# Sereja and Matrix Division - 解题报告

龚林源

## 1 Description

给定一个 $N \times M$ 的矩阵，你需要从中选出 $C$ 个子矩阵，使得这 $C$ 个子矩阵的并集所组成的多重集 等于 所有不在这些子矩阵中的元素组成的多重集。

请找到满足条件的尽量小的 $C$，并输出对应的方案。

## 2 Constraints

- 每个测试点的 $N = M = 100$
- 共有 $500$ 个测试点，总时限 $50s$
- 矩阵中的元素值 $\in [1, 1000]$，且随机分布
- 保证存在方案

## 3 Tags

- Challenge
- 贪心
- 随机化

## 4 Solution

这是一道 Challenge 题，选手的程序输出的 $C$ 值越小，分数就越高。在题目规定的时间限制和数据范围下，暂时还找不到有效的能在一般情况下求出精确最优解的算法。因此这篇题解主要讨论我的近似算法。这个算法在 Codechef 上能跑出不错的成绩，到 2015 年 11 月 5 日为止，比我最好一次提交的得分更高的程序都针对 Codechef 上的测试数据 做了特殊的优化，不具有一般性和可推广性。接下来我简单介绍一下自己的算法。

### 4.1 找出一组初始可行解

记录下对于每个 $k$，满足 $a_{i,j} = k$ 的所有有序数对 $(i, j)$；然后对于每个 $k$，随机选出一半的这样的有序数对；这样就能得到一组初始可行解了。令初始可行解中被选中的元素集合为 $S_0$。

## 4.2 随机调整当前的解

遍历矩阵上所有元素。假设遍历到 $a_{xx,yy} = num$。从 $S_0$ 中找出所有满足 $a_{x0,y0} = num$ 的 $(x0, y0)$，如果 $(xx, yy)$ 和 $(x0, y0)$ 目前一个被选一个没有被选，此时我们如果交换它们被选或没有被选的状态，得到的仍然是一组可行解。

找出这么一对点之后，何时应该交换、何时不应该交换它们的被选定的状态 $in_{x,y}$ 呢？我尝试过许多不同的估价，发现效果最好的是根据它们左右两边的元素进行估价。具体估价函数是这样的：

$$h(x, y) = (in_{x,y} \ xor \ in_{x,y-1}) + (in_{x,y} \ xor \ in_{x,y+1}) \tag{1}$$

对于一个元素 $(x, y)$，它的 $h(x, y)$ 越大，说明改变它的 $in_{x,y}$，能让被选中的点集在水平方向上更加集中，从而使得最后我们选出的矩形数目尽量小。

- 若 $h(xx, yy) + h(x0, y0) > 2$，说明这次交换能起到优化效果，则执行这次交换
- 若 $h(xx, yy) + h(x0, y0) < 2$，说明这次交换会使原状态退化，不执行交换
- 若 $h(xx, yy) + h(x0, y0) = 2$，从短期来看这次交换是没有必要也没有坏处的，可以以一定的概率交换——经测试，当这个概率为 $50\%$ 的时候是最优的。

反复执行以上操作，直到程序运行时间快要达到题目的时间限制为止（俗称“卡时”）。

我也尝试过用模拟退火算法来代替这样的随机调整，结果效果反而不如直接这样做。对此感到好奇的同学可以自己尝试一下。

## 4.3 根据选中的元素集合输出答案

前面我们已经得出了要选中的元素集合，接下来就是要把这些元素用最少数目的矩形框起来。这是一个十分简单的子问题。可以使用贪心算法解决：逐行遍历这个矩阵的每一行，碰到一横排连续的被选中的元素，就不断向下拓展，直到不能拓展为止；把产生的矩形记录到答案中，然后标记这些点为已经被选过即可。因为每个被选中的元素都必须被包含在至少一个子矩形中，所以这个贪心算法的最优性是显然的。并且这个算法能在 $O(N^3)$ 内完成。

## 4.4 一个关键的优化

要让矩形越少越好，那么单个的矩形肯定是越大越好。如果我们能在算法开始之前先选走一个较大的矩形，那么不仅可能可以略微优化答案，还能减小后面随机调整时涉及到的元素数目。

接下来的问题是：找出一个最大的矩形，使得里面每一种元素的数目都不超过它在大矩阵中总数目的一半（否则肯定无解）。然而这需要较高的复杂度（我目前只想到 $O(N^4)$ 的做法），因此我们不妨退一步——规定这个子矩形左边界与大矩形的左边界重合。这样就可以在 $O(N^3)$ 的复杂度内解决这个问题了：

枚举上边界 $U$，令初始的右边界 $R$ 为 $M$，令初始的下边界 $D$ 为 $U - 1$，记录这个矩形中每个元素的数目。逐行向下移动 $D$，维护矩形中每个元素的数目。每次发现有某种元素的数目超过该种元素总数目的一半，就不断向左移动 $R$ 直到该种元素的数目不过半为止。在移动 $D$ 和 $R$ 的过程中记录矩形的最大面积和此时矩形的位置。直接记录到答案中，在之后的算法中都不再考虑这个矩形中的元素。

这样可以大幅度减少随机调整时涉及的元素数目，从而在总时间不变的情况下能调整出尽量优的解。

### 4.5 总结

这道题是一道比较中规中矩的 Challenge 题，需要我们灵活运用搜索、随机、贪心等思想和算法。