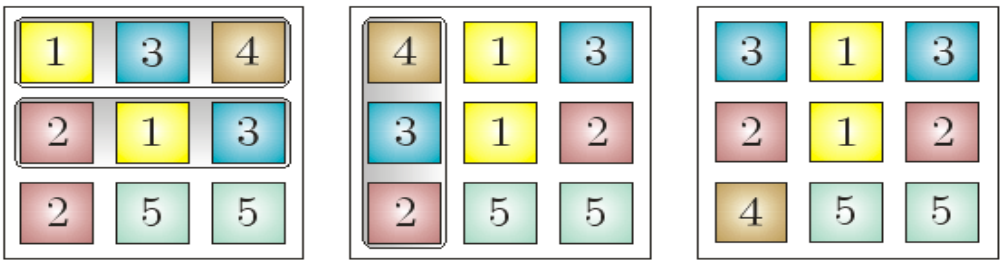
SPOJ241 Arranging the Block 解题报告

试题描述

有一个 n*n 的网格棋盘，每个格子都有一个标号。不会有三个或者三个以上的格子拥有同样的标号。

现在有两种操作：

1. 行操作。把每一行都按任意顺序打乱重排。
2. 列操作。把每一列都按任意顺序打乱重排。



以上执行了两步操作。第一步操作是行操作、第二步操作是列操作。

现在给定初始状态和目标状态，问：能不能从初始状态变换到目标状态？如果能，最少需要几步？

无解情况的分析

我们首先明确广义集合的概念。广义集合和一般集合的意义基本相同，唯一的区别是：广义集合里面允许重复的元素。{1,1}这个广义集合不同于{1,1,1}这个广义集合。

把初始状态中的数放在一起组成一个广义集合 S，目标状态构成广义集合 T。如果 S 和 T 不相等，那么显然无解；如果 S 和 T 相等呢？答案是：肯定有解。

下面我们就将给出“有解”的分析。

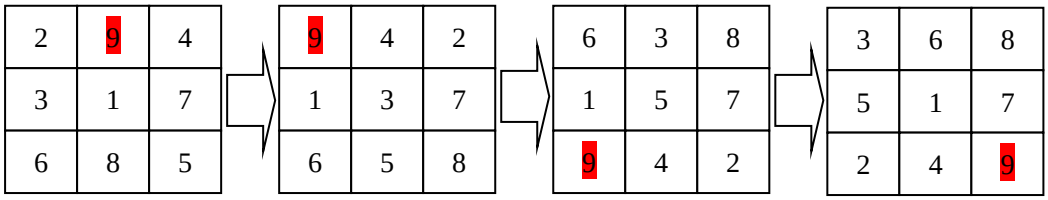
答案最大值分析

定理 1 如果初始状态和目标状态构成的广义集合相等，那么至多只要 3 步就能将初始状态转换成目标状态。

证明 很显然，连续两次进行同意一种操作是毫无意义的。下面我们只考虑“行操作→列操作→行操作”或者“列操作→行操作→列操作”。

这个题目有个很烦人的地方：可能有 2 个格子具有相同的标号。我们实际上只要证明，当所有格子标号都不同时，定理成立即可。（如果有相同标号的格子，可以强行把他们改得不同。）

现在我们考虑：所有格子标号都不同，执行“行→列→行”操作必然可以把初始状态变到目标状态。



初始状态　　目标状态

在上面例子中，注意 9 这个数。它的运动轨迹是(1,2)→(1,1)→(3,1)→(3,3)。

所谓的“行操作”就是把某个数的列坐标变一下，比如(1,2)→(1,1)，就是把列坐标 2 改成 1；所谓“列操作”就是把某个数的行坐标变一下，比如(1,1)→(3,1)，就是把行坐标 1 改成 3。

9 这个数字的初始坐标是(1,2)，目标坐标是(3,3)，因为采用“行-列-行”的方式操作，所以它的轨迹必然是：

(1,2)→(1,y)→(x,y)→(3,3)

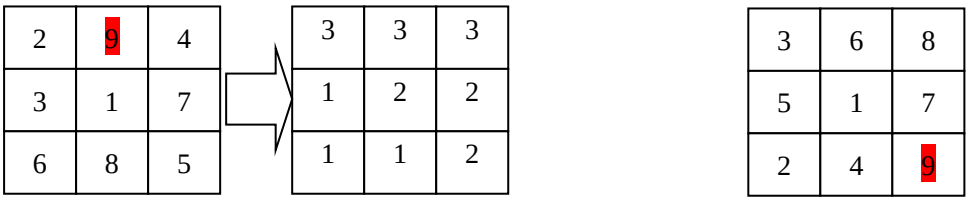
注意到(x,y)→(3,3)这一次是“行操作”，而行操作只能改变列坐标，也就是行坐标保持不变。即 x=3。所以 9 的轨迹必然是：

(1,2)→(1,y)→(3,y)→(3,3)

换句话说，只要第一步“行操作”中的 y 确定了，9 的整个轨迹就确定了。

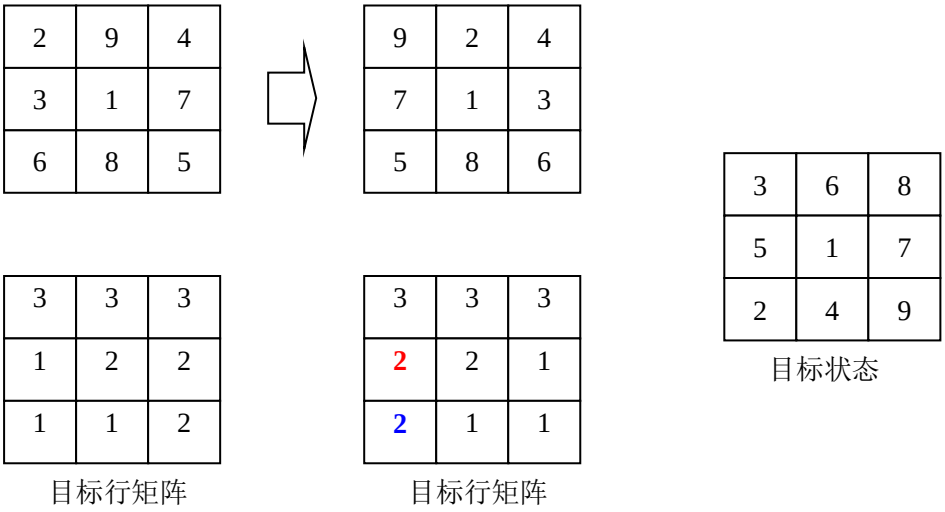
特别的，**第二步“列操作”必然是把数从初始状态中所在的行移动到目标状态中所在的行**。

我们把每个数用它在目标状态中的行号替代，就得到一个“目标行矩阵”：



初始状态　　目标行矩阵　　目标状态

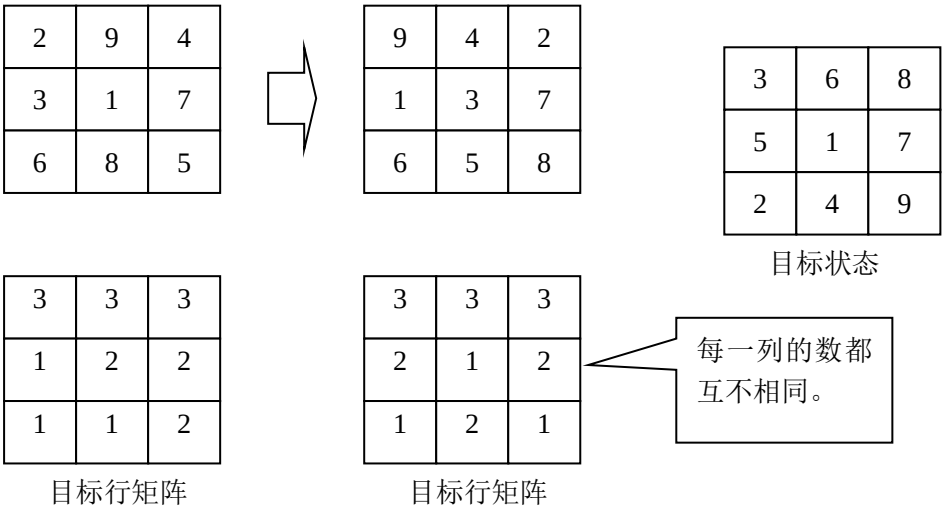
我们考虑进行一次“行操作”：



这样进行了一次行操作之后，新的目标行矩阵中，第一列出现了两个 2。前面我们提到了**“第二步‘列操作’必然是把数从初始状态所在行移动到目标状态所在行”**。也就是说在进行了第一次行操作之后，接下来我们要把每个数移动到他们相对应的行上面去。

可是在上面的例子中，第一列出现了两个 2。这两个数都要求在接下来的操作中立即被移动到第二行。通过一次“列操作”把同一列的两个数移动到同一行是不可能的！所以我们如果想要在 3 步之内完成移动，就不能采取上面的方案。

换句话说，我们第一步采用的行操作应该使得：在目标行矩阵中，每一列的数互不相同。比如：



如果第一步操作保证了目标行矩阵每一列的数都互不相同，那么第二步列操作就可以把每个数交换到他们的目标行；第三步行操作再把每一行内的数排好顺序。整个操作就完成了。

下面的问题是：是不是一定可以找到一种行操作方式，使得操作后目标行矩阵的每一列都互不相同？

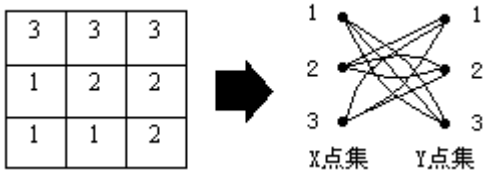
我们观察初始状态的目标行矩阵有如下特点：

1. n*n 的矩阵里面填了 1~n 这些数字。

2. 每个数出现 n 次。（总共有 1~n 这 n 个数，每个数出现 n 次，正好填满 n*n 的矩阵）

我们要对这个矩阵执行“行操作”，使得操作之后每一列的数刚好组成{1,2,…,n}这个集合。

构造二分图 G=(X,Y)。X=Y={1..n}。X 代表第 1, 2, ……, n 行，Y 代表数字 1, 2, ……, n。初始状态的目标行矩阵中，第二行的广义集合是{1,2,2}，所以就在 2∈X 和 1∈Y 之间连一条边、在 2∈X 和 2∈Y 之间连两条边。第一行、第三行类似，如下：（这个二分图不是一个简单图，两点之间可以有多条边连接）



“行操作”可以这样来执行：

从每一行选出一个数来，构成第一列，把这些数删除；然后再从每一行中选出一个数来，构成第二列，把这些数也删除；……如是反复，直到 n 列都被确定下来。一次“行操作”就执行完毕了。

从“每一行选一个数出来构成第一列”，实际上就是在上面的构造的二分图中找一个完备匹配；把这些匹配边删除，再找一个完备匹配；……如是反复。

我们只需要证明：以上构造的二分图可以不断的找匹配、删匹配、再找匹配……直到所有的边都被删除为止。

因为目标行矩阵中每一行有 n 个数、每个数出现 n 次，所以二分图 G=(X,Y)的一个特点是：每个点的度都等于 n。

找到一个匹配后，将其删除，所有点的度都变成 n-1；再找一个，再删除，所有点的度都变成 n-2；……也就是说任意时刻，图中的每个点度都相等。

我们只要证明：如果二分图的每个点度都相等，必然存在完备匹配。

这个证明就非常容易了。

在二分图 G=(X,Y)中，对于 X 的一个子集 S，如果一个顶点 v 与 S 中的某一个顶点相邻，那么就称 v 与 S 相邻。所有与 S 相邻的顶点组成的集合称之为 S 的邻集，记为 A(S)。显然 A(S)是 Y 的子集。

设所有点的度都等于 d，考虑 X 的任意子集 S，从 S 发出的边有 d|S|条；这 d|S|条边都被 A(S)集合接收了。但 A(S)集合总共从 X 集合收到了 d|A(S)|条边，也就是说：d|A(S)|≥d|S|。即|A(S)|≥|S|。

有一定图论基础的读者应该想到了以下著名定理：

Hall 定理 二分图 G=(X,Y,E)有完备匹配的充要条件是：对于 X 的任意一个子集 S 都满足|S|≤|A(S)|。

至此整个命题得到证明。至多 3 步就能完成从初始状态到目标状态的转换，并且用“行-列-行”的操作形式是肯定可行的。

其他情况分析

因为答案<=3，所以只有 0，1，2，3 几种可能。

什么时候答案等于 0？

如果初始状态和目标状态完全一样，那么答案等于 0。

什么时候答案等于 1？

如果初始状态的每一行都和目标状态的对应行构成的广义集合都相等，那么用 1 次行操

作即可；如果初始状态的每一列都和目标状态的对应列构成的广义集合都相等，那么用 1 次列操作即可。

什么时候答案等于 2?

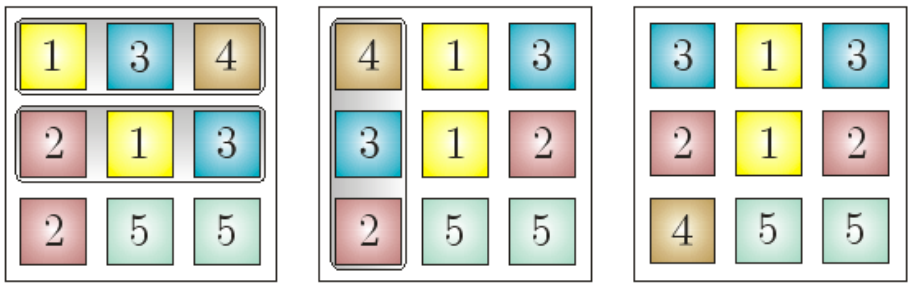
如果你发现初始和目标状态不满足上面提到的任何一组条件，那么答案就只有可能是 2 或者 3 了。我们看看什么时候可以用 2 步操作实现从初始到目标状态的转换。

首先考虑“行操作-列操作”的操作方式。(“列-行-列”类似)

如果一个数的初始位置是(x_1,y_1)，目标位置(x_2,y_2)。那么该数的行动轨迹必然是：

$(x_1,y_1)\rightarrow(x_1,y_2)\rightarrow(x_2,y_2)$

现在的问题是：如果一个数出现了两次，那么目标位置就无法确定：



start of minute 1　　start of minute 2　　end of minute 2

数字 1 在初始状态中的位置是(1,1)和(2,2)；在目标状态中的位置是(2,1),(2,2)。那么到底应该是(1,1)→(2,1), (2,2)→(2,2)还是(1,1)→(2,2), (2,2)→(2,1)呢？

如果采用第一种配对方法，则：

(1,1)→(1,1)→(2,1), (2,2)→(2,2)→(2,2)

两个中转点是(1,1)和(2,2)，记为集合 A={(1,1), (2,2)}。

采用第二种配对方法，则：

(1,1)→(1,2)→(2,2), (2,2)→(2,1)→(2,1)

两个中转点是(1,2)和(2,1)，记为集合 B={(1,2),(2,1)}。

所以对于数字 1，我们有两种配对方案；两种方案对应的中转点集合是 A_1 和 B_1。

对于数字 4，因为初始和目标位置唯一，所以它的中转站肯定是(1,3)，不妨令 $A_4=B_4=\{(1,3)\}$。

这样对于每个数字 v 我们都可以构造出 A_v 和 B_v。我们的任务是给每个数字选择一个方法（A 或者 B），使得被选中的这些集合互相都没有交集。(因为中转点不能相等)

这个模型可以设法转化成 2-SAT 问题（即二判定性问题）。

定义布尔变量 T_v，如果数 v 选了 A 方案，则 T_v=True；如果数 v 选了 B 方案则 T_v=False。

如果集合 A_x 和 A_y 有交集，那就表示他们不能同时被选中，表示为$!T_x$ or $!T_x$。

如果集合 A_x 和 B_y 有交集，那就表示他们不能同时被选中，表示为$!T_x$ or T_x。

如果集合 B_x 和 A_y 有交集，那就表示他们不能同时被选中，表示为 T_x or $!T_x$。

如果集合 B_x 和 B_y 有交集，那就表示他们不能同时被选中，表示为 T_x or T_x。

把所有的形如$(T_?+T_?)$这样的逻辑变量全部用&符号连接起来，就得到了一个逻辑表达式。

对于形如$(X_?$ or $X_?)\&(X_?$ or $X_?)\&\cdots\cdots\&(X_?$ or $X_?)$的逻辑表达式，判断存不存在一组逻辑变量$\{X_?\}$的取值使得该逻辑表达式成立，称之为 2-SAT 问题（二判定性问题）。

判断 2 步之内可不可以把初始状态转化成目标状态，实际上就被转化成了一个 2-SAT 问题。

关于 2-SAT 问题的解法，请参看 2003 年集训队员伍昱的论文。(其中涉及到对称性分析、有向无环图、以及拓扑序，比较复杂)

什么时候答案等于 3？

如果你发现以上针对无解、答案等于 0、答案等于 1 和答案等于 2 的分析全部都对输入数据无效，那么答案就是 3 了。☺

小结

这个题目异常复杂。

首先需要很强的猜想、归纳能力，发现答案不大于 3 这个事实。

证明答案不大于 3 需要大量的观察、转化和一定的图论知识；判断答案是否等于 2 需要极强的转化能力，还需要掌握 2-SAT 的模型和解法。

附录

原题可参考(http://spoj.sphere.pl/LIGA04/problems/BLOCKS/)

SPOJ Problem Set

241. Arranging the Blocks

Problem code: BLOCKS

A group of n children are playing with a set of n^2 flat square blocks. Each block is painted from above with one colour, and there are no more than 2 blocks of each colour. The blocks are initially arranged in an n x n square forming some sort of picture.

The children have been provided with some other n x n picture and asked to rearrange the blocks to that form. Since this is not really what they enjoy doing most, they intend to solve the task together and spend as little time on it as possible. Thus, every minute each child chooses a single 1 x n row or n x 1 column of blocks to rearrange. This row/column may never intersect with rows/columns chosen by other children in the same minute. A child takes one minute to perform any rearrangement (permutation) of the blocks within its row/column it likes.

Determine whether the children can perform their task of converting one block image into the other, and if so -- find the minimum possible time in minutes required to achieve this.

Input

The input starts with a line containing a single integer t<=200, the number of test cases. t test cases follow. Each test case begins with a line containing integer n (1<=n<=500). The next n lines contain n integers $P_{i,j}$ each, forming a bitmap matrix representing the colours of the blocks in their initial configuration ($1<=P_{i,j}<=n^2$). The following n lines contain n integers $Q_{i,j}$ each, corresponding to the matrix for the final configuration ($1<=Q_{i,j}<=n^2$).

Output

For each test case output a line with a single non-negative integer corresponding to the number of minutes required to transform matrix P into matrix Q, or the word no if no such transformation is possible.

Example

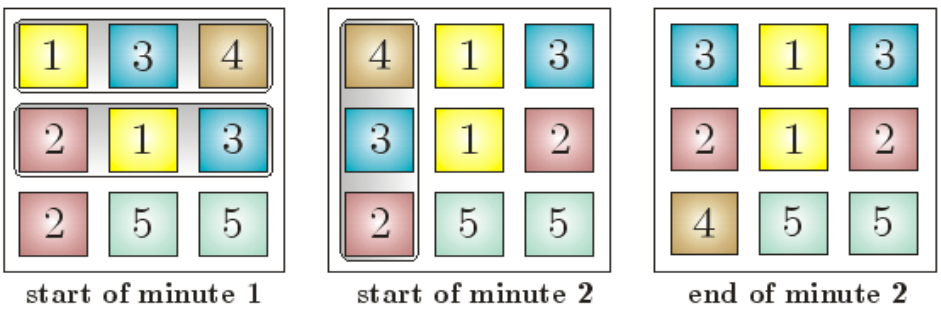
Input:

```
3
3
1 3 4
2 1 3
2 5 5
3 1 3
2 1 2
4 5 5
3
1 2 3
4 5 6
7 8 9
1 5 6
4 2 9
7 8 3
2
1 2
1 2
1 3
1 2
```

Output:

```
2
1
no
```

The actions taken in the first test case are illustrated below.



Warning: enormous Input/Output data, be careful with certain languages