1 Single-Crossing（qoj10083）

1.1 题目大意

给两个正整数 n, m，然后给出 n 个 $[1, m]$ 的排列，要求把这 n 个排列重排，使得满足以下条件：

假设重排后第 i 个排列是 a_i，那么不存在 $1 \le i < j < k \le n; 1 \le x, y \le m; x \ne y$ 使得 x 在第 i 个排列所处的位置比 y 小，在第 j 个排列所处的位置比 y 大，在第 k 个排列所处的位置比 y 小。

也就是说，任意一对 (x, y) 的相对顺序至多只能改变一次。

若无解，输出 -1，否则输出重排的方案。

1.2 数据范围

$nm \le 10^6$。

1.3 解题过程

首先考虑怎么判定一种重排方式是否合法，也就是写一个 spj。为了方便描述，以下假设第一个排列满足 $p_i = i$。

那么条件其实就是，不存在 $2 \le i < n; 1 \le x < y \le m$ 满足 x 在的 i 个排列的位置在 y 的后面，在第 $i+1$ 个排列的位置在 y 的前面。

我们可以枚举 i，然后这就变成了一个简单的二维数点问题（从后往前扫描线第 i 行，用树状数组维护第 $i+1$ 行在它前面的最大值）。

回到原问题，我们观察一下，可以发现最终的答案至多只有两个（没有相同排列的情况下），这是因为，我们假设我们枚举每一对 (x, y)，第一对会把 n 个排列分成两部分，我们可以任选一部分放在前面。而之后的 (x, y) 只能把前面分出来的一个部分再细分，而细分出来的两个部分是可以得知哪个在前哪个在后的，因此最终一定可以得到一个确定的序列。

直接实现这个做法可以做到 $O(nm^2)$ 的复杂度，无法通过。

不过经过一些观察我们发现，有效的 (x, y) 至多只有 $n-1$ 个，因为分裂至多执行 $n-1$ 次，所以我们每次对于一个部分任选两个排列出来，找到它们任意一对相对顺序不同的 (x, y)（这一点可以通过找这两个排列第一个不同的位置做到），就可以做到 $O(n^2 + nm)$ 的复杂度。

直接实现这个做法已经可以通过（我也不知道为啥，可能是如果每次分的大小比较平均复杂度是 $O(nm \log n)$ 的），不过我们还是看看怎么做到 $O(nm \log n)$。

注意到，如果我们可以找出第一个排列，那么后面就简单了，因为两个排列不同，一定有一对 (x, y) 在这两个排列中的相对顺序不同，所以利用这个做排序就可以了，具体来说的话就是，对于两个排列 p, q，我们找到任意一个在它们中相对顺序不同的 (x, y)，然后根据第一个排列中 (x, y) 的顺序就可以确定哪个在前，哪个在后了。

我们虽然不能找出第一个排列，但是只要我们能找到任意一个位置上的排列是什么，以及它前面的排列有哪些，就可以分成两半做上面这个做法了。这个是比较容易的，我们套用分裂的做法，不过每次只往较小的那一个部分去递归，就可以实现这个要求。

时间复杂度 $O(nm + nm \log n + nm \log m) = O(nm \log nm)$。

1.4 参考资料

无