ACM算法与程序设计
（十一）
组合数学

杜育根

Ygdu@sei.ecnu.edu.cn

11. 组合数学

- 本课介绍了组合计数的一些基础原理，然后讲解了容斥原理，莫比乌斯反演，置换群 和 Polya 计数等高级计数方法。

11.1. 组合计数

- 排列数
- 从 n 个不同的元素中取 r（r≤n）个元素，按照顺序排成一列，可以有 A_n^r 种不同的排列，A_n^r 也称为从 n 个不同的元素取 r 个元素的排列数。
- 从 n 个不同的元素取 r个元素按顺便排成一列，选择第 1 个元素可以有 n 种方法，选择第 2 个元素可以有 n−1 种方法，以此类推。所以 A_n^r的计算方法如下：

$$A_n^r = n(n-1)(n-2)\cdots(n-r+1).$$

- 如果 n 和 r 都是整数，且 0≤r≤n，则有：

$$A_n^r = \frac{n!}{(n-r)!}$$

组合数

- 从 n个不同的元素中取 r（r≤n）个元素，不考虑顺序性，可以有 C_n^r 种不同的取法，C_n^r也称为从 n 个不同的元素取 r 个元素的组合数。
- 具有 n 个不同元素的集合的 r 组合数记为 C_n^r ，也记作 $\binom{n}{r}$，称为二项式系数。
- 如果 n 和 r 都是整数，且 0≤r≤n ，则有：

$$C_n^r = \frac{n!}{r!\,(n-r)!}$$

二项式

- 两个整数 x,y 和的 2次幂和 3 次幂的展开式如下：
- $(x+y)^2 = x^2 + 2xy + y^2$
- $(x+y)^3 = x^3 + 3x^2y + 3xy^2 + y^3$
- 推广到任意整数幂次，可以得到 二项式定理。
- 二项式定理
- 二项式定理，又称牛顿二项式定理，是指两个整数的整数幂次的展开式。

$$(x+y)^n = \sum_{i=0}^{n} C_n^i x^{n-i} y^i = \binom{n}{0} x^n + \binom{n}{1} x^{n-1} y \quad + \cdots + \binom{n}{n} y^n$$

- 二项式定理中的二项式系数，即是组合数。

一些有用的恒等式

- 在组合数学中，有一些常用的恒等式，如下所示：

$$\sum_{i=0}^{n} \binom{n}{i} = 2^n$$

$$\sum_{i=0}^{n} (-1)^i \binom{n}{i} = 0$$

$$\sum_{i=0}^{n} 2^i \binom{n}{r} = 3^n$$

- 最后一个式子可以这样证明：

$$3^n = (1+2)^n = \sum_{i=0}^{n} \binom{n}{r} 2^i$$

帕斯卡恒等式

- 帕斯卡恒等式 是组合数中一个常见的等式：

$$C_{n+1}^{k} = C_{n}^{k-1} + C_{n}^{k}$$

- 也可以表示成：

$$\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k}$$

习题：小明的多项式

- 小明遇到一个难解的多项式 $(px+qy)^k$，现在小明知道了参数 p，q，k，他想知道把多项式展开后，其中 $x^a y^b$ 这一项的系数 s，小明告诉你 a 和 b，想请聪明的你帮他计算一下 s%10007 的结果。
- 输入格式
- 输入一行，输入五个整数，每两个整数之间用一个空格隔开，分别为p，q，k，a，b（0≤k≤1000,0≤a,b≤k），保证 a+b=k。
- 输出格式
- 输出一行，输出一个整数，输出 s%10007的结果。
- 样例输入
- 1 2 5 3 2
- 样例输出
- 40

习题：吃辣椒

- 小明喜欢吃辣椒。他一共有 N 种辣椒，每种辣椒有一个辣度值 p_i。当小明吃辣椒的时候，他恰好一口吃 K 种辣椒。这辣椒有一点奇怪，当同时吃下 K 种辣椒时，只能感受到辣度值最大的那种辣椒的辣度值。小明会试下每一个 K 种辣椒的组合，他想要知道他能感受到的辣度值之和是多少。
- 输入格式
- 第一行输入两个正整数 N和 K（N≤100,000,K≤50），表示有 N 种辣椒和一口会吃 K 种辣椒。
- 第二行输入 N 个整数 p_i（$0 \le p_i \le 10^9$），表示每种辣椒的辣度值。
- 输出格式
- 输出一行，输出一个整数，表示总辣度值之和，结果对 1,000,000,007 取余。
- 样例输入1
- 5 2
  1 2 3 4 5
- 样例输出1
- 40
- 样例输入2
- 5 3
  1 2 3 4 5
- 样例输出2
- 45

习题：小明走迷宫

- 小明从一个 n 行 m 列的迷宫的左上角走到右下角，小明每次只能向下或者向右走一步，小明想知道他有多少种走法。
- 输入格式
- 输入两个个整数 $n(2 \le n \le 10^5), m(2 \le m \le 10^5)$。
- 输出格式
- 由于方案数太多，输出最后结果对 1000000007 取模的结果。
- 样例输入1
- 2 3
- 样例输出1
- 3
- 样例输入2
- 5 4
- 样例输出2
- 35

- 提示: 用 $inv[i] = (p - p/i)inv[p\%i]\%p$ 这个公式来线性预处理逆元。

11.2. 鸽巢原理

- 这一节我们介绍一个组合数学中一个重要而又初等的原理。它能够用来解决很多有趣的问题，常常得出一些令人惊讶的结论。
- 鸽巢原理
- 鸽巢原理 又称为抽屉原理。其最简单的形式如下。
- 如果 n+1 个物体被放进 n个盒子，那么至少有一个盒子包含两个或者两个以上的物体。

- 鸽巢原理最简单的应用比如在 13 个人中存在两个人，他们生日在同一个月份里。

应用

- 例题：给定一个含有 n 个正整数的数列 A，找到一对 i,j(1≤i≤j≤n) 使得 $\sum_{k=i}^{j} A_k$是 n 的整数倍数。
- 我们可以预处理出所有的前缀的区间和对于 n 取模的结果。如果这 n 个模数中有一个结果为 0，那么这个前缀和就是 n的整数倍。否者，剩下的取模结果只有 n−1 种，但是一共有 n 个结果，那么这 n个结果中必然后两个模是一样的。那么对于这两个取模结果相同的区间相减，就得到了一个 n 的整数倍的区间。这是鸽巢原理在竞赛中一个最经典的应用。

习题：购买礼品

- 计算之道比赛马上开始了，小明需要购买一些小礼品用来抽奖。
- 小明一共带了 n 位同事前往，第 i位同事身上携带了 pi 元钱。真巧，礼品店的礼物的单价也正好都是 n。但是礼品店的老板需要告诉小明，他这里没有零钱，所以小明需要凑到正好买整数个礼物的钱。现在小明发动同事开始凑钱，每位同事要么不参与凑钱，要么就把所有的钱交给小明。
- 输入格式: 输入第一行一个整数 n(1≤n≤100000)，表示一同前往的同事数量。
- 接下来一行输入每个人携带的钱的数量 pi(1≤pi≤100000)
- 输出格式: 如果小明不能凑出合适的钱，输出一行"Sadly"。
- 否者第一行输出参与凑钱的人数 k，接下来一行以任意顺序输出 k个人的编号（编号从 1 开始）用空格隔开。如果有多个方案，输出任意一个即可。
- 本题答案不唯一，符合要求的答案均正确
- 样例输入
- 5
  1 9 4 2 6
- 样例输出
- 3
  2 3 4

习题：凑数

- 给出 n 种数，第 i 个数为 a_i，每种数能选任意多个，问你最少需要选多少个数使得选出来的数的和 正好 是 m。
- 输入格式：输入第一行两个整数 n(1≤n≤100),m(1≤m≤10^9)。
- 接下里一行输入 n个数 a_i (1≤a_i≤100)
- 输出格式：输出最少需要的数的个数（一种数选多少个就算多少个）。如果不能正好组成 m，输入"-1"。
- 样例输入1
- 3 12
  5 1 4
- 样例输出1
- 3
- 样例输入2
- 1 3
  2
- 样例输出2
- -1

- 提示: 把 a 从小打大排序以后，比 a_{n-1} 小的数最多用 a_{n-1} 个，因为根据鸽巢原理，如果用超过 a_n 个，必然会出现一个区间和是 a_n的整数倍，就可以用 a_n 直接替代。这样可以把背包容量减少，然后进行 dp。

11.3. 康拓展开和逆展开

- 康拓展开
- 已知集合A有n个不同的元素，$S = (k_n, k_{n-1}, \cdots, k_2, k_1)$ 是集合A的一个排列，假设S是所有按字典序从小到大排序的排列中的第x个（从0开始）。则整数x可以展开成如下形式：

$$x = a_n (n-1)! + a_{n-1}(n-2)! + \cdots + a_2 1! + a_1 0!$$

- 其中 a_i 为所有当前未出现的元素中比元素 k_i 小的元素个数。
- 比如对于 1,2,3,4的全排列中 3214是第 $2 \times 3! + 1 \times 2! + 0 \times 1! + 0 \times 0! = 14$ 个排列。
- 康拓展开主要用于对排列做哈希，最经典的应用就八数码问题，对于一个状态可以用康拓展开来标记，这样比直接用字符串记录更节省内存。

康拓逆展开

- **康拓逆展开是康拓展开的逆操作。已知道n个元素的某个排列是第x个，求这个排列具体是什么。康拓逆展开还是基于$x = \boldsymbol{a_n}\,(n-1)! + \boldsymbol{a_{n-1}}(n-2)! + \cdots + \boldsymbol{a_2}1! + \boldsymbol{a_1}0!$这个公式。**
- **首先我们证明一个不等式**

$$a! > (a-1)(a-1)! + (a-2)(a-2)! + \cdots 1 \times 1! + 0 \times 0!$$

- **可以用数学归纳法证明：**
- 初始条件 $a = 1$ 的时候 $1! > 0 \times 0!$ 成立。
- 假设 $a = x$ 的时候成立，即

$$x! > (x-1)(x-1)! + (x-2)(x-2)! + \cdots 1 \times 1! + 0 \times 0!$$

- 那么有

$$(x+1)! = x! + x \times x! > x \times x! + (x-1)(x-1)! + (x-2)(x-2)! + \cdots + 1 \times 1! + 0 \times 0!$$

- 于是 $a = x + 1$ 的时候也成立。根据数学归纳法可以得到该结论正确。
- **证毕。**

- 所以可以得到 $a_n = \frac{x}{(n-1)!}$，$a_{n-1} = \frac{x\%(n-1)!}{(n-2)!}$，$\cdots$，$a_1 = 0$。现在要根据 a数组来复原排列。我们可以从 a_n 开始求 p_n，然后标记 p_n 已经使用，然后在剩余的数中查找第$a_{n-1}+1$ 大的元素就是 p_2。依次下去直到剩下最后一个数。
- 对于 n比较小的情况可以直接用暴力来查找。但是 n比较大的时候，需要用到一些数据结构，比如线段树，树状数组或者平衡树来维护，这样提高在剩下元素中查找第 k大元素的效率。

习题：矩阵变换

- 已知矩阵 $A=\begin{bmatrix}1 & 2 & 3 & 4\\ 8 & 7 & 6 & 5\end{bmatrix}$
- 矩阵的状态可以用序列 12345678来表示。即从左上角开始，按顺时针方向依次写下每个数字。
- 对于这个矩阵，我们可以有 3 种不同的操作，具体如下：
- A：互相交换上下两行，如上例可变换为状态 87654321；
- B：两行分别循环右移一位，如上例可变换为状态 41236785；
- C：中间 4 个元素同时顺时针旋转一位，如上例可变换为状态 17245368。
- 现在已知矩阵的初始状态与目标状态，现在请问聪明的你，最小需要多少步变换，如果有多种方案则输出字典序最小的方案。
- 输入格式：输入有两行。第一行输入一个状态，表示初始状态，第二行输入一个状态，表示目标状态。两个状态长度均为 8，均由 1 到 8这 8 个数字组成，且没有重复数字。
- 输出格式: 输出一行，输出变换的顺序。每一步变换，输出对应的编号。
- 样例输入
- 12345678
  87654321
- 样例输出
- A

习题：*排列加法（选做）

- p, q 是集合 $\{0,1,2,\cdots n-1\}$ 上的两个全排列。
- 定义 Ord_p 为排列 p 在所有排列中的按照字典序排序的排名（从 0 开始）。比如 $Ord_{\{0,1,\cdots n-1\}} = 0$，$Ord_{\{n-1,n-2,\cdots 0\}} = n! - 1$。
- $Perm(x)$ 为排名第 x 的排列。比如 $Perm(0) = \{0,1,\cdots n-1\}$，$Perm(n!-1) = \{n-2, n-1, \cdots 0\}$。
- 定义排列 p和 q的加法如下：

$$p + q = Perm((Ord_p + Ord_q)\%n!)$$

- 给定两个排列，计算着两个排列相加的结果。
- 输入格式
- 输入第一行一个整数 n(1≤n≤50000)。
- 接下来一行输入第一个排列 p，用空格隔开。
- 接下来一行输入第二个排列 q，用空格隔开。
- 输出格式
- 输出 p+q 的结果。

样例输入1	样例输入2	样例输入3
2 0 1 0 1	3 1 2 0 2 1 0	5 2 4 0 1 3 1 2 4 0 3
样例输出1	样例输出2	样例输出3
0 1	1 0 2	4 0 3 1 2

11.4. 容斥原理

- 概述
- 容斥原理是组合数学中一种常用的计数方法。把符合每个条件的数目相加，然后再把计数时重复计数的排斥掉，使得计数即没有重复也没有遗漏。
- 容斥原理的计数，一般情况下，首先加上所有满足 1个条件的数目，去掉所有满足 2 个条件的数目，加上所有满足 3个条件的数目，以此类推。

公式

- 求集合的并集，可以用类似容斥原理的方法来求。
- 已知两个集合A和B，则两个的并集可以这么求解：

$$|A \cup B| = |A| + |B| - |A \cap B|$$

- 同样三个集合的并集可以这么求解：

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |B \cap C| - |A \cap C| + |A \cap B \cap C|$$

- 推广到 n 个集合，可以这么求解集合的并集：

$$|A1 \cup A2 \cup \cdots \cup An| = \sum |Ai| - \sum |Ai \cap Aj| + \sum |Ai \cap Aj \cap Ak| - \cdots + (-1)^{n-1} |A1 \cap \cdots \cap An|$$

严格证明(利用数学归纳法证明容斥原理)

- 证明： 当 n=2 时，等式成立。
- 假设 $n = s(s \geq 2, s \in N^+)$ 时结论成立，则当 n=s+1 时，

$$
\begin{aligned}
\left|\bigcup_{i=1}^{m} A_i\right| &= \left|\bigcup_{i=1}^{s+1} A_i\right| = \left|\left(\bigcup_{i=1}^{s} A_i\right) \cup A_{s+1}\right| \\
&= \left|\bigcup_{i=1}^{s} A_i\right| + |A_{s+1}| - \left|\left(\bigcup_{i=1}^{s} A_i\right) \cap A_{s+1}\right| \\
&= \left|\bigcup_{i=1}^{s} A_i\right| + |A_{s+1}| - \left|\bigcup_{i=1}^{s} (A_i \cap A_{s+1})\right| \\
&= \sum_{1 \leq i_1 \leq s+1} |A_i| + \sum_{k=2}^{s} (-1)^{k-1} \sum_{1 \leq i_1 < i_2 < \cdots i_k \leq s} |A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_k}| \\
&+ \sum_{k=1}^{s-1} (-1)^k \sum_{1 \leq i_1 < i_2 < \cdots < i_k \leq s} |A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_k} \cap A_{s+1}| + (-1)^s |A_1 \cap A_2 \cap \cdots \cap A_s \cap A_{s+1}| \\
&= \sum_{1 \leq i_1 \leq s+1} |A_i| + [\sum_{k=2}^{s} (-1)^{k-1} \sum_{1 \leq i_1 < i_2 < \cdots < i_k \leq s} |A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_k}| \\
&+ \sum_{k=2}^{s} (-1)^{k-1} \sum_{1 \leq i_1 < i_2 < \cdots < i_{k-1} \leq s} |A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{k-1} \cap A_{s+1}|] + (-1)^s |A_1 \cap A_2 \cap \cdots \cap A_{s+1}| \\
&= \sum_{1 \leq i_1 \leq s+1} |A_{i_1}| + \sum_{k=2}^{s} (-1)^{k-1} \sum_{1 \leq i_1 < i_2 < \cdots < i_k \leq s+1} |A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_k}| + (-1)^s |A_1 \cap A_2 \cap \cdots \cap A_{s+1}| \\
&= \sum_{k=1}^{s+1} (-1)^{k-1} \sum_{1 \leq i_1 < i_2 < \cdots < i_k \leq s+1} |A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_k}| \\
&= \sum_{k=1}^{s+1} (-1)^{k-1} \sum_{1 \leq i_1 < i_2 < \cdots < i_k \leq n} |A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_k}|
\end{aligned}
$$

- 所以当 n=s+1时，结论仍是成立的。因此对于任意 n$\in N^+$,n≥2，等式均成立。

小学数学题：

- 在期末考试上，每位学生参加了语文数学两门课的考试。现在已知有 x个人语文成绩及格，y个人数学成绩及格，z个人两门课的成绩都及格。那么请问，一共有多少位学生至少一门课的成绩是及格的。
- 我们可以集合求并的方法来求解，假设语文成绩及格的同学集合为 A，数学成绩及格的同学集合为 B，那么至少一门课的成绩是及格的集合就是集合 A 和 B 的并集 $A \cup B$。通过前面的讲解，我们知道，可以用容斥原理的方法来计数：
- $|A \cup B| = |A| + |B| - |A \cap B| = x+y-z$

例题1

- 求区间 [a,b] 上和 x互质的数的个数。
- 解法：首选把x分解质因子，$x = p_1^{a1} p_2^{a2} \cdots p_m^{am}$。其中$p_1, p_2, \cdots p_m$ 都是质数，我们称为x的因子。这是我们来容斥。初始的时候有 b−a+1 个数，减去 [a,b] 上和 $p_1, p_2, \cdots p_m$不互质的数的个数，然后加上 [a,b] 上同时和 $p_1p_2, p_1p_3, \cdots p_1p_m$，$p_2p_3, \cdots p_{m-1}p_{m-2}$不互质的数个数 ……。

代码

```
#include <iostream>
#include <vector>
using namespace std;
vector<int> breakdown(int x) {
    vector<int> l;
    for (int i = 2; i * i <= x; ++i) {
        int cnt = 0;
        while (x % i == 0) {
            ++cnt;
            x /= i;
        }
        if (cnt) {
            l.push_back(i);
        }
    }
    if (x > 1) {
        l.push_back(x);
    }
    return l;
}
int main() {
    int a = 10, b = 20, x = 15;
    vector<int> list = breakdown(x);
    int ans = 0, len = list.size();
    for (int i = 0; i < (1 << len); ++i) {
        int cnt = 0, pp = 1;
        for (int j = 0; j < len; ++j) {
            if (i & (1 << j)) {
                ++cnt;
                pp *= list[j];
            }
        }
        ans += (b / pp - (a - 1) / pp) * ((cnt &
1) ? -1 : 1);
    }
    cout << ans << endl;
    return 0;
}
```

例题2

- 小明和小花在玩一个游戏。小明先手，他选择一个非空集合，集合中的元素都是整数对（a,b），满足条件：$1 \le a < b \le N$，且 $\gcd(a,b) = 1$。比如 N=5，可以选择集合{(1,2),(3,4),(3,5}。
- 小花后手，他需要找到一个整数 x∈{2,3,…,N}，使得对于任意整数对 (a,b)，满足以下两个条件的任意一个：a,b<x；a,b≥x。例如集合 {(1,2),(3,4)}，x可以等于 3。如果小花找不到这样的 x，则小明获胜。现在小明想知道，有多少种不同的方案能使他获胜。（N≤20）。
- 解析：
- 首先预处理出来能选择的整数对为 m 对。那么总的集合数为 2m。我们以 x为基础容斥。首先枚举一个 x，减去满足 a,b<x 或者 a,b≥x 的集合个数。然后减去同时满足两个 x 的方案数，加上同时满足三个 x 的方案数 ……。
- 关于如何求同时满足多个x的方案数。假设需要同时满足的数从小到大排序为 $x1, x2, \cdots xm$。那么选择的整数对 a,b 除了需要满足 $\gcd(a,b) = 1$ 之外，还要同时满足
- $(a,b) < x1$ || $(a,b) \ge x1, (a,b) < x2$ || $(a,b) \ge x2, \cdots (a,b) < xm$ || $(a,b) \ge xm$
- 实际上，a,b 必须属于区间 $[1,x1), [x1,x2), [x2,x3) \ldots [xm,N]$ 中的同一个。

习题：互质数

- 如果两个数 a 和 b 满足条件 gcd(a,b)=1，则数 a 和 b 互质。现在要求出第 k 小的且与 n 互质的数。
- 输入格式
- 输入一行，输入两个整数 n 和 k $(1 \le n \le 10^6, 1 \le k \le 10^8)$。
- 输出格式
- 输出一行，输出第 k 小的且与 n 互质的数。
- 样例输入
- 2321 4
- 样例输出
- 4
- 提示：二分一下。

习题：小游戏

- 小明和小花在玩一个游戏。小明先手，他选择一个非空集合，集合中的元素都是整数对 (a,b)(a,b)(a,b)(a,b)，满足条件：$1 \le a < b \le N$，且 $\gcd(a,b)=1$。比如 $N=5$，可以选择集合 {(1,2),(3,4),(3,5)}。
- 小花后手，他需要找到一个整数 $x \in \{2,3,\cdots,N\}$，使得对于任意整数对 (a,b)，满足以下两个条件的任意一个：$a,b<x$；$a,b \ge x$。例如集合 {(1,2),(3,4)}，x可以等于 3。如果小花找不到这样的 x，则小明获胜。现在小明想知道，有多少种不同的方案能使他获胜。
- 输入格式
- 输入一行，输入一个整数 N（$2 \le N \le 20$）。
- 输出格式
- 输出一行，输出一个整数，表示小明可以选择的集合个数，结果可能会很大，输出对 10^9 取余的结果即可。
- 样例输入
- 3
- 样例输出
- 5

习题：*蛤（选做）

- 有 m 个石头排列成一个环（编号重 0 到 m−1），有 n 只青蛙从第 0 块石头开始跳跃，第 i 只青蛙每次跳跃 a_i的距离。就是说如果第 i只青蛙在第 j 块石头，那么跳跃之后它将到达第 $j + a_i \% m$ 块石头。
- 所有的青蛙都一直跳跃，请你计算所有青蛙能到达的石头的编号的和（一块石头如果多个青蛙能达到，也只计算一次）。
- 输入格式: 输入第一行两个整数 $n(1 \le n \le 10^4), m(1 \le m \le 10^9)$。
- 第二行输入每个青蛙能跳的距离$(1 \le a_i \le 10^9)$。
- 输出格式: 输出答案。
- 样例输入1
- 2 12
  9 10
- 样例输出1
- 42
- 样例输入2
- 9 96
  81 40 48 32 64 16 96 42 72
- 样例输出2
- 1872

提示: 首先应该得出结论，每个青蛙能到达的是 $\gcd(a_i, m)$ 的倍数的位置。那么我们可以按照 m 的因子进行容斥。预处理出来所有的 m的因子。然后根据$\gcd(a_i, m)$ 判断每个因子的所有倍数是否应该被加上。然后容斥，有的因子可能被多加或者多减了，这里可以用 dp来实现。

11.5. 莫比乌斯反演

- 莫比乌斯函数
- 首先先介绍一个特殊的函数，莫比乌斯函数，其定义如下：

$$\mu(n)=\begin{cases}1 & n=1\\ (-1)^k & n=p_1p_2\cdots p_k\\ 0 & other\end{cases}$$

- 对于第二种情况，$p_1,p_2,\cdots p_k$ 是互不相同质数。
- 而莫比乌斯函数有一个重要的性质就是，对n所有因子d的和，有

$$\sum_{d|n}\boldsymbol{\mu(d)}=\begin{cases}1,n=\mathbf{1}\\ 0,n>1\end{cases}$$

- 在后面我们需要用到这个重要的性质。这个性质这里不做证明。

莫比乌斯反演

- 定理：设 $g(n)$和$f(n)$是定义在非负整数集合上的两个函数，并且满足 $g(n)=\sum_{d|n} f(d)$，那么可以得到结论

$$f(n)=\sum_{d|n}\mu(d)g\left(\frac{n}{d}\right)$$

- 这就是莫比乌斯反演。
- 证明：
-

$$\begin{aligned}
&\sum_{d|n}\mu(d)g\left(\tfrac{n}{d}\right)\\
=&\sum_{d|n}\mu(d)\sum_{k|\frac{n}{d}}f(k)\\
=&\sum_{d|n}\sum_{k|\frac{n}{d}}f(k)\mu(d)\\
=&\sum_{k|n}\sum_{d|\frac{n}{k}}f(k)\mu(d)\\
=&\sum_{k|n}f(k)\sum_{d|\frac{n}{k}}\mu(d)\\
=&f(n)
\end{aligned}$$

- 证毕 。实际上莫比乌斯反演是容斥的一种高级形式的表征。

几种常见的反演

1.

$$f(n) = \mu(n)$$

$$g(n) = \begin{cases} 1, n = 1 \\ 0, n > 1 \end{cases}$$

$$f(n) = \sum_{d|n} \mu(d) g\left(\frac{n}{d}\right) = \mu(n)$$

所以莫比乌斯函数本身也是可以反演的。

2.

$$g(n) = n$$

$$f(n) = \phi(n)$$

- 其中 $\phi(n)$ 表示欧拉函数。欧拉函数有一个性质就是 $n = \sum_{d|n} \phi(d)$，所以可以满足反演的性质。

线性预处理

- **莫比乌斯函数可以在线性时间内预处理出来，给出预处理的代码。**

```
int mu[MAXN], primes[MAXN];
bool vis[MAXN];
void init() {    //此处为线性求mu 函数
    memset(vis, 0, sizeof(vis));
    mu[1] = 1;
    int cnt = 0;
    for( int i = 2; i < MAXN; ++i) {
        if (!vis[i]) {
            prime[cnt++] = i;
            mu[i] = -1;
        }
        for (int j = 0; j < cnt && i * prime[j] < MAXN; ++j) {
            vis[i * prime[j]] = 1;
            if (i % prime[j]) {
                mu[i * prime[j]] = -mu[i];
            } else {
                mu[i * prime[j]] = 0;
                break;
            }
        }
    }
}
```

应用

- 例题：求解区间 $[1,n]$ 和 区间 $[1,m]$ 上互质的数的个数。

$$
\begin{aligned}
&\sum_{i=1}^{n}\sum_{j=1}^{m}[gcd(i,j)==1]\\
&=\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{d|gcd(i,j)}\mu(d)\\
&=\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{d|i\ and\ d|j}\mu(d)\\
&=\sum_{d=1}^{min(n,m)}\mu(d)\sum_{i=1}^{\lfloor\frac{n}{d}\rfloor}\sum_{j=1}^{\lfloor\frac{m}{d}\rfloor}1\\
&=\sum_{d=1}^{min(n,m)}\mu(d)\lfloor\frac{n}{d}\rfloor\lfloor\frac{m}{d}\rfloor
\end{aligned}
$$

- 经过上面的化简以后，我们能够在 $O(\min(n,m))$ 的时间内完成单次查询。但是还不够优化，可以继续优化。我们会发现 $\lfloor\frac{n}{d}\rfloor$的值最多有$2\sqrt{n}$种， $\lfloor\frac{m}{d}\rfloor$最多有 $2\sqrt{m}$ 种，并且相同取值对应的 d 是一段连续的区间，所以我们可以把区间最多分成 $2(\sqrt{n}+\sqrt{m})$ 个，在预处理出 μ(i) 的前缀和之后，可以在 $O(2(\sqrt{n}+\sqrt{m}))$ 的时间复杂度完成单次查询。查询部分的写法可以参考如下。

```
int t = min(n, m), ans = 0;
for (int i = 1, j = 1; i <= tmp; i = j + 1) {
    j = min(n / ( n / i), m / (m / i));
    ans += (summu[j] - summu[i - 1]) * (n / i) * (m / j);
}
```

- 思考：如果是求 $\sum_{i=1}^{n}\sum_{j=1}^{m}gcd(i,j)$应该怎么求?

习题：GCD对数

- 求出 gcd(a,b)=k(1≤a≤n,1≤b≤m) 的对数。
- 输入格式
- 输入第一行一个整数 T(1≤T≤10000) 表示询问的次数。
- 接下来 T 行数，每行输入三个整数 n,m,k(1≤n,m≤100000，1≤k≤min(n,m)。
- 输出格式
- 对于每次询问一行表示答案。
- 样例输入
- 3
  6 9 2
  7 7 1
  2 8 1
- 样例输出
- 9
  35
  12
- 提示:转换成 gcd(a,b)=1。

习题：青云的机房组网方案

- 联想公司推出一款新型显示屏。该显示屏由 n 行 m 列的点阵组成。任选点阵中两个点可以构造出一条“基准线”，质检员通过校正每条基准线上的像素信息来确保显示屏没有一丝一毫的缺陷。现在，质检员想知道对于一个由 n 行 m 列的点阵组成的显示屏而言，有多少条不同的基准线。两条基准线不同当且仅当它们不重合，即两条基准线的交点数量是有穷的。
- 由于显示屏的基准线的数量会很多，为了简化我们只需要计算数量对 2^{30}取模后的结果。
- 输入格式
- 第一行有一个正整数 T 为数据组数，接下来有 T 行，每行两个正整数 n,m。
- T≤10,000, 2≤n,m≤400,000。
- 输出格式
- 输出一共有 T 行 ， 每行一个整数，为该组数据的基准线数量。
- 样例输入
- 4
  2 2
  7 10
  23 34
  100 100
- 样例输出
- 6
  1111
  139395
  22791174

习题：*阿里云秘钥池（选做）

- 对于每个正整数 n，我们定义它的 p进制表示 由 m个非负整数 $a1, a2, \cdots, am$ 组成，并且这些数字满足$n = \sum_{i=1}^{m} a_i p^{i-1}$，以及 $0 \le a_i < p\ (i = 1,2, \cdots, m-1)$ 和 $1 \le a_m < p$。
- 在阿里云台上，用户登录的秘钥都是由一个正整数组成。一个可被用作秘钥的正整数 n，它的 p 进制表示需要满足 $1 \le ai < p\ (i = 1,2, \cdots, m)$ 且 $\gcd(a_i, a_{i+1}) = 1\ (i = 1,2, \cdots, m-1)$。比如当 p=10 时，61 是一个合法秘钥，3216 也是；但是当 p=100 时，61 依旧是一个合法秘钥，3216 却不是。
- 给出正整数 L,R和 p，请你统计满足 L≤n≤R 并且 n 是一个合法秘钥的正整数 n 的数量。
- 输入格式: 第一行包含一个正整数 $T(1 \le T \le 10^3)$，表示有 T组测试数据。
- 接下来依次给出每组测试数据，每组测试数据仅一行，包含三个正整数 $L, R(1 \le L \le R \le 10^{18})$ 和 $p(2 \le p \le 10^5)$，含义见题目描述。
- 保证不超过 50 组数据满足 $p > 10^3$。
- 输出格式: 对于每组数据输出一行，包含一个整数，表示满足条件的数字数量。
- 样例输入
- 4
  5 7 9
  1 100 2
  1 100 5
  2017 2121 10
- 样例输出
- 3
  6
  41
  10

提示:这题需要数位 dp 的知识，如果不会数位 dp，可以先跳过这题。

11.6. 置换群和 Polya 计数

- 置换群
- 设 $a1, a2, a3 \cdots an$ 是 1 到 n上的一个排列。定义集合 $X = \{1,2,3 \cdots n\}$ 上的函数 $f(x)$ 为
- $f(x) = ax, \ \ x \in X$
- 所以函数 $f(x)$ 是集合 X 到自身的一个一一映射。我们称这样的一个映射为一个置换。可以用如下符号表示置换 f。

$$f = \begin{pmatrix} \mathbf{1} & \mathbf{2} & \dots & \boldsymbol{n} \\ a1 & a2 & \dots & an \end{pmatrix}$$

- 一个置换作用到一个排列上生成了另外一个排列。
- 比如

$$f = \begin{pmatrix} \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} \\ \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{1} \end{pmatrix}$$

$$p = (2 \ 4 \ 1 \ 3)$$

- 那么 f 作用于 p 的结果是得到另外一个排列

$$p' = (3 \ 1 \ 2 \ 4)$$

置换之间也能定义乘法

- 置换 f 乘 g 定义如下:

$$f = \begin{pmatrix} 1 & 2 & \dots & n \\ a1 & a2 & \dots & an \end{pmatrix} \qquad g = \begin{pmatrix} 1 & 2 & \dots & n \\ b1 & b2 & \dots & bn \end{pmatrix}$$

- 那么

- $f \circ g = \begin{pmatrix} 1 & 2 & \dots & n \\ g(a1) & g(a2) & \dots & g(an) \end{pmatrix}$

- 举例

$$f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 5 & 6 & 2 & 4 & 3 \end{pmatrix} \qquad g = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 2 & 4 & 3 & 5 & 1 \end{pmatrix}$$

$$f \circ g = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 5 & 1 & 2 & 3 & 4 \end{pmatrix}$$

循环

- 对于一个置换

$$f = \begin{pmatrix} 1 & 2 & \dots & n \\ a1 & a2 & \dots & an \end{pmatrix}$$

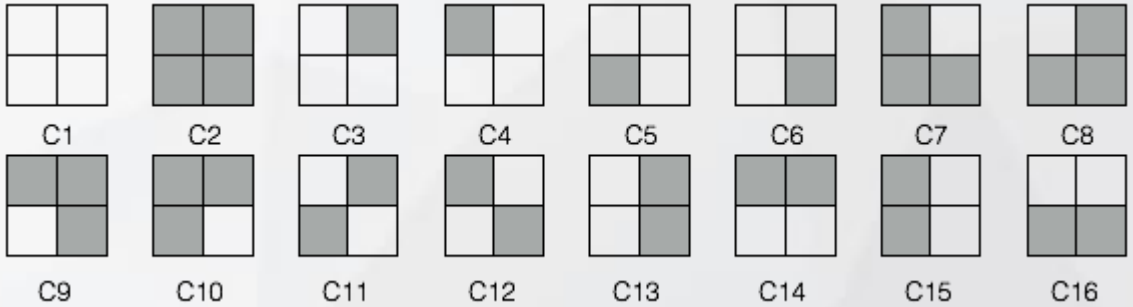
- 1到n上的每个元素都能通过置换f还原。也就是一定存在$f(f(f(f\cdots f(x)))) = x$。这样相当于形成了很多个环。我们把在同一个环中的数称为一个循环。这样一个置换可以分解成若干个循环的乘积。比如

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 1 & 4 & 2 \end{pmatrix} = (1\ \ 3)(2\ \ 5)(4)$$

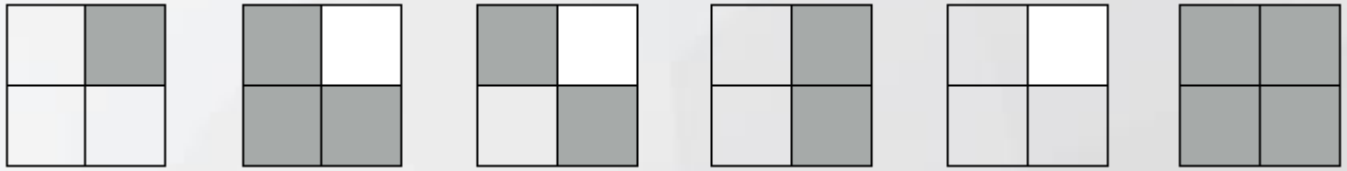
- 我们称一个置换分解出来的循环的个数为该置换的循环节。比如(1 3)(2 5)(4)的循环节为 3。

Polya 计数

- Polya 计数是用来做等价类计数的很方便快捷的方法。有一个经典的问题，给 2×2 的方格中涂黑白两色，一共有多少种方法？一共是 16 种。如下图。



- 但是如果定义了“旋转对称性”，规定逆时针旋转 90^o，180^o，270^o后方案算作一种，那么答案就变成了 6 种了，如下图。



- 为了统计等价类计数问题。首先需要用一个置换集合 G 来描述等价关系。如果一个置换把其中一个方案映射到另外一个方案，就说两者是等价的。比如“逆时针转 90^o”这个置换把 C3 映射到 C4。F 中任意两个置换的乘积也应当在 F 中，这样才能构成一个置换群。

Polya 计数方法如下：

- 如果用 k种颜色着色，对于 G 中的每一个置换 f，我们用 m(f) 表示这个置换的循环节个数。
- 那么不同的方案数为

$$\frac{\sum_{f \in G} k^{m(f)}}{|G|}$$

- 证明略。
- 比如对于上面的例子有 4个置换，分别为（标号如下图），

$$\begin{pmatrix}1&2&3&4\\1&2&3&4\end{pmatrix}\begin{pmatrix}1&2&3&4\\2&3&4&1\end{pmatrix}\begin{pmatrix}1&2&3&4\\3&4&1&2\end{pmatrix}\begin{pmatrix}1&2&3&4\\4&1&2&3\end{pmatrix}$$

1	2
4	3

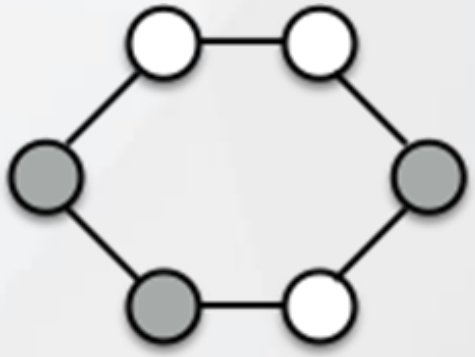
- 写成循环的乘积如下 (1)(2)(3)(4)，(1 2 3 4)，(1 3)(2 4)，(1 4 3 2)。总得方案数为 。

例题1

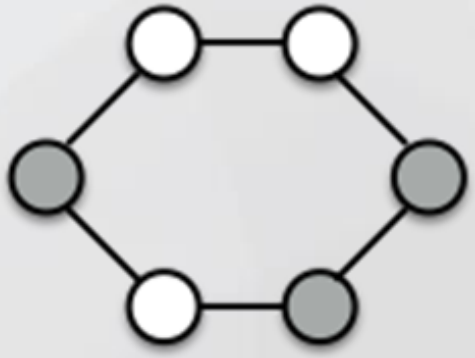
- 给一个置换 f，求最小的k>2使得 $f^k = f$。
- 解析：
- 实际上就是问 f 自己作用于自己多少次能变回到自己。这个问题很简单，把 f 写成循环的形式，实际上就是所有循环的长度的最小公倍数。

例题2

- 项链和手镯都是由 n 颗珠子穿成的环形首饰，每颗珠子可以用 t 种颜色着色。它们的区别在于手镯可以翻转，但是项链不可以翻转。换句话说，图(a) 和 图(b)，如果是手镯看做相同，如果是项链则看做不同。当然不管是项链还是手镯，旋转之后一样看做相同。求出各有多少个不同项链和手镯。n=5,t=1的时候，项链有 51 种，手镯有 39 种。



图(a)



图(b)

例2解析：

- 我们给珠子按照顺时针从 0 到 n−1 编号。
- 我们首先考虑旋转：如果逆时针旋转 i课珠子，那么珠子 0,i,2i,⋯,(k−1)i。如果当 n|ki 的时候，相当于0，所以最小的 k=n/gcd(i,n)。也就是说一个循环有 n/gcd(i,n) 个元素。根据对称性，每个循环都是这样长，所以有 gcd(i,n) 个循环。
- 翻转：分两种情况。n是奇数，有 n条对称轴，每条对称轴对应 (n−1)/2 个长度为 2的循环和一个长度为 1的循环。也就是 (n+1)/2 个循环。n为偶数的时候，有两种对横轴，一种是穿过珠子的对轴轴，有n/2 条，各形成 n/2−1 个长度为 2 的循环和 2个长度为 1的循环，另外一种不穿过珠子的对称轴有 n/2 条，各形成 n/2 个长度为 2 的循环。

参考代码

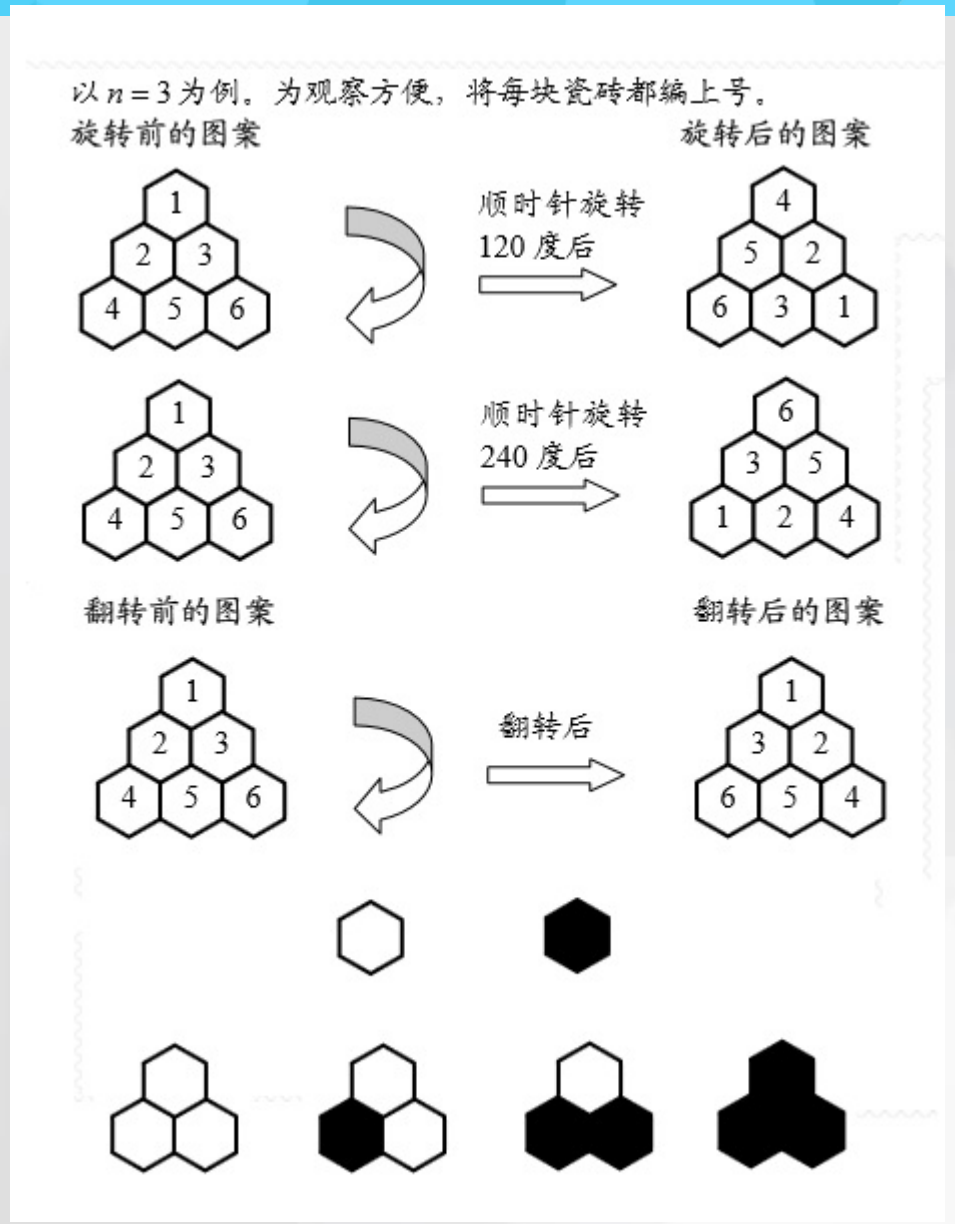
```
1.  void count(int n, int t) {
2.      LL pow[maxn];
3.      pow[0] = 1;
4.      for (int i = 1; i <= n; ++i) {
5.          pow[i] = pow[i - 1] * t;
6.      }
7.      long long a = 0;
8.      for (int i = 0; i < n; ++i) {
9.          a += pow[gcd(i, n)];
10.     }
11.     long long b = 0;
12.     if (n % 2 == 1) {
13.         b = n * pow[(n + 1) / 2];
14.     } else {
15.         b = n / 2 * (pow[n / 2 + 1] + pow[n / 2]);
16.     }
17.     printf("%lld %lld\n", a/ n, (a + b) / 2 / n);
18. }
```

习题：小明的暗号

- 小明和小花妹使用了一种全新的编码方式进行通信。密钥是一组整数序列：$a1, a2, \cdots an$。每个数都大于 0，小于等于 n。编码方式为：第 i个字符换至 ai位置处。然后得到的编码再编码一次，如此重复 k次。信息的长度小于等于 n。如果长度小于 n。则后面用空格补齐。
- 写一个程序读入编码规则和待编码字符串，输出编码结果。
- 输入格式
- 第一行输入两个整数 $n(1 \leq n \leq 200)$，$k(1 \leq k \leq 10^8)$。
- 第二行输入 n个整数，表示密钥，保证输入的每个数都不一样。
- 接下来一行输入一行字符串表示需要加密的串，由大写小写字母，数字和空格组成。
- 输出格式
- 输出加密之后的字符串。末尾有空格也要输出。
- 样例输入
- 10 1
  4 5 3 7 2 8 1 6 10 9
  Hello Bob
- 样例输出
- BolHeol b

习题：黑白瓷砖

- 小可可在课余的时候受美术老师的委派从事一项漆绘瓷砖的任务。首先把n(n+1)/2 块正六边形瓷砖拼成三角形的形状，下图给出了 n=3时拼成的“瓷砖三角形”。然后把每一块瓷砖漆成纯白色或者纯黑色，而且每块瓷砖的正、反两面都必须漆成同样的颜色。
- 有一天小可可突发奇想，觉得有必要试试看这些瓷砖究竟能够漆成多少种本质不同的图案。所谓两种图案本质不同就是其中的一种图案无论如何旋转、或者翻转、或者同时旋转和翻转都不能得到另外一种图案。
- 旋转是将瓷砖三角形整体顺时针旋转 120^o 或 240^o。
- 翻转是将瓷砖三角形整体左右翻动 180^o。
- 一开始，小可可觉得这项实验很有意思，他知道n=2时有两个本质不同的漆绘方案，n=4时也只有四个本质不同的漆绘方案。小可可还把这些漆绘方案画了出来。
- 但是后来小可可发现在变大的过程中，漆绘方案的数目增长很快，在 n=14 的时候，居然有 676080320121725950345755597209 6 种不同的漆绘方案。这果然是一项非常艰巨的实验。因此他决定请你编写程序帮他求解本质不同的漆绘方案数。
- 输入格式: 一个正整数 n(1≤n≤20)。
- 输出格式: 一行正整数，代表问题的解 s。s 不超过 200 位。
- 样例输入　　样例输出
- 3　　20



习题：小明的玩具

- 给定一个由 n 个点和 m 条无向边组成平面图。现在用 c 种颜色给图上的顶点着色，一共有多少种不同的着色方案。如果一种着色方法绕重心旋转能变成另外一种着色方法（包括边也要重合），那么认为这两种着色方案相同。
- 输入格式
- 第一行输入三个整数 n(1≤n≤50)，m(0≤m≤n(n−)/2)，c(1≤c≤100)。
- 接下来 n 行每行输入两个 [−10000,10000] 之间的整数，表示顶点点坐标。
- 接下里 m 行，每行输入两个整数 u,v(1≤u,v≤n) 表示 u,v 之间存在一条边。保证没有重边和自环。
- 输出格式
- 输出不同的着色方案数对 10^9+7 取模的结果。
- 样例输入1
- 5 6 2
  0 0
  1 0
  0 1
  -1 0
  0 -1
  1 2
  1 3
  2 3
  3 4
  4 5
  5 2
- 样例输出1
- 32

样例输入2

```
5 4 2
0 0
1 0
0 1
-1 0
0 -1
2 3
3 4
4 5
5 2
```

样例输出2

```
12
```

提示:首先图形的重心的所有点的 x，y 的和的平均值。因为顶点都是整数，如果把坐标放大 n 倍，那么重心必然也是整点了。

一个很重要的结论，整点绕整点旋转只有旋转 90 的被倍数的角度的时候才能到达整点。

几类特殊的计数序列

- **卡特兰数**
- **卡特兰数的递推公式如下：令 h(0)=h(1)=1**

$$h(n) = h(0)h(n-1) + h(1)h(n-2) + \cdots + h(n-1)h(0), \quad n \geq 2$$

$$h(n) = \frac{(4n-2)h(n-1)}{n+1}$$

$$h(n) = \frac{C_{2n}^{n}}{n+1}$$

$$h(n) = C_{2n}^{n} - C_{2n}^{n-1}$$

- **卡特兰数的几种应用**

1. 出栈次序：一个栈(无穷大)的进栈序列为1，2，3，…，n，不同的出栈序列有 h(n) 种 。
2. 凸多边形三角划分：在一个凸多边形中，通过若干条互不相交的对角线，把这个多边形划分成了若干个三角形。对于凸 n 边形，有 h(n) 种不同划分方案数。
3. 给定节点组成二叉树: 给定 n个节点，能构成 h(n) 种不同的二叉树。

排队

- 12 个高矮不同的人，排成两排，每排必须是从矮到高排列，而且第二排比对应的第一排的人高，问排列方式有多少种?
- 我们先把这 12 个人身高从小到大排好序，然后用一个 01 串来标识每个人是在前排还是后排，用 0 表示在前排，用 1 表示在后排。那么应该正好有 6个 000 和 6 个 1。比如 000000111111，010101010101。这样每一排就自然满足了高矮关系。而对于每个后排 1，需要在前面找到一个 0作为他的前排，这样就抽象成为了括号匹配问题。每个 1需要在前面有一个 0 匹配。

Stirling 数

1. 第一类 Stirling 数 $S_1\ (p,k)$ 表示将 p 个不同元素构成 m 个圆排列的数目。递推公式为

$$S_1\ (p,k) = (p-1)S_1\ (p-1,k) + S_1\ (p-1,k-1), 1 \le k \le p-1$$

边界条件：

$$S_1\ (p,0) = 0,\ p \ge 1, S_1\ (p,p) = 1\ p \ge 0$$

2. 第二类 Stirling 数 $S_2\ (p,k)$ 表示：将 p个物体划分成 k个非空的不可辨别的（可以理解为盒子没有编号）集合的方法数。 $k!\,S_2\ (p,k)$是把 p个人分进 k 间有差别(如：被标有房号）的房间(无空房）的方法数。第二类斯特林数的展开式：

$$S_2\ (p,k) = \frac{1}{k!}\sum_{i=0}^{k}(-1)^i C_k^i (k-i)^p$$