俄罗斯方块 (tetris) 题解

PinkRabbit

1 问题回顾

1.1 问题描述

在一个 $n \times m$ 的方格图中画线，遵循以下规则：

- 可以画线若干次，也可以不画。每次画线前从 c 种颜色种选取一种，从一个没有被画过的方格中心开始画线。
- 只能往上下左右四个方向画线，对应方向上的方格记作目标方格。如果目标方格**没有被画过**，则可以经过两个方格公共边的中点，直接画到方格中心；如果目标方格已经被画过了，则它必须被画上**贯穿整个方格的线**并且方向与将要画过去的线**垂直**的时候，才能画过去，并且要直接穿过整个方格，再继续考虑下一个方格的情况；最后一种情况是目标方格是本次画线的起笔方格，画过去代表本次画线结束；当然，不能画到整个方格图外面去。
- 必须在某个没有被画过的方格或这次画线起笔处结束此次画线。

方格图中有一些位置不能被画到。

询问在这些限制下，最终能画出多少种本质不同的图。

当 $\mathrm{op} = 0$ 时，两张图本质相同当且仅当不考虑坏掉的方格，它们看起来相同 (每个位置上的线条方向和颜色均相同)。

当 $\mathrm{op} = 1$ 时，两张图本质相同当且仅当不考虑坏掉的方格，它们看起来相同，或旋转 $180°$ 后看起来相同。

为了方便，请输出答案对 $998,244,353$ 取模的结果。

1.2 子任务

***Subtask* 1(10 *points*)**：$n = 1$，$\mathrm{op} = 0$，没有不能画到的方格。

***Subtask* 2(12 *points*)**：$c = 1$，$\mathrm{op} = 0$。

***Subtask* 3(10 *points*)**：$c = 1$，$\mathrm{op} = 1$。

***Subtask* 4(13 *points*)**：$m \le 5$，$\mathrm{op} = 0$。

***Subtask* 5(28 *points*)**：$\mathrm{op} = 0$。

***Subtask* 6(9 *points*)**：$n \bmod 2 = 0$，$\mathrm{op} = 1$。

***Subtask* 7(9 *points*)**：$n \bmod 2 = 1$，$m \bmod 2 = 0$，$\mathrm{op} = 1$。

***Subtask* 8(9 *points*)**：$n \bmod 2 = 1$，$m \bmod 2 = 1$，$\mathrm{op} = 1$。

对于所有数据，$1 \le n, m \le 9$，$1 \le c \le 10^6$，$\mathrm{op} \in \{0, 1\}$。

2 题解

为了表述方便清晰，下文中定义**闭线**为在起始点结束的画线，**开线**定义为不在起始点结束的画线。

2.1 算法一: *Subtask* 1

限制: $n=1$，$\mathrm{op}=0$，没有不能画到的格子。

$n=1$，即方格图仅有一行，且 $\mathrm{op}=0$，不需要考虑旋转对称的限制。

进一步地，发现在这个方格图中无法画出闭线，仅能画出开线。

因为画线的顺序对最终的图像没有影响，假定是从左至右画的线。

考虑令 a_i 为当 $m=i$ 时的答案，则不难发现如下递推式:

$$a_m = \begin{cases} 1 & ,0 \le m \le 1 \\ a_{m-1} + c \times \sum_{i=0}^{m-2} a_i & ,m \ge 2 \end{cases}$$

因为 m 不大，直接按照递推式计算即可。

代码长度大约在 500B 以内，实现难度低。

期望得分: 10 分。

2.2 算法二: *Subtask* 2

限制: $c=1$，$\mathrm{op}=0$。

对问题进行一个转化: 枚举每一种放置方案：在 $n \times m$ 的方格图中的每一个能画到的方格放置如下 12 种**纹路**中的 1 种，并使得相邻方格的公共边有着相同的**状态** (是否有纹路经过)。最后统计出这种方案需要多少次画线，假设为 k 次，将答案加上 c^k 即可。不难证明每一种合法的放置方案都对应了唯一一种画线方案 (不考虑画线颜色和顺序)，每一种画线方案也对应了一种合法的放置方案，所以只要知道了放置方案，它所需的画线次数也被唯一确定。

因为 $\mathrm{op}=0$，所以不需要考虑旋转对称的限制。

当 $c=1$ 时，每种合法方案对答案的贡献均为 $1^k=1$，所以不需要统计每种方案需要多少次画线，只需要计算出合法方案的个数即可。

考虑轮廓线状态压缩动态规划，记 DP 状态 $\mathrm{dp}[i][j][S]$ 为从上到下，从左到右考虑到第 i 行第 j 列 (从 0 开始标号)，轮廓线上状态的二进制表示为 S 的合法放置方案数。

每次从 $\mathrm{dp}[i][j]$ 转移至 $\mathrm{dp}[i][j+1]$ ($j<m-1$) 或 $\mathrm{dp}[i+1][0]$ ($j=m-1$)。

转移时考察当前轮廓线状态 S 的第 j 位和第 $j+1$ 位的情况，这对应了轮廓线中 (i,j) 左方和上方的状态。分成有无线经过 $2\times 2=4$ 类讨论，总共 12 种转移方式 (对应 12 种不同的纹路)。在转移时应注意边界情况 ($j=m-1$ 时) 和对不能画线的方格的处理。

时间复杂度为 $\mathcal{O}(nm2^m)$。

空间复杂度可以通过滚动数组优化到 $\mathcal{O}(nm+2^m)$。

代码长度大约在 2KB 以内，实现难度低。

期望得分: 12 分，结合算法一可以获得 22 分。

2.3 算法三: *Subtask* 4

限制: $m\le 5$，$\mathrm{op}=0$。

同样地，按照算法二的思路转化问题。因为 c 不一定等于 1，现在需要统计每一种放置方案所需的画线次数。如果仍然采用算法二的轮廓线状压 DP 的方法，则无法统计出画线次数。考虑更改状态表示，使得能统计出所需画线次数。

仍然考虑 (i,j) 时的轮廓线，但是状态表示需要更加复杂，否则无法统计出次数。不难发现，如果将每次画线的贡献 (即 c) 记在经过的方格中的最下方，最右方的方格上，则可以直接在 DP 的阶段中表示出来: 当一次画线从轮廓线中消失时，转移时就乘上一个 c 的系数。问题在于判断一次画线是否在轮廓线中完全消失 (即原本与轮廓线相交，转移后完全被轮廓线的上半部分包含)，12 种纹路中仅有 3 种是满足与左上方相交但不与右下方相交的，只需要对这 3 种进行考虑即可。

如果将轮廓线上的状态表示为: 0 - 不相交，1 - 开线相交，$2,3,\ldots$ - 不同的闭线相交 (同一条闭线与轮廓线的两个交点的标号相同，不同闭线的标号不同)。则可以判断出转移时的左方、上方的具体状态并进行正确转移。开线不需要重复标号，是因为开线必然不会和轮廓线相交多次，两条开线必然是不同的，不需要刻意区分。(注: 此处所说的开线或闭线与轮廓线相

交，是指单考虑轮廓线和轮廓线上方状态，不考虑轮廓线下方状态，如果考虑下方状态，每一条线都有可能与轮廓线相交超过 3 次)

如此设计状态后，根据 (i, j) 左方和上方的状态，依据无线/开线/相同的闭线/不同的闭线，大致可以分成 7 类本质不同的状态讨论，每一类下方又有 $1 \sim 4$ 种转移，总共大约有 18 种本质不同的转移，具体数目取决于状态表示方法，如果算错/漏/多任何一种都有可能导致答案错误。

m 不大，不用刻意压缩状态，使用 8 进制或其他方法存储状态即可。

同样的，DP 时注意 $j = m - 1$ 时的边界以及对不能画线的方格的处理。

时间复杂度为 $\mathcal{O}(nt)$ 到 $\mathcal{O}(nm^2k)$，t 为总转移数，k 为状态数。不同复杂度取决于状态存储方法和转移处理方法。

空间复杂度取决于状态存储方法，一般不超过 $\mathcal{O}(nm8^m)$。

代码长度大约在 4KB 以内，细节较多，实现难度偏高。

期望得分：13 分，结合算法一和算法二可以获得 35 分。

2.4 算法四：*Subtask* 3

限制: $c = 1$，$\mathrm{op} = 1$。

因为 $c = 1$，所以仍然用算法二的方法计算出总方案数，不妨设为 Ans。然而现在加上了 $\mathrm{op} = 1$ 的限制，需要统计旋转 180° 不同构的图的个数。这时所有方案分成 3 类: 无法旋转的图 (旋转后与不能画到的方格有冲突)、旋转后与自身不同的图、旋转后与自身相同的图。它们的个数不妨分别设为 P_1、P_2 和 P_3。

显然有 $\mathrm{Ans} = P_1 + P_2 + P_3$，而要求的答案是 $P_1 + \frac{P_2}{2} + P_3$。

考虑统计旋转后合法的图的个数，这可以通过简单地将方格图的不能画到的方格的位置对称，然后再进行一遍 DP 得到，得到的方案数不妨设为 Sum。则有 $\mathrm{Sum} = P_2 + P_3$。

最后考虑计算得出旋转对称的图的个数 $\mathrm{Sym} = P_3$，则有最终答案等于 $\mathrm{Ans} + \frac{\mathrm{Sum} - \mathrm{Sym}}{2}$。除以 2 可以用乘以 2 的逆元进行转化。

结合旋转对称的性质以及 DP 的阶段，可以发现如果获得了 DP 正中间阶段的信息 (这里的 DP 指的是将不能画到的方格位置对称后做的 DP)，就能计算出旋转对称的图的个数。这是因为旋转对称的图的前一半和后一

半状态完全相同。如果获知了前一半状态，也确定了后一半的状态。之后只要确定两部分状态 (前一半的某个状态与其对称状态) 的拼接是否合法即可。关于 DP 的正中间阶段，这与 n, m 的奇偶性有关，需要分成 3 类讨论，注意细节处理。

时间复杂度为 $\mathcal{O}(nm2^m)$。

空间复杂度可以通过滚动数组优化到 $\mathcal{O}(nm+2^m)$。

代码长度大约在 3KB 以内，细节较多，实现难度中等。

期望得分: 10 分，结合算法一和算法二可以获得 32 分。

2.5 算法五: *Subtask* 5

限制: $\mathrm{op}=0$。

按照算法三的思路转化问题，但是 m 变大了，如果用 8 进制存储状态都无法开下数组。考虑将状态进一步压缩到连续整数中。

因为 $m \le 9$，所以状态数偏多，需要注意去除非法状态和重复状态，当 $m=9$ 时状态数为 $k_9 = 123,109$。

对于转移的处理，如果边 DP 边处理转移则可能时间超限，需要预处理所有的转移。要注意实现快速的让状态在 8 进制表达和压缩后的表达之间转化的算法，否则预处理的常数可能会太大，标程中用的是哈希链表。当 $m=9$ 时总转移数为 $t_9 = 2,741,310$。

预处理所有转移后就可以直接 DP 了，同样的，注意 $j=m-1$ 时的边界以及对不能画线的方格的处理。

时间复杂度为 $\mathcal{O}(nt)$，t 为总转移数。

空间复杂度可以通过滚动数组优化到 $\mathcal{O}(t)$。

代码长度大约在 6KB 以内，细节较多，实现难度高。

期望得分: 63 分，结合算法四可以获得 73 分。

2.6 算法六: *Subtask* 6 ∼ 8

限制: $\mathrm{op}=1$，对 n, m 的奇偶性有不同的限制。

结合算法四和算法五的思路，使用算法五的方法统计出总方案数 Ans，将不能画到的方格的位置对称，再 DP 一次统计出 Sum，并记录 DP 中途

恰好一半的位置的信息 (取决于 n, m 的奇偶性)，最后通过这些信息计算出 Sym。则答案为 $\mathrm{Ans} + \frac{\mathrm{Sum}-\mathrm{Sym}}{2}$。

注意这里计算 Sym 的方法和算法四有所不同。算法四中因为 $c = 1$，所以只需要统计出放置方案的数量即可，但是这里还需要统计出所需画线次数 k。因为统计的是旋转对称的图的数量，所以颜色也是要对称的，两条**不同且对称**的线的颜色必须相同。而对于正中间阶段，没有跨越轮廓线的线条已经被统计进答案，也包括它们的旋转对称后的线条。所以只需要考虑跨越轮廓线的线条与其对称状态的拼接是否合法，如果合法，会形成多少个可以自由选择颜色的线条即可，这可以通过并查集维护联通块数量来处理。同样地，根据 n, m 的奇偶性需要分成 3 类讨论，这便是分成 3 个子任务的目的。注意实现细节。

时间复杂度为 $\mathcal{O}(nt)$，空间复杂度可以通过滚动数组优化到 $\mathcal{O}(t)$。

代码长度大约在 8KB 以内，细节较多，实现难度高。

期望得分: 37 分，结合算法五可以获得 100 分。