因数与最小公倍数有以下性质:

- $\forall d, d \mid a \wedge d \mid b \Leftrightarrow d \mid \gcd(a, b)$.
- $\forall m, a \mid m \wedge b \mid m \Leftrightarrow \operatorname{lcm}(a, b) \mid m$.
- $\gcd(a, b) \times \operatorname{lcm}(a, b) = a \times b$.
- $\gcd(a, 0) = |a|$, $\operatorname{lcm}(a, 0) = 0$.
- $\gcd(a, 1) = 1$, $\operatorname{lcm}(a, 1) = |a|$.
- $\gcd(ka, kb) = k \gcd(a, b)$.

# 最大公因数与最小公倍数的性质

最大公因数与最小公倍数有以下性质:

- $\forall d, d \mid a \wedge d \mid b \Leftrightarrow d \mid \gcd(a, b)$.
- $\forall m, a \mid m \wedge b \mid m \Leftrightarrow \operatorname{lcm}(a, b) \mid m$.
- $\gcd(a, b) \times \operatorname{lcm}(a, b) = a \times b$.
- $\gcd(a, 0) = |a|$, $\operatorname{lcm}(a, 0) = 0$.
- $\gcd(a, 1) = 1$, $\operatorname{lcm}(a, 1) = |a|$.
- $\gcd(ka, kb) = k \gcd(a, b)$.
- $\operatorname{lcm}(ka, kb) = k \operatorname{lcm}(a, b)$.

# 最大公因数与最小公倍数的性质

最大公因数与最小公倍数有以下性质:

- $\forall d, d \mid a \wedge d \mid b \Leftrightarrow d \mid \gcd(a, b)$.
- $\forall m, a \mid m \wedge b \mid m \Leftrightarrow \operatorname{lcm}(a, b) \mid m$.
- $\gcd(a, b) \times \operatorname{lcm}(a, b) = a \times b$.
- $\gcd(a, 0) = |a|$, $\operatorname{lcm}(a, 0) = 0$.
- $\gcd(a, 1) = 1$, $\operatorname{lcm}(a, 1) = |a|$.
- $\gcd(ka, kb) = k \gcd(a, b)$.
- $\operatorname{lcm}(ka, kb) = k \operatorname{lcm}(a, b)$.
- $\gcd(a, b) = \gcd(a, a \pm b)$.

# 最大公因数与最小公倍数的性质

最大公因数与最小公倍数有以下性质:

- $\forall d, d \mid a \wedge d \mid b \Leftrightarrow d \mid \gcd(a, b)$.
- $\forall m, a \mid m \wedge b \mid m \Leftrightarrow \operatorname{lcm}(a, b) \mid m$.
- $\gcd(a, b) \times \operatorname{lcm}(a, b) = a \times b$.
- $\gcd(a, 0) = |a|$, $\operatorname{lcm}(a, 0) = 0$.
- $\gcd(a, 1) = 1$, $\operatorname{lcm}(a, 1) = |a|$.
- $\gcd(ka, kb) = k \gcd(a, b)$.
- $\operatorname{lcm}(ka, kb) = k \operatorname{lcm}(a, b)$.
- $\gcd(a, b) = \gcd(a, a \pm b)$.
- $\gcd(a, b) = \gcd(b, a \bmod b) \quad (b \neq 0)$.

# 欧几里得算法

**Quiz 7**

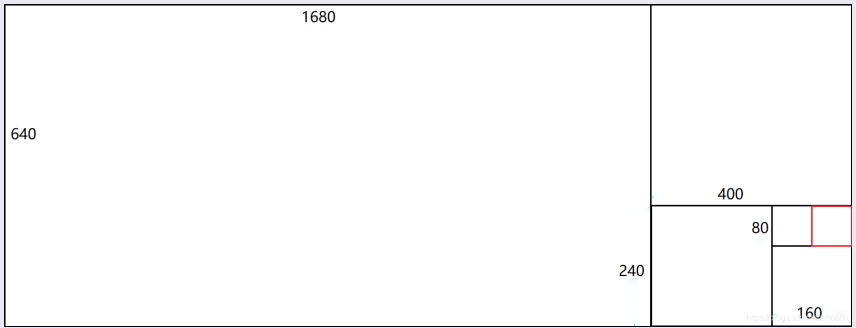
如何证明 $\gcd(a, b) = \gcd(b, a \bmod b) \quad (b \neq 0)$.

# 欧几里得算法

## Quiz 7

如何证明 $\gcd(a, b) = \gcd(b, a \bmod b) \quad (b \neq 0)$.

## 解



# 欧几里得算法

我们可以根据以上性质来快速求出 $\gcd(a, b)$.

## 代码

```
1 int gcd(int x, int y) {
2     return y == 0 ? x : gcd(y, x % y);
3 }
```

复杂度为 $\mathcal{O}(\log \min(a, b))$.

# 欧几里得算法

对于最小公倍数, 我们可以根据性质 $\gcd(a,b) \times \mathrm{lcm}(a,b) = a \times b$ 来求解.

## 代码

```
1 int lcm(int x, int y) {
2     return x / gcd(x, y) * y;
3 }
```

# 欧几里得算法

对于最小公倍数, 我们可以根据性质 $\gcd(a,b) \times \mathrm{lcm}(a,b) = a \times b$ 来求解.

## 代码

```
1 int lcm(int x, int y) {
2     return x / gcd(x, y) * y;
3 }
```

实际上, 对于 C++17, 我们可以使用 <numeric> 头文件中的 std::gcd 与 std::lcm 来求最大公约数和最小公倍数.

# 取模

虽然我们对于取模已经习以为常, 但是我们还是有必要明确它的定义.

# 取模

虽然我们对于取模已经习以为常, 但是我们还是有必要明确它的定义.

普遍地, 我们可以这样表达除法:

$$a = \lfloor \frac{a}{p} \rfloor \times p + a \bmod p$$

其中 $p$ 是除数, $\lfloor \frac{a}{p} \rfloor$ 是商, $a \bmod p$ 是余数.

这里的 $a \bmod p$ 就是我们常说的取模运算.

# 模运算的性质

值域: 由于模是取余, 所以 $a \bmod p$ 一定落在 $[0, p-1]$ 之间.

# 模运算的性质

值域: 由于模是取余, 所以 $a \bmod p$ 一定落在 $[0, p-1]$ 之间.

随时取模性质: 在**只含加法和乘法**的式子中, 如果最后的运算结果需要对 $p$ 取模, 那么我们可以在运算过程中随便取模. 只需要最后把结果对 $p$ 再取模, 答案就是正确的.

# 模运算的性质

值域: 由于模是取余, 所以 $a \bmod p$ 一定落在 $[0, p-1]$ 之间.

随时取模性质: 在**只含加法和乘法**的式子中, 如果最后的运算结果需要对 $p$ 取模, 那么我们可以在运算过程中随便取模. 只需要最后把结果对 $p$ 再取模, 答案就是正确的.

由于这两条好用的性质, 所以很多题目都会要求我们输出模意义的结果, 以此避免精度问题. 常用的模数有 $10^9 + 7$ 和 $998244353$. 这两个数都是质数.

# 乘法逆元

我们前面提到随时取模原理适用于**只含加法和乘法**的式子. 但是很多时候我们需要用到除法, 那么除法是否适用随时取模原理呢?

# 乘法逆元

我们前面提到随时取模原理适用于**只含加法和乘法**的式子. 但是很多时候我们需要用到除法, 那么除法是否适用随时取模原理呢?

答案是否定的. 因为即便取模后, 我们也无法保证模完的数能够整除分子.

# 乘法逆元

我们前面提到随时取模原理适用于**只含加法和乘法**的式子. 但是很多时候我们需要用到除法, 那么除法是否适用随时取模原理呢?

答案是否定的. 因为即便取模后, 我们也无法保证模完的数能够整除分子.

能否找到一种方法来表示模意义下的分数呢?

# 乘法逆元

我们前面提到随时取模原理适用于**只含加法和乘法**的式子. 但是很多时候我们需要用到除法, 那么除法是否适用随时取模原理呢?

答案是否定的. 因为即便取模后, 我们也无法保证模完的数能够整除分子.

能否找到一种方法来表示模意义下的分数呢?

这里我们引入乘法逆元的概念.

**定义**

在模 $p$ 意义下, 如果存在一个整数 $b$ 使得

$$a \times b \equiv 1 \pmod{p}.$$

这个 $b$ 就是 $a$ 的乘法逆元, 记作 $a^{-1}$.

# 乘法逆元

**定义**

在模 $p$ 意义下, 如果存在一个整数 $b$ 使得

$$a \times b \equiv 1 \pmod{p}.$$

这个 $b$ 就是 $a$ 的乘法逆元, 记作 $a^{-1}$.

# 乘法逆元

**定义**

在模 $p$ 意义下, 如果存在一个整数 $b$ 使得

$$a \times b \equiv 1 \pmod{p}.$$

这个 $b$ 就是 $a$ 的乘法逆元, 记作 $a^{-1}$.

我们发现, 这里的 $b$ 具有类似于 $\frac{1}{a}$ 的功能. 当我们需要计算 $\frac{c}{a} \pmod{p}$ 时, 可以将其转化为 $c \times b \pmod{p}$.

# 求逆元

求逆元主要有以下几种方法:

# 求逆元

求逆元主要有以下几种方法:

- 费马小定理: 当 $p$ 是质数时, 有 $a^{p-1} \equiv 1 \pmod{p}$, 因此 $a^{p-2} \equiv a^{-1} \pmod{p}$.

# 求逆元

求逆元主要有以下几种方法:

- 费马小定理: 当 $p$ 是质数时, 有 $a^{p-1} \equiv 1 \pmod{p}$, 因此 $a^{p-2} \equiv a^{-1} \pmod{p}$.
- 扩展欧几里得算法: 通过求解 $ax + py = 1$ 来得到 $b$.

# 求逆元

求逆元主要有以下几种方法:

- 费马小定理: 当 $p$ 是质数时, 有 $a^{p-1} \equiv 1 \pmod{p}$, 因此 $a^{p-2} \equiv a^{-1} \pmod{p}$.
- 扩展欧几里得算法: 通过求解 $ax + py = 1$ 来得到 $b$.
- 线性递推求逆元: 通过递推关系 $b_i = -\lfloor \frac{p}{i} \rfloor (p \bmod i)^{-1} \bmod p$ 来计算.

# 费马小定理求逆元

**定理**

**费马小定理** 当 $p$ 是质数时, 对任意整数 $a$ 有 $a^{p-1} \equiv 1 \pmod{p}$.

证明参考 OI-wiki.

# 费马小定理求逆元

**定理**

**费马小定理** 当 $p$ 是质数时, 对任意整数 $a$ 有 $a^{p-1} \equiv 1 \pmod{p}$.

证明参考 OI-wiki.

**推论**

当 $p$ 是质数时, 对任意整数 $a$ 有 $a^{p-2} \equiv a^{-1} \pmod{p}$.

# 费马小定理求逆元

**定理**

**费马小定理** 当 $p$ 是质数时, 对任意整数 $a$ 有 $a^{p-1} \equiv 1 \pmod{p}$.

证明参考 OI-wiki.

**推论**

当 $p$ 是质数时, 对任意整数 $a$ 有 $a^{p-2} \equiv a^{-1} \pmod{p}$.

由此我们可以快速求出 $a^{-1}$, 只需要计算 $a^{p-2} \bmod p$ 即可.

这部分可以用快速幂在 $\mathcal{O}(\log p)$ 的时间内完成.

# 线性递推求逆元

线性递推求逆元通过如下递推关系来在 $\mathcal{O}(p)$ 的时间内计算 $1, 2, \ldots, p$ 的逆元.

$$i^{-1} \equiv \begin{cases} 1, & \text{if } i = 1, \\ -\lfloor \frac{p}{i} \rfloor (p \bmod i)^{-1}, & \text{otherwise.} \end{cases} \pmod p$$

证明参考 OI-wiki.

## 代码

```
1 inv[1] = 1;
2 for (int i = 2; i <= n; ++i) {
3   inv[i] = (int)(p - p / i) * inv[p % i] % p;
4 }
```

# 扩展欧几里得算法 (exgcd)

**定义**

形如

$$ax + by = d,$$

其中 $a, b, d$ 是已知整数, $x, y$ 是未知整数的方程为**二元一次不定方程**.

为了求解二元一次不定方程, 我们需要学习扩展欧几里得算法.

# 扩展欧几里得算法 (exgcd)

**定理**

**裴蜀定理** 二元一次不定方程

$$ax + by = d$$

有解当且仅当 $\gcd(a, b) \mid d$.

**证明.**

# 扩展欧几里得算法 (exgcd)

**定理**

**裴蜀定理** 二元一次不定方程

$$ax + by = d$$

有解当且仅当 $\gcd(a, b) \mid d$.

**证明.**

必要性: 由于 $d$ 是 $a$ 和 $b$ 的线性组合, 所以 $\gcd(a, b)$ 必然整除 $d$.

# 扩展欧几里得算法 (exgcd)

**定理**

**裴蜀定理** 二元一次不定方程

$$ax + by = d$$

有解当且仅当 $\gcd(a, b) \mid d$.

**证明.**

必要性: 由于 $d$ 是 $a$ 和 $b$ 的线性组合, 所以 $\gcd(a, b)$ 必然整除 $d$.

充分性: 我们可以通过反复应用欧几里得算法来构造这样的 $x, y$. □

# 扩展欧几里得算法 (exgcd)

根据裴蜀定理, 我们只需要会求解

$$ax + by = 1 \quad (a, b \text{ 互质})$$

的整数解即可.

# 扩展欧几里得算法 (exgcd)

设

$$ax_1 + by_1 = \gcd(a, b)$$
$$bx_2 + (a \bmod b)y_2 = \gcd(b, a \bmod b)$$

# 扩展欧几里得算法 (exgcd)

设

$$ax_1 + by_1 = \gcd(a, b)$$
$$bx_2 + (a \bmod b)y_2 = \gcd(b, a \bmod b)$$

由欧几里得定理可知:

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

# 扩展欧几里得算法 (exgcd)

设

$$ax_1 + by_1 = \gcd(a, b)$$
$$bx_2 + (a \bmod b)y_2 = \gcd(b, a \bmod b)$$

由欧几里得定理可知:

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

所以

$$ax_1 + by_1 = bx_2 + (a \bmod b)y_2.$$

# 扩展欧几里得算法 (exgcd)

因为

$$a = \lfloor \frac{a}{b} \rfloor \times b + a \bmod b,$$

# 扩展欧几里得算法 (exgcd)

因为

$$a = \lfloor \frac{a}{b} \rfloor \times b + a \bmod b,$$

所以

$$\begin{aligned} ax_1 + by_1 &= (\lfloor \frac{a}{b} \rfloor \times b + a \bmod b)x_1 + by_1 \\ &= (\lfloor \frac{a}{b} \rfloor x_1 + y_1)b + (a \bmod b)x_1. \end{aligned}$$

# 扩展欧几里得算法 (exgcd)

所以

$$(\lfloor\frac{a}{b}\rfloor x_1 + y_1)b + (a \bmod b)x_1 = bx_2 + (a \bmod b)y_2.$$

# 扩展欧几里得算法 (exgcd)

所以

$$(\lfloor\frac{a}{b}\rfloor x_1 + y_1)b + (a \bmod b)x_1 = bx_2 + (a \bmod b)y_2.$$

对比等式两边可以得出

$$x_1 = y_2, y_1 = x_2 - \lfloor\frac{a}{b}\rfloor y_2$$

将 $x_2, y_2$ 不断代入求解直至 $b = 0$ 递归 $x = 1, y = 0$ 回去构造答案.

# 扩展欧几里得算法 (exgcd)

## 代码

```
1 int exgcd(int a, int b, int& x, int& y) {
2     if (!b) return x = 1, y = 0, a;
3     int r = exgcd(b, a % b, y, x);
4     y -= (a / b) * x;
5     return r;
6 }
```

时间复杂度: $\mathcal{O}(\log(\min(a,b)))$.

# 扩展欧几里得算法求逆元

- $ab = km + 1$, 扩展欧几里得算法 (exgcd).

# 扩展欧几里得算法求逆元

- $ab = km + 1$, 扩展欧几里得算法 (exgcd).
- 有逆元 (不定方程有解) 当且仅当 $(a, m) = 1$.

# 扩展欧几里得算法求逆元

- $ab = km + 1$, 扩展欧几里得算法 (exgcd).
- 有逆元 (不定方程有解) 当且仅当 $(a, m) = 1$.
- 如果 $m$ 为质数, 则任意 $0 < a < m$, $a$ 均有逆元.

# 中国剩余定理 (CRT)

**《孙子算经》**

今有物不知其数, 三三数之剩二, 五五数之剩三, 七七数之剩二, 问物几何?

# 中国剩余定理 (CRT)

**《孙子算经》**

今有物不知其数, 三三数之剩二, 五五数之剩三, 七七数之剩二, 问物几何?

- 答案: 23.

# 中国剩余定理 (CRT)

**《孙子算经》**

今有物不知其数, 三三数之剩二, 五五数之剩三, 七七数之剩二, 问物几何?

- 答案: 23.
- $x \equiv 23 \pmod{105}$.

# 中国剩余定理 (CRT)

## 中国剩余定理

方程组 $\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \vdots \\ x \equiv a_k \pmod{m_k} \end{cases}$ $\left(\forall i \neq j, (m_i, m_j) = 1\right)$ 的解

# 中国剩余定理 (CRT)

## 中国剩余定理

方程组 $\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \vdots \\ x \equiv a_k \pmod{m_k} \end{cases}$ $\left(\forall i \neq j, (m_i, m_j) = 1\right)$ 的解为

$$x \equiv \sum_{i=1}^{k} M_i' M_i a_i \pmod{M}$$

其中 $M = m_1 m_2 \cdots m_k$, $M_i = \dfrac{M}{m_i}$, $M_i' M_i \equiv 1 \pmod{m_i}$.

# 扩展中国剩余定理 (exCRT)

## 扩展中国剩余定理

求方程组 $\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \vdots \\ x \equiv a_k \pmod{m_k} \end{cases}$ 的解.

# 加法原理和乘法原理

- 加法原理: 做某件事情有几种选择, 每种选择的方案数之和就是做这件事情的方案数.

# 加法原理和乘法原理

- 加法原理: 做某件事情有几种选择, 每种选择的方案数之和就是做这件事情的方案数.
- 乘法原理: 做某件事情分为几步, 每步的方案数是独立的, 则它们的积就是做这件事情的方案数.

# 加法原理和乘法原理

- 加法原理: 做某件事情有几种选择, 每种选择的方案数之和就是做这件事情的方案数.
- 乘法原理: 做某件事情分为几步, 每步的方案数是独立的, 则它们的积就是做这件事情的方案数.

**Quiz 9**

求满足 $x + y \leq n$ 的正整数解的数量.

# 加法原理和乘法原理

- 加法原理: 做某件事情有几种选择, 每种选择的方案数之和就是做这件事情的方案数.
- 乘法原理: 做某件事情分为几步, 每步的方案数是独立的, 则它们的积就是做这件事情的方案数.

**Quiz 9**

求满足 $x + y \leq n$ 的正整数解的数量.

**Quiz 9**

证明: 因数个数公式: $d(n) = \prod_{i=1}^{k} (\alpha_i + 1)$.

# 排列数

## 排列

从 $n$ 个不同元素中取出 $m(m \leq n)$ 个元素, 按照一定的顺序排成一列, 叫做从 $n$ 个元素中取出 $m$ 个元素的一个排列. 所有不同的排列的个数称为**排列数**, 记作 $P_n^m$ 或 $A_n^m$. 特别地, 当 $m = n$ 时, 这个排列被称作**全排列**.

# 排列数

**排列**

从 $n$ 个不同元素中取出 $m(m \leq n)$ 个元素, 按照一定的顺序排成一列, 叫做从 $n$ 个元素中取出 $m$ 个元素的一个排列. 所有不同的排列的个数称为**排列数**, 记作 $P_n^m$ 或 $A_n^m$. 特别地, 当 $m = n$ 时, 这个排列被称作**全排列**.

- 下降幂: $n^{\underline{r}} = n(n-1)(n-2)\cdots(n-r+1)$, $n^{\underline{0}} = 1$.

# 排列数

## 排列

从 $n$ 个不同元素中取出 $m(m \leq n)$ 个元素, 按照一定的顺序排成一列, 叫做从 $n$ 个元素中取出 $m$ 个元素的一个排列. 所有不同的排列的个数称为**排列数**, 记作 $P_n^m$ 或 $A_n^m$. 特别地, 当 $m = n$ 时, 这个排列被称作**全排列**.

- 下降幂: $n^{\underline{r}} = n(n-1)(n-2)\cdots(n-r+1)$, $n^{\underline{0}} = 1$.
- 上升幂: $n^{\overline{r}} = n(n+1)(n+2)\cdots(n+r-1)$, $n^{\overline{0}} = 1$.

# 排列数

## 排列

从 $n$ 个不同元素中取出 $m(m \leq n)$ 个元素, 按照一定的顺序排成一列, 叫做从 $n$ 个元素中取出 $m$ 个元素的一个排列. 所有不同的排列的个数称为**排列数**, 记作 $P_n^m$ 或 $A_n^m$. 特别地, 当 $m = n$ 时, 这个排列被称作**全排列**.

- 下降幂: $n^{\underline{r}} = n(n-1)(n-2)\cdots(n-r+1)$, $n^{\underline{0}} = 1$.
- 上升幂: $n^{\overline{r}} = n(n+1)(n+2)\cdots(n+r-1)$, $n^{\overline{0}} = 1$.
- 阶乘: $n! = n(n-1)\cdots 1$, $0! = 1$.

# 排列数

**排列**

从 $n$ 个不同元素中取出 $m(m \leq n)$ 个元素, 按照一定的顺序排成一列, 叫做从 $n$ 个元素中取出 $m$ 个元素的一个排列. 所有不同的排列的个数称为**排列数**, 记作 $P_n^m$ 或 $A_n^m$. 特别地, 当 $m = n$ 时, 这个排列被称作**全排列**.

- 下降幂: $n^{\underline{r}} = n(n-1)(n-2)\cdots(n-r+1)$, $n^{\underline{0}} = 1$.
- 上升幂: $n^{\overline{r}} = n(n+1)(n+2)\cdots(n+r-1)$, $n^{\overline{0}} = 1$.
- 阶乘: $n! = n(n-1)\cdots 1$, $0! = 1$.
- 排列数: $A_n^m = \dfrac{n!}{(n-m)!} = n^{\underline{m}}$.

# 组合数

## 组合

从 $n$ 个不同的元素中取出 $m(m \leq n)$ 个元素为一组, 叫做从 $n$ 个元素中取出 $m$ 个元素的一个组合. 所有不同的组合的个数称为**组合数**, 记作 $C_n^m$ 或 $\binom{n}{m}$.

# 组合数

## 组合

从 $n$ 个不同的元素中取出 $m(m \leq n)$ 个元素为一组, 叫做从 $n$ 个元素中取出 $m$ 个元素的一个组合. 所有不同的组合的个数称为**组合数**, 记作 $C_n^m$ 或 $\binom{n}{m}$.

- 组合数: $\binom{n}{m} = \frac{n^{\underline{m}}}{m!} = \frac{n!}{(n-m)!m!}$.

# 组合数

## 组合

从 $n$ 个不同的元素中取出 $m(m \leq n)$ 个元素为一组, 叫做从 $n$ 个元素中取出 $m$ 个元素的一个组合. 所有不同的组合的个数称为**组合数**, 记作 $C_n^m$ 或 $\binom{n}{m}$.

- 组合数: $\binom{n}{m} = \frac{n^{\underline{m}}}{m!} = \frac{n!}{(n-m)!m!}$.
- $\binom{n}{m} = \binom{n-1}{m-1} + \binom{n-1}{m}$. (递推求组合数).

# 组合数

## 组合

从 $n$ 个不同的元素中取出 $m(m \leq n)$ 个元素为一组, 叫做从 $n$ 个元素中取出 $m$ 个元素的一个组合. 所有不同的组合的个数称为**组合数**, 记作 $C_n^m$ 或 $\binom{n}{m}$.

- 组合数: $\binom{n}{m} = \frac{n^{\underline{m}}}{m!} = \frac{n!}{(n-m)!m!}$.
- $\binom{n}{m} = \binom{n-1}{m-1} + \binom{n-1}{m}$. (递推求组合数).
- $\binom{n}{m} = \frac{n}{m}\binom{n-1}{m-1}$.

# 组合数

## 组合

从 $n$ 个不同的元素中取出 $m(m \leq n)$ 个元素为一组, 叫做从 $n$ 个元素中取出 $m$ 个元素的一个组合. 所有不同的组合的个数称为**组合数**, 记作 $C_n^m$ 或 $\binom{n}{m}$.

- 组合数: $\binom{n}{m} = \frac{n^{\underline{m}}}{m!} = \frac{n!}{(n-m)!m!}$.
- $\binom{n}{m} = \binom{n-1}{m-1} + \binom{n-1}{m}$. (递推求组合数).
- $\binom{n}{m} = \frac{n}{m}\binom{n-1}{m-1}$.
- $\binom{n}{m} = \frac{n-m+1}{m}\binom{n}{m-1}$.

# 组合数

$$\binom{n}{m} = \binom{n-1}{m-1} + \binom{n-1}{m}, \binom{n}{0} = 1$$

| $n\backslash k$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 1 | 3 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 1 | 4 | 6 | 4 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 1 | 5 | 10 | 10 | 5 | 1 | 0 | 0 | 0 | 0 | 0 |
| 6 | 1 | 6 | 15 | 20 | 15 | 6 | 1 | 0 | 0 | 0 | 0 |
| 7 | 1 | 7 | 21 | 35 | 35 | 21 | 7 | 1 | 0 | 0 | 0 |
| 8 | 1 | 8 | 28 | 56 | 70 | 56 | 28 | 8 | 1 | 0 | 0 |
| 9 | 1 | 9 | 36 | 84 | 126 | 126 | 84 | 36 | 9 | 1 | 0 |
| 10 | 1 | 10 | 45 | 120 | 210 | 252 | 210 | 120 | 45 | 10 | 1 |

# 组合数

**不定方程解的数量**

不定方程 $x_1 + x_2 + \cdots + x_k = n$ 的解的数量, 其中 $x_i$ 为整数, 且 $x_i \geqslant 1$.

# 组合数

**不定方程解的数量**

不定方程 $x_1 + x_2 + \cdots + x_k = n$ 的解的数量, 其中 $x_i$ 为整数, 且 $x_i \geqslant 1$.

- $\binom{n-1}{k-1}$.

# 组合数

## 不定方程解的数量

不定方程 $x_1 + x_2 + \cdots + x_k = n$ 的解的数量, 其中 $x_i$ 为整数, 且 $x_i \geqslant 1$.

- $\binom{n-1}{k-1}$.
- $x_i \geqslant a_i$ ?

# 组合数

## 不定方程解的数量

不定方程 $x_1 + x_2 + \cdots + x_k = n$ 的解的数量, 其中 $x_i$ 为整数, 且 $x_i \geqslant 1$.

- $\binom{n-1}{k-1}$.
- $x_i \geqslant a_i$ ?
- $x_1 + x_2 + \cdots + x_k \leq n$ ?

# 组合数

**网络路径计数问题**

在 $n \times m$ 的网格图上, 从 $(0,0)$ 走到 $(n,m)$, 每次只能向右走或向上走, 求方案数.

# 组合数

**网络路径计数问题**

在 $n \times m$ 的网格图上, 从 $(0,0)$ 走到 $(n,m)$, 每次只能向右走或向上走, 求方案数.

- 组合数学, $\binom{n+m}{n}$.

# 组合数

## 网络路径计数问题

在 $n \times m$ 的网格图上, 从 $(0,0)$ 走到 $(n,m)$, 每次只能向右走或向上走, 求方案数.

- 组合数学, $\binom{n+m}{n}$.
- 动态规划, `dp[i][j] = dp[i-1][j] + dp[i][j-1]`, 可以处理有障碍物的情况.

# 第二类斯特林数

## 第二类斯特林数

第二类斯特林数表示将 $n$ 个不同的小球, 放入 $k$ 个相同的盒子中, 每个盒子至少放 $1$ 个小球的不同的方案数, 记作 $S_2(n,k)$ 或 $\left\{ {n \atop k} \right\}$.

# 第二类斯特林数

## 第二类斯特林数

第二类斯特林数表示将 $n$ 个不同的小球, 放入 $k$ 个相同的盒子中, 每个盒子至少放 $1$ 个小球的不同的方案数, 记作 $S_2(n,k)$ 或 $\left\{ {n \atop k} \right\}$.

- 插入 $1$ 个小球时, 有两种方案:

# 第二类斯特林数

## 第二类斯特林数

第二类斯特林数表示将 $n$ 个不同的小球, 放入 $k$ 个相同的盒子中, 每个盒子至少放 $1$ 个小球的不同的方案数, 记作 $S_2(n,k)$ 或 $\left\{ {n \atop k} \right\}$.

- 插入 $1$ 个小球时, 有两种方案:
  1. 将小球单独放入一个空盒子中, 有 $\left\{ {n-1 \atop k-1} \right\}$ 种方案;

# 第二类斯特林数

## 第二类斯特林数

第二类斯特林数表示将 $n$ 个不同的小球, 放入 $k$ 个相同的盒子中, 每个盒子至少放 $1$ 个小球的不同的方案数, 记作 $S_2(n,k)$ 或 $\left\{ {n \atop k} \right\}$.

- 插入 $1$ 个小球时, 有两种方案:
  1. 将小球单独放入一个空盒子中, 有 $\left\{ {n-1 \atop k-1} \right\}$ 种方案;
  2. 将小球放入一个现有的非空盒子中, 有 $k \left\{ {n-1 \atop k} \right\}$ 种方案.

# 第二类斯特林数

## 第二类斯特林数

第二类斯特林数表示将 $n$ 个不同的小球, 放入 $k$ 个相同的盒子中, 每个盒子至少放 $1$ 个小球的不同的方案数, 记作 $S_2(n,k)$ 或 $\left\{ {n \atop k} \right\}$.

- 插入 $1$ 个小球时, 有两种方案:
  1. 将小球单独放入一个空盒子中, 有 $\left\{ {n-1 \atop k-1} \right\}$ 种方案;
  2. 将小球放入一个现有的非空盒子中, 有 $k \left\{ {n-1 \atop k} \right\}$ 种方案.
- 递推式: $\left\{ {n \atop k} \right\} = \left\{ {n-1 \atop k-1} \right\} + k \left\{ {n-1 \atop k} \right\}, \left\{ {n \atop 0} \right\} = [n=0]$.

# 第二类斯特林数

$$\left\{ {n \atop k} \right\} = \left\{ {n-1 \atop k-1} \right\} + k \left\{ {n-1 \atop k} \right\}, \left\{ {n \atop 0} \right\} = [n = 0]$$

| $n\backslash k$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 1 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 1 | 7 | 6 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 1 | 15 | 25 | 10 | 1 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 1 | 31 | 90 | 65 | 15 | 1 | 0 | 0 | 0 | 0 |
| 7 | 0 | 1 | 63 | 301 | 350 | 140 | 21 | 1 | 0 | 0 | 0 |
| 8 | 0 | 1 | 127 | 966 | 1701 | 1050 | 266 | 28 | 1 | 0 | 0 |
| 9 | 0 | 1 | 255 | 3025 | 7770 | 6951 | 2646 | 462 | 36 | 1 | 0 |
| 10 | 0 | 1 | 511 | 9330 | 34105 | 42525 | 22827 | 5880 | 750 | 45 | 1 |

# $k$ 部分拆数

**定义**

$k$ **部分拆数** 表示将 $n$ 个相同的小球, 放入 $k$ 个相同的盒子中, 每个盒子至少放 $1$ 个小球的不同的方案数, 记作 $p(n,k)$.

# $k$ 部分拆数

**定义**

$k$ **部分拆数** 表示将 $n$ 个相同的小球, 放入 $k$ 个相同的盒子中, 每个盒子至少放 $1$ 个小球的不同的方案数, 记作 $p(n,k)$.

- $k$ 部分拆数是下面方程的解的个数.

$$n-k=x_1+x_2+\cdots+x_k, x_1\geq x_2\geq\cdots\geq x_k\geq 0$$

# $k$ 部分拆数

**定义**

$k$ **部分拆数** 表示将 $n$ 个相同的小球, 放入 $k$ 个相同的盒子中, 每个盒子至少放 $1$ 个小球的不同的方案数, 记作 $p(n,k)$.

- $k$ 部分拆数是下面方程的解的个数.

$$n-k=x_1+x_2+\cdots+x_k, x_1\geq x_2\geq\cdots\geq x_k\geq 0$$

- 若其中有 $i$ 个数非零, 恰好有 $p(n-k,i)$ 个解.

# $k$ 部分拆数

**定义**

$k$ **部分拆数** 表示将 $n$ 个相同的小球, 放入 $k$ 个相同的盒子中, 每个盒子至少放 $1$ 个小球的不同的方案数, 记作 $p(n,k)$.

- $k$ 部分拆数是下面方程的解的个数.

$$n-k=x_1+x_2+\cdots+x_k, x_1\geq x_2\geq\cdots\geq x_k\geq 0$$

- 若其中有 $i$ 个数非零, 恰好有 $p(n-k,i)$ 个解.
- 可以得到 $p(n,k)=\sum_{i=0}^{k}p(n-k,i)$.

# $k$ 部分拆数

**定义**

$k$ **部分拆数** 表示将 $n$ 个相同的小球, 放入 $k$ 个相同的盒子中, 每个盒子至少放 $1$ 个小球的不同的方案数, 记作 $p(n,k)$.

- $p(n,k) = \sum_{i=0}^{k} p(n-k,i)$.

# $k$ 部分拆数

**定义**

$k$ **部分拆数** 表示将 $n$ 个相同的小球, 放入 $k$ 个相同的盒子中, 每个盒子至少放 $1$ 个小球的不同的方案数, 记作 $p(n,k)$.

- $p(n,k)=\sum_{i=0}^{k} p(n-k,i)$.
- $p(n-1,k-1)=\sum_{i=0}^{k-1} p(n-k,i)$.

# $k$ 部分拆数

**定义**

$k$ **部分拆数** 表示将 $n$ 个相同的小球, 放入 $k$ 个相同的盒子中, 每个盒子至少放 $1$ 个小球的不同的方案数, 记作 $p(n,k)$.

- $p(n,k)=\sum_{i=0}^{k} p(n-k,i)$.
- $p(n-1,k-1)=\sum_{i=0}^{k-1} p(n-k,i)$.
- 两式相减得递推式: $p(n,k)=p(n-1,k-1)+p(n-k,k), p(n,0)=[n=0]$.

# $k$ 部分拆数

$$p(n,k)=p(n-1,k-1)+p(n-k,k),p(n,0)=[n=0]$$

| $n\backslash k$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 1 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 1 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 1 | 3 | 3 | 2 | 1 | 1 | 0 | 0 | 0 | 0 |
| 7 | 0 | 1 | 3 | 4 | 3 | 2 | 1 | 1 | 0 | 0 | 0 |
| 8 | 0 | 1 | 4 | 5 | 5 | 3 | 2 | 1 | 1 | 0 | 0 |
| 9 | 0 | 1 | 4 | 7 | 6 | 5 | 3 | 2 | 1 | 1 | 0 |
| 10 | 0 | 1 | 5 | 8 | 9 | 7 | 5 | 3 | 2 | 1 | 1 |

# 球盒问题

**例**

$n$ 个相同/不同的小球, 放入 $k$ 个相同/不同的盒子, 每个盒子可以/不可以为空, 求方案数.

# 球盒问题

**例**

$n$ 个相同/不同的小球, 放入 $k$ 个相同/不同的盒子, 每个盒子可以/不可以为空, 求方案数.

| $n$ 个球 | $k$ 个盒子 | 盒子可以为空 | 盒子不可以为空 |
|---|---|---|---|
| 有标号 | 有标号 | $k^n$ | $k!S_2(n,k)$ |
| 有标号 | 无标号 | $\sum_{i=1}^{k} S_2(n,i)$ | $S_2(n,k)$ |
| 无标号 | 有标号 | $\binom{n+k-1}{k-1}$ | $\binom{n-1}{k-1}$ |
| 无标号 | 无标号 | $p(n+k,k)$ | $p(n,k)$ |