目录

> 群论、置换、Bunrnside引理、Pólya定理等概念是群论的内容，也是《组合数学》第十四章Pólya计数的内容（一般认为组合数学属于离散数学hhh），更加详细的概念以及具体的证明参见《离散数学》或者《组合数学》。这里我参考的是机械工业出版社翻译出版的原书第5版《组合数学》。
>
> ---
>
> 由于时间，篇幅等限制，我这里**只会简单列举一些重要性质定理概念以及竞赛中常考的一些用法，精选例题** ，实际上群论等相关定理在算法竞赛中常常用来解决一些计数问题，并且一般都是直接拿去使用的，常用的有 **Bunrnside 引理、 polya 定理等等。**
>
> **实际上群论里绝大多数概念、证明什么的一点都不了解也对于我们做题没有任何影响，这里涉及的很多概念、证明都不用强求，简单理解就好 ~** 都不会太难，**小学二年级都能看懂**，嘿嘿嘿。

0x00 群的定义

给定一个集合 $G=\{a,b,c,\dots\}$ 和集合上的二元运算 "$*$",如果满足以下条件：

（1）**封闭性**。对于任意的 $a,b\in G,a*b\in G$ 成立。

（2）**结合律**。对于任意 $a,b,c\in G,a*(b*c)=(a*b)*c$ 成立。

（3）**存在单位元**。G 中存在一个元素 e，使得对于 G 中任意元素 a ，都有$a*e=e*a=a$，元素 e 为单位元素。

（4）**存在逆元**。对于 G 中任意元素 a ，恒有一个 $b\in G$ ，使得 $a*b=b*a=e$，则元素 b 为元素 a 的逆元，记作 a^{-1}.

则称集合 G 是运算 "$*$" 下的一个群，记为 $(G，*)$ 若 G 是一个有限集，则称 $(G，*)$ 为有限群，其中有限群元素个数成为有限群的阶。记作 $|G|$ ；若 G 是无限集，则称$(G，*)$ 是无限群。

如果群G中任意两个元素a，b满足交换律，则该群为Abel群。

子群

设 G 在 "$*$" 下是一个群，若 H 是 G 的非空子集且 H 在 "$*$" 运算下也是一个群，则称 $(H，*)$ 是 $(G，*)$ 的子群。

0x10 置换

0x11 置换

置换，就是一个 $1\sim n$ 的排列，是一个 $1\sim n$ 排列对 $1\sim n$ 的映射。其实置换就是给我们一个 n 个数的排列 $1,2,\cdots n$ ，我们通过置换得到 $1\sim n$ 的映射 $p_1,p_2,p_3\cdots p_n$

它们一一对应：$\begin{pmatrix}1\ 2\ 3\ \dots n\\ p_1\ p_2\ p_3\dots p_n\end{pmatrix}$

表示元素 1 被 1 到 n 的某个数 p_1取代，2 被 1 到 n的某个除 p_1 以外的数 p_2 取代 $\cdots n$被 1到 n 中除 $p_1,p_2,p_3,\cdots,p_{n-1}$ 的数 p_n取代。

也可以将置换看作是一个**函数** $f(x)$ ，其中 $f(1)=p_1$ ，$f(2)=p_2\cdots f(n)=p_n$。

其中 $p_1,p_2,\cdots,p_n$ **仍然是** $1\sim n$ 的一个排列，所以很明显不同的置换一共有 $n!$ 种 ， 因为所有排列有一共 $n!$ 种选择方式。

0x12 置换的乘法

我们发现置换实际上就表示为**元素位置的变化**。

例如给定一个四元置换：

序列 $1,2,3,4$ 经过该四元置换以后，变成了 $3,2,4,1$

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

若我们对 $3,2,4,1$ 再进行一次上述置换（同样的位置映射）就得到了 $4,2,1,3$。

置换实际上就表示元素的变化，因此就有**置换的乘法**：

$P_1=\begin{pmatrix}1&2&3&4\\a_1&a_2&a_3&a_4\end{pmatrix}\ P_2=\begin{pmatrix}a_1&a_2&a_3&a_4\\b_1&b_2&b_3&b_4\end{pmatrix}$

则 $P_1*P_2=\begin{pmatrix}1&2&3&4\\b_1&b_2&b_3&b_4\end{pmatrix}$

很明显置换并不支持交换律。

正如**置换**可以看作是一个**函数**，那么**置换的乘积**就可以看作是一个**复合函数**（就是疯狂置换，我们所了解的乘积实际上就是很多个加嘛，就是多次置换...）

0x13 置换的循环与对换

循环是表示置换的一种方法，能反映置换的结构，且便于运算。因此我们把 m 阶循环记为

$\begin{pmatrix}a_1&a_2\dots&a_{m-1}&a_m\\a_2&a_3\dots&a_m&a_1\end{pmatrix}$

特别的，当 $m=2$ 时，2 阶循环 (i,j) 叫做 i 和 j 的对换或换位。（很形象，不解释）

不相交的： 如果两个循环没有公共元素，则称这两个循环是不相交的。

定理13.1： 任何一个置换都可以分解为若干个互不相交的循环置换的乘积，且表示法是唯一的。

即：

$$\begin{pmatrix}a_1,a_2,a_3,a_4,a_5\\a_3,a_1,a_2,a_5,a_4\end{pmatrix}=(a_1,a_3,a_2)\circ(a_4,a_5)\tag{1}$$

如果把元素视为图的节点，映射关系视为有向边，则每个节点的入度和出度都为 1，因此形成的图形必定是若干个环的集合，而一个环即可用一个循环置换表示。

推论13.2： 任何一个置换都可以表示成若干个对换的乘积。因为任何一个循环都可以表示成若干个对换的乘积。

稳定核： 设 $G=\{g_1,g_2,\cdots,g_t\}$ 是 S_n 的子群，若 k 是 $1\sim n$ 中的某个整数，G 中使数 k 固定不变的置换全体所构成的集合，称为数 k 在 G 下的稳定核，或称 G 中使数 k 固定不变的置换群，记为 Z_k。$g_i(1\le i\le t)$ 中不变元的个数记为 $\lambda_1(g_i)$。

Example 13.3： 置换 $(a_1\ a_2\ a_3\ a_4\ \cdots a_m)$：

$(a_1\ a_2\ a_3\ a_4\ \cdots a_m)=(a_1\ a_2)(\ a_2\ a_3)(\ a_3\ a_4)\cdots(\ a_d-1\ a_d)$

0x14 置换群

一个置换规定为一种变换法则，将集合中的一些元素映射成另一些元素。

一般可以认为 “元素” 是一个数组 $\{a_i\}$，对这个数组的变换（如交换某两个元素，翻转等等）就是置换，置换群就是一个置换的集合加上一个 “叠加” 运算（就是两个置换一次操作）。

在置换群的作用下，元素存在**等价关系**。等价关系即满足**自反性**、**对称性**、**传递性**。满足等价关系的元素处于同一个**等价类**中。

置换群的子集同样是一个置换群。

0x20 Burnside引理与Polya定理

有了上面的这些基础知识，我们就可以引入 Burnside 引理了

0x21 Burnside引理

0x21.1 Burnside引理

下面先给出一些不理解也无伤大雅的一些概念，群论里实际上绝大多数概念、证明一点都不了解也对于我们做题没有任何影响，所以这里不用强求。

- **不动点**： 不动点 $c(a_i)$ ： 若某元素在置换 a_i 下不改变，则成它为置换 a_i 的不动点。
- **元素轨道E_k（等价类）**： 一个元素经过置换能得到的所有元素集合（这里元素可以看做一个点，置换可以看作走一条边，轨道就是能走到的所有点的集合）。
- **稳定化子Z_k**： 使操作后这个元素不变的**置换集合**（即这个元素是此集合内**所有置换的不动点**）。
- **拉格朗日定理**： 一个有限群的子群的元素个数必能整除这个群的元素个数。
- **轨道 - 稳定化子定理**： $|E_k| * |Z_k| = G$

由上式即可推出

定理21.1.1：《组合数学》中的 **Burnside 引理**：

设 G 是 X 的置换群，而 C 是 X 中一个满足下面条件的着色集合：对于 G 中所有的 f 和 C 中所有的 c 都有 $f * c$ 仍在 C 中，则 C 中非等价着色数 $N(G,C)$ 由下式给出：

$$N(G,C) = \frac{1}{|G|} \sum_{f \in G} |C(f)|$$

换言之， C 中非等价的着色数等于在 G 中置换作用下保持不变的着色的平均数。

简化一下：

对于一个置换 f ，若一个元素 s 经过 f 后不变，则称 s 为 f 的不动点。

记 f 的不动点个数为 $C(f)$，则对于一个置换群，等价类个数为所有 $C(f)$ 的平均值。

再简化一下：

一个置换群的等价类的个数等于各置换不动点个数的平均值。

说人话：

每个置换的不动点的个数的平均值就是不同的方案数。

Example 21.1 ： 【经典例题 1 】

如图 21.1.1 所示，2×2 方格中每个格子可以选择染上 2 种颜色（红色或白色）。那么总共是 $2^4 = 16$ 种情况。现在要问，如果 不旋转、顺时针旋转 90 度、逆时针旋转 90 度、旋转 180 度后相同的均算成同一种方案，问总共有多少种不同的方案。

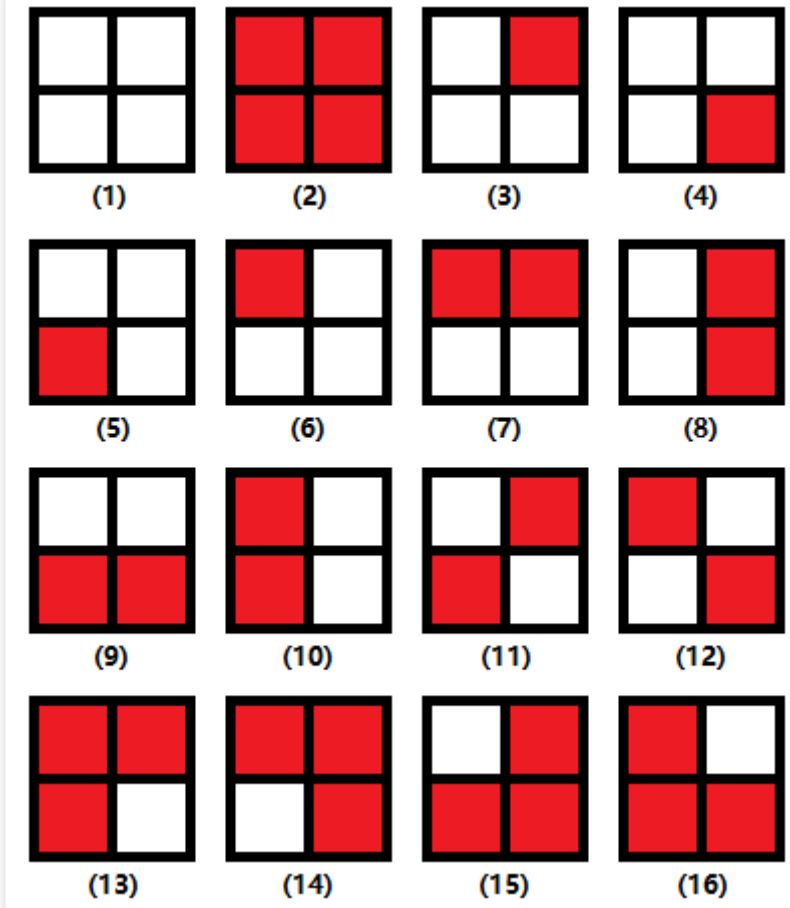

图 21.1.1

Solution

我们将每种旋转认为是一种 "置换" ，定义为f_i，根据上面的描述我们一共有四种置换，f_1 ~ f_4。

f_1=<旋转0°>
f_1=<顺时针旋转 90°>
f_1=<逆时针旋转 90°>
f_1=<旋转 180°>

我们根据上面的定义，设 $C(f_i)$ 来表示 f_i 的这个置换下的不动点集合（经过该置换以后不改变状态）

$C(f_i)$ 表示不动点个数。

则可以得到：

$C(f_1) = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16\}$

$C(f_2) = \{1, 2\}$

$C(f_3) = \{1, 2\}$

$C(f_4) = \{1, 2, 11, 12\}$

然后根据上面我们给出的 **Burnside 引理** 公式：

$N(G, C) = \frac{1}{|G|} \sum_{f \in G} |C(f_i)|$

其中 $|C(f_i)|$ 表示不动点集合 $C(f_i)$ 的个数。

N 表示 使用 m 种颜色给 n 个对象染色的总方案数

$|G|$代表置换个数

得到 $N = \frac{16+2+2+4}{4} = 6$

Example 21.2：【经典例题 2 】

如图 21.1.2 所示，一个3×3的方格，用 10 种颜色给每个格子染色，旋转 0 度、顺时针旋转 90 度、旋转 180 度、逆时针旋转 90 度后，相同的均算成同一种方案，问总共有多少种不同的方案。

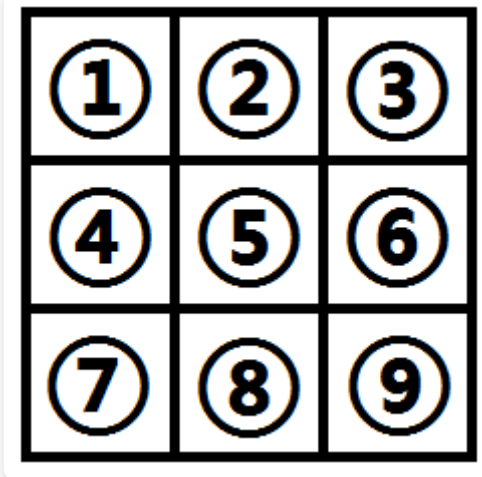

图 21.1.2

我们给每一个格子编一个号，因为每个格子一共有 10 种颜色可供选择，总共 9 个格子，总方案数为 10^9。我们使用 Burnside 引理，$\left|C(f_i)\right|$ 代表在 f_i 这种置换作用下不动点（没有改变状态）的方案个数，置换总共四种，那么我们将这四种置换都列出来：

1）旋转 0 度：也就是我们将这个 3×3 的方格旋转 0 度后，有多少种方案是没有改变状态的，答案显然是 10^9 。也就是说，无论你哪个格子染成什么颜色都没关系，旋转 0 度前后状态不变.

2）顺时针旋转 90 度：我们发现 ①③⑨⑦ 循环变换、 ②④⑧⑥ 循环变换，而 ⑤ 是永远不变。表示成置换群的乘积就是 $(1397)(2486)(5)$ 。那么我们发现，只要在同一个循环中的格子颜色一致，则在这种置换下状态永远不会改变。所以 (1397) 可以取 10 种颜色、 (2486) 同样也可以取 10 种颜色、(5) 可以取 10 种颜色，总方案数为 10^3。

3）旋转 180 度：置换群为 $(19)(28)(37)(46)(5)$（可以看作旋转之后 ① 和 ⑨ 交换等等），总方案数为 10^5。

4）逆时针旋转 90 度：类似顺时针旋转 90 度，总方案数为 10^3。

所以根据Burnside引理公式：

$$N(G,C)=\frac{1}{|G|}\sum_{f\in G}|C(f_i)|$$

可得总方案数为：

$$N=\frac{10^9+10^3+10^5+10^3}{4}$$

0x21.2Burnside 引理的证明

建议跳过（）

该段证明来源

轨道稳定子定理 G 和 X 的定义同上，$\forall x\in X,G^x=\{g|g(x)=x,g\in G\},G(x)=\{g(x)|g\in G\}$，其中 G^x 称为 x 的 **稳定子**，$G(x)$ 称为 x 的 **轨道**，则有

$$|G|=|G^x||G(x)| \tag{2}$$

轨道稳定子定理的证明 首先可以证明 G^x 是 G 的子群，因为

- 封闭性：若 $f,g\in G$，则 $f\circ g(x)=f(g(x))=f(x)=x$，所以 $f\circ g\in G^x$
- 结合律：显然置换的乘法满足结合律
- 单位元：因为 $I(x)=x$，所以 $I\in G^x$（I 为恒等置换）
- 逆元：若 $g\in G^x$，则 $g^{-1}(x)=g^{-1}(g(x))=g^{-1}\circ g(x)=I(x)=x$，所以 $g^{-1}\in G^x$

由群论中的拉格朗日定理，可得

$$|G|=|G^x|[G:G^x] \tag{3}$$

其中 $[G:G^x]$ 为 G^x 不同的左陪集个数。接下来只需证明 $|G(x)|=[G:G^x]$，我们将其转化为证明存在一个从 $G(x)$ 到 G^x 所有不同左陪集的双射。令 $\varphi(g(x))=gG^x$，下证 φ 为双射

- 若 $g(x)=f(x)$，两边同时左乘 f^{-1}，可得 $f^{-1}\circ g(x)=I(x)=x$，所以 $f^{-1}\circ g\in G^x$，由陪集的性质可得 $(f^{-1}\circ g)G^x=G^x$，即 $gG^x=fG^x$
- 反过来可证，若 $gG^x=fG^x$，则有 $g(x)=f(x)$
- 以上两点说明对于一个 $g(x)$，只有一个左陪集与其对应，即 φ 是一个从 $G(x)$ 到左陪集的映射
- 又显然 φ 有逆映射，因此 φ 是一个双射

Burnside 引理的证明

$$
\begin{aligned}
\sum_{g\in G}|X^g| &= |\{(g,x)|(g,x)\in G\times X, g(x)=x\}| \\
&= \sum_{x\in X}|G^x| \\
&= \sum_{x\in X}\frac{|G|}{|G(x)|} \qquad （轨道稳定子定理） \\
&= |G|\sum_{x\in X}\frac{1}{|G(x)|} \\
&= |G|\sum_{Y\in X/G}\sum_{x\in Y}\frac{1}{|G(x)|} \\
&= |G|\sum_{Y\in X/G}\sum_{x\in Y}\frac{1}{|Y|} \\
&= |G|\sum_{Y\in X/G}1 \\
&= |G||X/G|
\end{aligned} \tag{4}
$$

所以有

$$
|X/G|=\frac{1}{|G|}\sum_{g\in G}|X^g| \tag{5}
$$

0x22 Polya 定理

我们发现如果要使用 Burnside引理 求解计数问题，还需要一个一个数不动点的个数，非常难数，若是数据非常大的话基本上就废了，因此我们引入 Polya 定理，提供了一种直接计算不动点的方法，可以快速方便地解决 Burnside引理 难以求解的问题。

我们上面介绍了任意一个置换都可以分解为循环的乘积。

这里定义 **循环** 的 **循环节** 为循环的个数。

例如：

循环 $\begin{pmatrix}1&2&3&4&5\\3&5&1&4&2\end{pmatrix}=(1\ 3)(2\ 5)(4)$（可以理解为13对应31，25对应52，4对应4）

循环 $(1\ 3)(2\ 5)(4)$ 的循环节为 3 。

对于一个置换 f，$C(f)=k^{m(f)}$，其中 k 为颜色数，$m(f)$ 为 f 的循环节。

综上，在置换群中，**等价类个数（方案数）等于所有置换 f 的 $k^{m(f)}$ 的平均数。**

简单解释一下：

我们假设一个置换有 k 个循环，显然每个循环对应的所有位置颜色需一致，而任意两个循环之间选什么颜色互不影响。因此，若有 k 种可选颜色，则该置换 f 所对应的不动点个数为 $k^{m(f)}$ 。我们用它来替换掉 burnside 引理中的 $C(f)$，即 $C(f)=k^{m(f)}$。得到等价类数量（总方案数）为：

$$
N=\frac{\sum_{i=0}^{|G|}k^{m(f_i)}}{|G|}
$$

其中 $|G|$ 表示置换的数目，$m(f_i)$ 表示第 i 个置换包含的循环个数。

我们继续使用最开始提出的经典例题1

Problem 22.1 ： 【经典例题 1 】

如图 22.1.1 所示，2×2 方格中每个格子可以选择染上 2 种颜色（红色或白色）。那么总共是 $2^4=16$ 种情况。现在要问，如果 不旋转、顺时针旋转 90 度、逆时针旋转 90 度、旋转 180 度后相同的均算成同一种方案，问总共有多少种不同的方案。

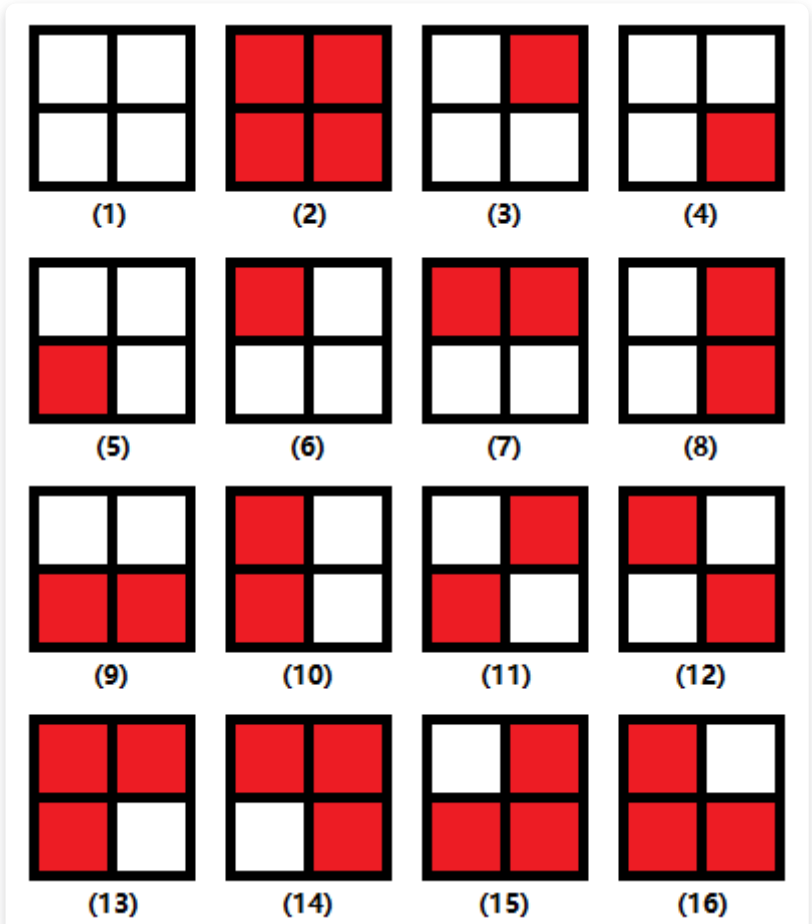

图 22.1.1

Solution

我们根据 Polya 定理可以得到下式：

不旋转

$$f_1=\begin{pmatrix}1&2&3&4\\1&2&3&4\end{pmatrix}=(1)(2)(3)(4),C(f)=k^{m(f)}=2^4=16$$

顺时针旋转 90°

$$f_2=\begin{pmatrix}1&2&3&4\\2&3&4&1\end{pmatrix}=(1\ 2\ 3\ 4),C(f)=k^{m(f)}=2^1=2$$

逆时针旋转 90°

$$f_3=\begin{pmatrix}1&2&3&4\\4&1&2&3\end{pmatrix}=(1\ 2\ 3\ 4),C(f)=k^{m(f)}=2^1=2$$

旋转 180°

$$f_4=\begin{pmatrix}1&2&3&4\\3&4&1&2\end{pmatrix}=(1\ 3)(2\ 4),C(f)=k^{m(f)}=2^2=4$$

这里**旋转** 180° 得到的循环的乘积为 $(1\ 3)(2\ 4)$ ，即代表 1、3 循环置换，互相变换，2、4 循环置换，很明显 1 和 3 的颜色必须相同，2 和 4 的颜色也必须相同，而 1、3 和 2、4 的颜色互不相干，没有影响。

最后我们使用 **Polya 定理** 同样可以得到正确答案：

$$N=\frac{16+2+2+4}{4}=6$$

0x23 特殊模型

0x23.1 旋转同构

模型简述： n个点，每个点移动 k 步（$0 \le k \le n-1$），循环个数为：$\gcd(k,n)$

证明：

1. 当 k 是 n 的约数，显然成立。一个环用 $\frac{n}{k}$ 个，可以分成$\frac{N}{(\frac{N}{k})}=k$ 个循环（$gcd(k,n)=k$）。
2. 当 k 不是 N 的约数，最小的循环长度是：$\mathrm{lcm}(k,n)$，循环使用的端点是：$\frac{\mathrm{lcm}}{k}$个，可以凑成 $\frac{n}{\frac{lcm}{k}}=\frac{n\times k}{\mathrm{lcm}}=gcd(N,k)$ 个。

Problem 23.1.1：循环同构

我们发现经典问题实际上就相当于：圆上有 4 个点，2 种颜色，旋转同构，求方案数。 我们来把它扩展一下：圆上有 n 个点，k 种颜色，旋转同构，求总方案数。

Solution

仔细分析可以发现，当我们旋转 i 个点时，一共会有 $\gcd(i,n)$ 个循环，每个循环会有 $\frac{n}{\gcd(i,n)}$ 个元素。

因此
$ans=\frac{1}{n}\sum_{i=0}^{n-1}k^{\gcd(i,n)}$

0x23.2 对称同构

Problem 23.2.1：对称同构

圆上有 n 个点，k 种颜色，旋转同构，翻转同构，求总方案数。

Solution

旋转同构总共形成 $a=\sum_{i=0}^{n-1}k^{\gcd(i,n)}$ 个不动点。

翻转同构分两种情况考虑：

当 n 为奇数时，对称轴有 n 条，每条对称轴形成 $\frac{n-1}{2}$ 个长度为 2 的循环，1 个长度为 1 的循环（对称轴一定过一个顶点），共 $\frac{n-1}{2}+1=\frac{n+1}{2}$ 个循环。

因此总共形成 $b=nk^{\frac{n+1}{2}}$ 个不动点。

当 n 为偶数时，有两种对称轴。穿过两个点的对称轴有 $\frac{n}{2}$ ，共形成 $\frac{n}{2}+1$ 个循环；不穿过点的对称轴有 $\frac{n}{2}$ ，共形成 $\frac{n}{2}$ 个循环。

因此总共形成 $b=\frac{n}{2}(k^{\frac{n}{2}+1}+k^{\frac{n}{2}})$ 个不动点。

综上，$ans=\frac{a+b}{2n}$

奇数点按边对称： $\frac{n-1}{2}+1=\frac{n+1}{2}$ 个循环。

奇数点按边对称： $\frac{n}{2}$ 个循环。

按点对称： $2+\frac{n-2}{2}=\frac{n+2}{2}$ 个循环。

P4980 【模板】Pólya 定理

给定一个 n 个点， n 条边的环，有 n 种颜色，给每个顶点染色，问有多少种本质不同的染色方案，答案对 10^9+7 取模

注意本题的本质不同，定义为：只需要不能通过旋转与别的染色方案相同。

$n \leq 10^9$

Solution

显然答案为 $ans=\sum_{i=0}^{n-1} k^{\gcd(i,n)}$

但是 $n \leq 10^9$ 数据较大，显然没办法直接暴力计算，考虑优化。

显然可以从 $\gcd$ 上入手。 设 $d_i=gcd(n,i)$ 。显然，所有 d 都是 n 约数。设有 x 个 d 是相等的，那么可以枚举 n 的约数。设枚举到了第 j 个约数 p_j ，那么 $d=n/p_j$ 。我们也很容易发现 $x=\varphi(n/d)$。所以
$ans=\frac{1}{n}\sum_{p|n}\varphi(p)\times k^{\frac{n}{p}}$

时间复杂度 $O(\sqrt{n})$

Code

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
const int N = 5e5 + 7, mod = 1e9 + 7;
const double PI = acos(-1.0);

int n, k;

int qpow(int a, int b)
{
    int res = 1;
    while(b) {
        if(b & 1) res = res * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return res;
}

int get_phi(int n)
{
    int res = 1;
    for(int i = 2; i * i ≤ n; ++ i) {
        if(n % i == 0) {
            n /= i;
            res = res * (i - 1);
            while(n % i == 0) n /= i, res = res * i;
        }
    }
    if(n > 1) res = res * (n - 1);
    return res;
}

```

```cpp
void solve()
{
    scanf("%lld", &n);
    k = n;
    int res = 0;
    for(int i = 1; i * i ≤ n; ++ i) {
        if(n % i == 0) {
            res = (res + get_phi(i) * qpow(k, n / i)) % mod;
            if(i * i ≠ n) {
                res = (res + get_phi(n / i) * qpow(k, i)) % mod;
            }
        }
    }
    res = res * qpow(n, mod - 2) % mod;
    printf("%lld\n", res);
}

signed main()
{
    int t;
    scanf("%lld", &t);
    while(t -- ) {
        solve();
    }
    return 0;
}
```

0x24 Burnside引理与Polya定理的抉择

我们知道 Polya定理 使用起来会比 Burnside引理 更加方便快捷，并且可以解决 Burnside引理 很难解决的大量不动点难以计数问题。那么我们每次只需要使用 Polya定理 就好了嘛?

我们知道 Polya定理 实际上就是 Burnside引理 的延伸拓展，对于 一种特殊情况的特解方法。即若不同循环之间没有任何关联就可以直接使用 Polya定理 解决。但是如果不同循环之间有一定的联系呢？此时我们只能使用 Burnside引理 解决问题。

下面给出两个例题，一个可以用 Polya定理 快速求解，一个只能使用 Burnside引理 ，希望大家可以理解并明白这两个求解方法的使用场景。

0x24.1 可以使用Polya定理

Problem 24.1.1 AcWing 3133. 串珠子

给定 M 种不同颜色的珠子，每种颜色的珠子的个数都足够多。

现在要从中挑选 N 个珠子，串成一个环形手链。

请问一共可以制作出多少种不同的手链。

注意，如果两个手链经**旋转**或**翻转**后能够完全重合在一起，对应位置的珠子颜色完全相同，则视为同一种手链。

Solution

会发现这道题就是 **Problem 23.2.1：对称同构** 。

直接按照上面推出来的公式实现以下即可。

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
```

```cpp
const int N = 5e5 + 7, mod = 1e9 + 7;
const double PI = acos(-1.0);

int n, m;

int qpow(int a, int b)
{
    int res = 1;
    while(b) {
        if(b & 1) res = res * a;
        a = a * a;
        b >>= 1;
    }
    return res;
}

void solve()
{
    int res = 0;
    for(int i = 0; i < n; ++ i) {
        res += qpow(m, __gcd(i, n));
    }
    if(n & 1) {
        res += n * qpow(m, (n + 1) / 2);
    }
    else {
        res += n / 2 * (qpow(m, n / 2 + 1) + qpow(m, n / 2));
    }
    res = res / n / 2;

    printf("%lld\n", res);
}

signed main()
{
    while(~scanf("%lld%lld", &m, &n) && n + m) {
        solve();
    }
    return 0;
}
```

0x24.2 只能使用Burnside引理

Problem 24.2.1 AcWing 3134. 魔法手链

给定 m 种不同颜色的魔法珠子，每种颜色的珠子的个数都足够多。

现在要从中挑选 n 个珠子，串成一个环形魔法手链。

魔法珠子之间存在 k 对排斥关系，互相排斥的两种颜色的珠子不能相邻，否则会发生爆炸。（同一种颜色的珠子之间也可能存在排斥）

请问一共可以制作出多少种不同的手链。

注意，如果两个手链经旋转后能够完全重合在一起，对应位置的珠子颜色完全相同，则视为同一种手链。

答案对 9973 取模。
$1 \leq T \leq 10$
$1 \leq n \leq 10^9$
$\gcd(n, 9973) = 1$
$1 \leq m \leq 10$
$0 \leq k \leq \frac{m(m+1)}{2}$
$1 \leq a, b \leq m$

Solution

这里因为多了很多互斥关系，也就是不同的循环之间有一定有了很多限制，所以不能使用 Polya 定理，只能使用 Burnside 引理 。

本题比上一题简化了，只有循环这一种置换。我们可以发现 n 很大，需要模 9973 ，并且保证 $\gcd(n, 9973) = 1$ 也就意味着我们最后除以 n 的时候可以直接使用乘法逆元。

我们设旋转的距离为 $k = 0, 1, 2, \cdots n-1$。

由于需要使用 Burnside引理，所以我们需要求一下不动点的数量。

设一个点初始位置为 x ，那么每转一次就会转到 $x + k$ 的位置

如图 24.2.1 所示

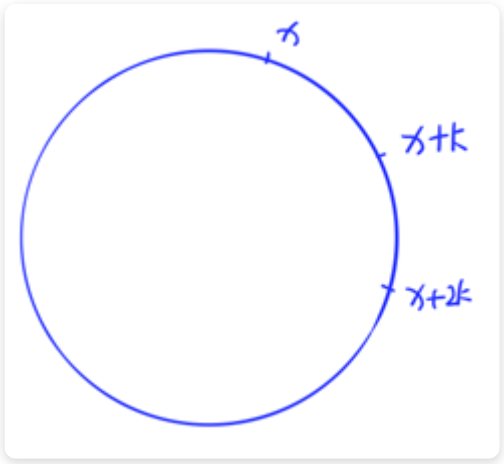

图 24.2.1

那么转多少会重复呢（回到起点，出现 **循环**）

回到起点不一定只赚一圈，可能需要转很多圈，由于点数是有限的，所以我们这样每次走 k 步是一定会重复的。

这里很明显可以得到一个同余方程：

$x + kt \equiv x (\mod n)$
$kt \equiv 0 (\mod n)$

即

$kt = nr + c$
$t = \frac{n}{d}$

其中 $d = \gcd(n, k)$

这也就意味着我们走 $\frac{n}{d}$ 步就会出现 **循环**

即：每一个循环一共有 $\frac{n}{d}$ 个点。

即：每个循环均使用了 $\frac{n}{d}$ 个点，而我们一共有 n 个点，

所以一共会有 $\frac{n}{\frac{n}{d}} = d$ 个循环。

即我们仅有旋转这一种置换操作，会得到 $d = \gcd(n, k)$ 个循环。

我们发现 每个循环均有 $\frac{n}{d}$ 个点，我们可以先分析一个循环，其余的循环与该循环性质相同，分析一个循环即可得到所有的循环的答案。

我们发现了这一个循环的 $\frac{n}{d}$ 个点的一个性质：**我们按照原本的点的编号的顺序看，任意两个相邻的点之间的距离均为 d 。**

证明： $d=\gcd(n,k)$

设 $k'=\frac{k}{d}$，$n'=\frac{n}{d}$。

由于我们每次走都是跳 k 步，然后如果跳过了一圈长度为 n 以后，就 $\%n$。因为 k 和 n 是 d 的倍数，所以我们从起点 x 出发，每次走到的点到起点的距离都必然是 d 的倍数。然后因为我们一共走了 $\frac{n}{d}$ 步，而每一步又都是 d 的倍数，也就是说我们每次走的就是距离可以写成表格，并且我们将 $0,d,2d\cdots$ 做一个映射:

距离	0	d	2d	3d	...	(n / d - 1) * d = n - 1
步数	0	1	2	3	...	n / d
映射	0	1	2	3	...	n / d

也就意味着我们每一步走的距离均为 d 。

综上所诉，该性质得证 □

这个性质也就意味着对于每一个循环（我们当前分析的就是一个循环），我们只需要考虑 d 的倍数的这 $\frac{n}{d}$ 个点即可。

所以我们仅考虑这些点的映射（$0,1,2,3,\cdots$），就会得到一个长度为 $n'=\frac{n}{d}$ 的一个小环（共有 $n'=\frac{n}{d}$ 个点），并且每次跳的距离是 k'，其中 $k'=\frac{k}{d}$，即 k' 与 n 互质。也就可以证明这样跳，一定能遍历到这个小环的所有的点。

小环如图 24.2.2 所示

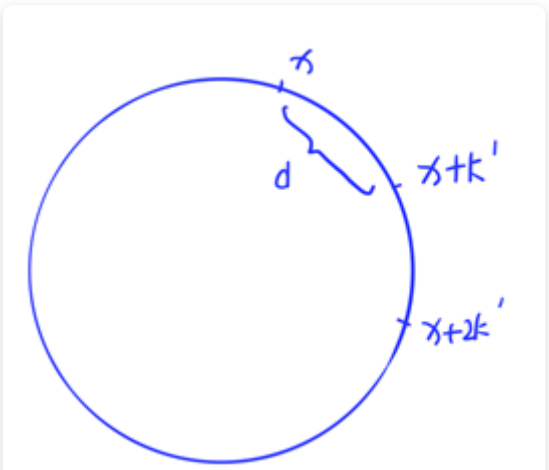

图 24.2.2

因为可以遍历这个小环的所有的点，可以得到：

$k'\times 0,k'\times 1,k'\times 2,\cdots,k'\times(n-1)$ 构成了一个 0 ~ n 的一个简化剩余系。（即该 k' 的序列在 $\mod n$ 意义下还是 0 ~ $n-1$）

证明：

我们使用反证法。

假设$k'\times 0,k'\times 1,k'\times 2,\cdots,k'\times(n-1)$ 不是0 ~ n 简化剩余系，即序列中存在两个下标，$i\neq j$ 且 $k'i=k'j$

$k'i=k'j$

实际上就是：

$k'(i-j)\equiv 0(\mod n)$

因为 k' 与 n 互质，所以可以吧 k' 去掉

即：$i\equiv j(\mod n)$

而 $0\leq i,j<n$。

若 $i\equiv j(\mod n)$，则定有 $i=j$，与前提不符，产生矛盾。

故对于所有的 $k'i$，$k'j$，$0\leq i,j<n$，任意两个数均不相同，故在 $\%n$ 的意义下一定能遍历完 $0\sim n-1$ 里的每一个数，故一定是 $0\sim n-1$ 的一个简化剩余系。

综上所诉，该性质得证 □

由于我们刚刚得到的这一个循环的 $\frac{n}{d}$ 个点的一个性质：**我们按照原本的点的编号的顺序看，任意两个相邻的点之间的距离均为 d 。**

以及另一个性质 $k'\times 0,k'\times 1,k'\times 2,\cdots,k'\times(n-1)$ 构成了一个 $0\sim n$ 的一个简化剩余系，也就意味着在 $\%n$ 的意义下一定能遍历完 $0\sim n-1$ 里的每一个数

而我们仅考虑这些点的映射（$0,1,2,3,\cdots$），就会得到一个长度为 $n'=\frac{n}{d}$ 的一个小环（共有 $n'=\frac{n}{d}$ 个点），并且每次跳的距离是 k'。我们再回代，即乘上一个 d 以后，就意味着可以遍历所有与 x 的距离为 d 的倍数的点。

而最开始我们由得到了仅考虑旋转这一种置换，我们一共会有 d 个循环，每次循环的步数（走的距离）均为 d，也就是每个循环的起点，为 $x,x+d,x+2d,\cdots$。

也就意味着第一个循环的区间为 $[x,x+d-1]$，第二个循环的区间为 $[x+d,x+2d-1]$，以此类推。我们之前说过，由于仅是旋转操作，得到的 d 个循环的解法一摸一样，每一个循环的不动点的个数均相同，即为每一个循环内部的点的颜色按照顺序相同。

例如：第一个循环的颜色为 `1 2 3 4 5` ，则第二个循环的颜色同样为 `1 2 3 4 5` ，以此类推。

具体图形如图 24.2.3 所示：

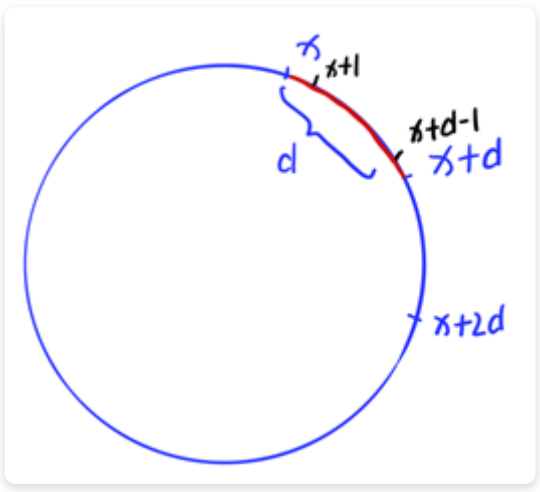

图 24.2.3

因为每一个循环区间内部点的颜色都是相同的，也就意味着每个区间的最后一个点和下一个区间的第一个点相邻，而下一个区间的第一个点和这个区间的第一个点的颜色是相同的，所以我们就可以看作每个区间的最后一个点和该区间的第一个点是相邻的，也就意味着每个长度为 d 的小区间又可以看作是一个循环，也就可以画成一个更小的圈。

也就意味着我们只需要讨论一下这个长度为 d 的圈有多少个染色方案就行啦！

这个染色方案的数量就是这个置换的这个循环的不动点的数量。（因为这个圈固定以后，这一个段就固定了，那么这一小段的这个区间固定了以后，因为整个环分为 d 段，也就是 d 个循环，那么整个环也就固定了）

也就是说我们只需要求一下长度为 d 的，满足要求的这个环的染色方案即可。

那么怎么求解这个方案数呢？我们仅需枚举一下这个环的起点

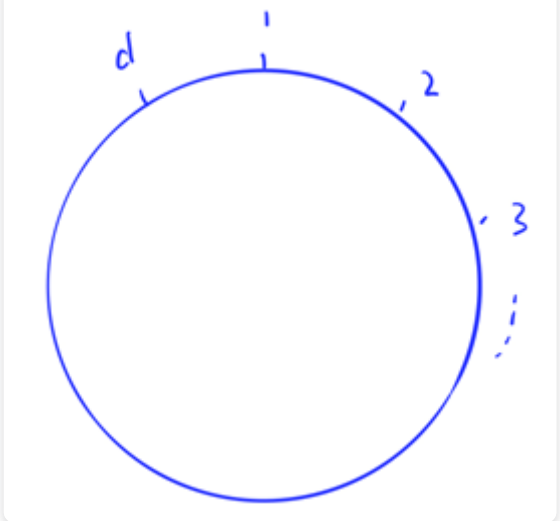

图24.24

我们分情况讨论一下，我们枚举一下 d这个点是那种颜色，因为一共有 m 种颜色，所以我们 $1 \sim m$ 枚举一遍。

我们求一下 d 染乘 1 的所有方案， d 染 2 的所有方案，$\cdots$ 到 d 染 m 的所有方案。

若 d 染的颜色为 i ，我们可以使用 DP 来求解这个答案。

$f[i][j]$ 表示的是染完了前 i 个珠子的颜色，并且最后一个珠子的颜色为 j 的所有方案的数量。

例如我们将 d 染为 i，即 `f[0][i] = 1`，其余均为 0：`f[0][1] = f[0][2] = ...f[0][m] = 0;`。（d 就是 1 前面的这个珠子，编号为 0）

暴力转移即可：

```
f[i][j] += vis[k][j] ? f[i - 1][k] : 0;
```

其中 `vis[k][j]` 表示 k 和 j 是否互斥（由题目输入），如果不互斥的话说明 k 和 j 可以相邻。

最后的答案：分类讨论：

若 d 染色为 1 ，答案为 `f[d][1]`。其余同理。

但是由于本题的 $n \leq 10^9$ ，暴力转移无法通过，所以我们可以使用矩阵乘法来优化。

设 $F[i] = f[i][1...m]$

则 $F[i] = F[i-1] * M$

其中转移矩阵 M 就代表 若k 和 j 可以相邻，互不排斥，就为 1 ，否则就为 0 。（该矩阵的转移方程实际上展开以后就是上面的DP转移方程）

则答案 $F[d] = F[0] * M^d$，我们可以来使用快速幂优化。

总时间复杂度为 $O(M^3logn)$

最后一个问题，因为 n 很大，所以我们枚举 k 的时候不可能直接枚举。

我们发现 k 的唯一作用就是得到 $d = \gcd(n,k)$，而我们非常容易就可以发现很多 $gcd(n,k)$ 是相同的，因此我们就可以将 k 按照所有的最大公约数分类，也就是直接枚举$\gcd(n,k)$ 即可，也即是说我们只需要枚举 n 的约数即可，那么就意味着最多只会有 $\varphi(n \leq 10^9) \leq 1600$ 种。

则对于所有的 $d = \gcd(n,k)$ 都可以直接使用快速幂来求解。分类之后我们肯定要算一下每一个 d 都包含了多少个 k，换句话说就是一共有多少个 k 满足 $\gcd(n,k) = d$ 呢?

我们发现这就是一个 非常经典的欧拉函数问题。

$\gcd(n,k) = d$

即：$\gcd(\frac{n}{d}, \frac{k}{d}) = 1$ 的个数，也就是在0 ~ $\frac{n}{d}$ 种互质的个数，也就是 $\varphi(\frac{n}{d})$。直接暴力求，复杂度 $O(\sqrt{\varphi(\frac{n}{d})}$。

这里的答案（所有不动点的和）就是 $ans = \sum_{d=0}^{n} F[0] * M^d * \varphi(\frac{n}{d})$

最后的答案就是所有不动点的平均值即不动点个数和除以所有置换的个数。

因为我们这里的置换是旋转，一共有 n 个点，所以一共有 n 种置换

即最后的答案为：$\frac{ans}{n}$，也就是乘上 n 模 9973 的逆元 $ans \times inv(n)$

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>

using namespace std;

const int N = 11, P = 9973;

int m;
struct Matrix
{
    int a[N][N];
    Matrix()
    {
        memset(a, 0, sizeof a);
    }
};

Matrix operator* (Matrix a, Matrix b)
{
    Matrix c;
    for (int i = 1; i ≤ m; i ++ )
        for (int j = 1; j ≤ m; j ++ )
            for (int k = 1; k ≤ m; k ++ )
                c.a[i][j] = (c.a[i][j] + a.a[i][k] * b.a[k][j]) % P;
    return c;
}

int qmi(Matrix a, int b)
{
    Matrix res;
    for (int i = 1; i ≤ m; i ++ ) res.a[i][i] = 1;
    while (b)
    {
        if (b & 1) res = res * a;
        a = a * a;
        b >>= 1;
    }

    int sum = 0;
    for (int i = 1; i ≤ m; i ++ ) sum += res.a[i][i];
    return sum % P;
}

int phi(int n)
{
    int res = n;
    for (int i = 2; i * i ≤ n; i ++ )
        if (n % i == 0)
        {
            res = res / i * (i - 1);
            while (n % i == 0) n /= i;
```

```
        }
    if (n > 1) res = res / n * (n - 1);
    return res % P;
}

int inv(int n)
{
    n %= P;
    for (int i = 1; i < P; i ++ )
        if (i * n % P == 1)
            return i;
    return -1;
}

int main()
{
    int T;
    cin >> T;
    while (T -- )
    {
        int n, k;
        cin >> n >> m >> k;
        Matrix tr;
        for (int i = 1; i ≤ m; i ++ )
            for (int j = 1; j ≤ m; j ++ )
                tr.a[i][j] = 1;
        while (k -- )
        {
            int x, y;
            cin >> x >> y;
            tr.a[x][y] = tr.a[y][x] = 0;
        }
        int res = 0;
        for (int i = 1; i * i ≤ n; i ++ )
            if (n % i == 0)
            {
                res = (res + qmi(tr, i) * phi(n / i)) % P;
                if (i ≠ n / i)
                    res = (res + qmi(tr, n / i) * phi(i)) % P;
            }
        cout << res * inv(n) % P << endl;
    }
    return 0;
}
```

讲完啦！光着一道题就是四五千字的题解，快写死我了，你看懂了嘛？(●ˇ∀ˇ●)

看不懂没关系，点个赞就懂了 =￣ω￣=

0x30 竞赛例题选讲

待更...

- UVA10294 Arif in Dhaka (First Love Part 2)
- P4128 [SHOI2006]有色图
- P4727 [HNOI2009]图的同构记数

待更...

参考资料：

- 群论习题集
- 群论
- 漫谈OI中的群论入门
- Burnside引理与Polya定理
- 置换群学习笔记
- 解题报告 (五) Burnside引理和Polya定理
- 【组合数学】通俗解释 Burnside引理和Polya定理
- Burnside引理与Polya定理
- 群论 学习笔记
- http://www.doc88.com/p-118697800993.html

聪明的你，一定学会 ~~(废)~~ 了吧 (●ˇ∀ˇ●)