Creek 解析

成员：

陈杭昇（17345008）

蒋青海（17345023）

陈潇（17345012）

所属班级：数学学院（珠海）2017 级 1 班

注释：由于要求中未指明输入的行和列是数字的还是格子的，所以代码中输入的行和列按照**数字的行和列**进行输入。如“.....1.2...4.1....1..0...”对应的行和列为 5 和 5。

解题思路：

（1） “推土机”算法：从上到下，从左到右不断尝试将格子填黑。在遇到不连通的时候，若有 n 个连通分量，则将其中 n-1 个连通分量填黑，只剩下一个连通分量，继续尝试。具体可参看伪代码（一）。

（2） 改进一（非递归）：从最有可能是黑格的格子开始填（确定为黑格的当然最先填），不断尝试将格子填黑。在遇到不连通的时候，若有 n 个连通分量，则将其中 n-1 个连通分量填黑，只剩下一个连通分量，继续尝试。具体可参看伪代码（二）。

（3） 改进二（递归）：根据题目的特点发现某些可以确定的格子后，计算每个数字旁边格子涂黑的概率(opportunity[i][j])，然后从概率最大的不确定格子（charge[i][j]==false）开始尝试涂黑，进行递归。由于可确定格子越多，递归的次数越少，所以一开始根据已知题目的数字特点确定尽可能多的格子会大大减少运行时间。具体可参看伪代码（三）。

（下述三个伪代码，第三个的实现过程作为未注释的主函数，前两个的实现过程放在主函数之前，被注释掉了。）

伪代码：

(一)

1. 输入，初始化相关变量。
2. 若 creek 没有实现完毕：
   1) 若是第一次将格子涂黑，或者之前曾经将格子涂黑且不违规（即涂黑之后，剩下非黑色格子仍连通而且黑格数目不超过数字限制），则在此基础上，得到数组中第一个（顺序按照从左到右，从上到下）不确定颜色的格子 A。
   2) 从 A 开始尝试涂黑，若涂黑后不违规，则记录格子对应的行、列等相关信息。刷新相关数据，返回 2.1）。
   3) 若涂黑后违规是因为超过了数字的限制，则尝试继 A 之后下一个不确定颜色的格子，作为新的 A，返回 2.2）。
   4) 若 A 涂黑后违规，是因为涂黑之后不连通，则算出此时对应的连通分量个数 m，将其中（m-1）个连通分量都涂黑，返回 2.1）。
   5) 若所有不确定颜色的格子尝试涂黑后都违规，则回溯。得到上一个涂黑的格子 C 相关的信息，判断 C 格涂黑的原因是不违规而涂黑，还是之前某个格子 D 填黑之后不连通时处于某个连通分量统一涂黑。若是前者，则放弃将格子 C 涂黑，将其还原成原来未涂黑前的状态，尝试将在 C 之后的下一个不确定颜色的格子作为新的 C，继续 2.2）（此时 C 即是 2.2）中的 A）；若是后者，则将在 D 填黑之后不连通时，C 格对应的连通分量还原为原来的状态，将 D 填黑后原来未填黑的连通分量填黑，返回 2.1）。
3. 输出。

(二)

1. 输入，初始化相关变量。
2. 若 creek 没有实现完毕：
   1) 若是第一次将格子涂黑，或者之前曾经将格子涂黑且不违规（即涂黑之后连通而且黑格数目不超过数字限制），则在此基础上，计算出一系列可能是黑格的格子列{Bn}，作为下一次待定涂黑的格子。

2) 从 B0 开始尝试涂黑，若涂黑后不违规，则记录格子对应的行、列以及{Bn}等相关信息。刷新相关数据，返回 2.1)。

3) 若涂黑后违规是因为超过了数字的限制，则返回 2.2)。【尝试{Bn}中的下一项】

4) 若{Bn}中有某一项 Bk 涂黑后违规，是因为涂黑之后不连通，则算出此时对应的连通分量个数 m，将其中（m-1）个连通分量都涂黑，记录 m 以及剩下的连通分量等相关数据，将剩下的连通分量作为新的{Bn}，返回 2.2)。

5) 若{Bn}中所有的项都尝试涂黑后违规，则回溯。得到上一个涂黑的格子 C 相关的信息，判断 C 格涂黑的原因是不违规而涂黑，还是之前某个格子 D 填黑之后不连通时处于某个连通分量统一涂黑。若是前者，则放弃将格子 C 涂黑，将其还原成原来未涂黑前的状态，尝试将此格子所对应的{Bn}的下一项尝试涂黑，即继续 2.2)；若是后者，则将在 D 填黑之后不连通时，C 格对应的连通分量还原为原来的状态，作为新的{Bn}，将 D 填黑后原来未填黑的连通分量填黑，返回 2.2)。

3. 输出。

(三)

1. 输入，初始化相关变量。
2. 按照不确定情况下平均分配的原则计算出来 charge[i][j]==false&&opportunity[i][j]>0 的格子，并存放在*temp,记录下 lengthOftemp,进入递归判断函数
3. 递归函数 Recursion
   1) 将*temp 中的方格在循环中变黑，更新 opportunity 与 charge。
   2) 判断是否连通，如果连通，进入下次递归。
   3) 如果不连通，调用 tryMakeBlack 检查是否有可能存在没有数字但可以涂黑的情况，如果存在，将这个连通分量变黑，继续递归，如果仍然不连通，此方格确定，更新数组，继续执行循环。
4. 输出。

符号说明：（以下数组均是二维的）

lineofnum，rowofnum	数字数组的行数与列数
lineofge,rowofge	格子数组的行数与列数
fangge	结构体：方格（存储格子所在的行和列）
int** num	数字数组
fangge** ge	方格数组
bool** charge	判断数组，判断格子的状态是否确定
double** opportunity	概率数组
lengthOftemp	存放递归过程中用的临时变量
chargeCopy	存放递归过程中上一层的** opportunity
opportunityCopy	存放递归过程中上一层的** charge

补充：可以确定（基于当前的方格数组的信息，可得到有格子必定是黑格或白格的结论）的情况：

1. num=0,num=numofge
2. 数字 3 与数字 1 相邻时的情况
3. 数字 3 与数字 3 对角的情况
4. 角部出现数字 1 同时与其相对的格子为黑格的情况

重点函数的作用：

1. **helper**：检查下标是否越界，若没越界回传 1，否则回传 0。
2. **getNumOfGrayge**：计算目前有多少个格子是待定的。（暂时没有被涂成白格或者黑格。）
3. **getNumOfBlack**：计算目前有多少个格子是被涂黑的。
4. **getNumOfBlackOrWhite**：计算目前有多少个格子是被涂黑或者涂白的。
5. **initial**：输入，并初始化相关数据。
6. **checkcontinue**：检查是否只有一个连通分量，若只有一个，则回传 true，

否则回传 false。

7. checknumberblack1：检查所有数字旁边的黑格数是否不超过该数字，若不超过，回传 true，否则回传 false。
8. checknumberblack2：检查所有数字旁边的黑格数是否不等于该数字，若等于，回传 true，否则回传 false。
9. tryMakeBlack：出现不连通时，调用此函数。把能变黑的都变黑连通分量都变黑。具体而言，就是每次先尝试把一个连通分量全部变黑，若不超过数字的限制，则此连通分量确定变黑；若超过连通分量，则把该连通分量还原为其本来的颜色。若整个过程中，有连通分量变黑，则回传 true，否则回传 false。
10. insertionAscB：排序。
11. UpdateOpportunityAndCharge01、UpdateGe、UpdateOpportunityAndCharge：随着不断涂黑方格，更新相关数据。
12. whattoadd：计算出 charge[i][j]==false&&opportunity[i][j]>0 的格子，通过排序，按照 opportunity 从大到小并存放在*temp。
13. printge、printgem：输出函数。
14. Recursion：递归判断函数，将*temp 中的方格在循环中变黑，更新 opportunity 与 charge。判断是否连通，如果连通，进入下次递归。如果不连通，调用 tryMakeBlack 函数检查是否有可能存在没有数字但可以涂黑的情况，如果存在，将这个连通分量变黑，继续递归，如果仍然不连通，此方格确定，更新数组，继续执行循环。

实现难点：

1. 伪代码（一）和（二）的实现过程中，在回溯到上一个格子 A 时，如何判断 A 格涂黑的原因是不违规而涂黑，还是之前某个格子填黑之后不连通时处于某个连通分量统一涂黑？同时，在判断出来之后，如何更新数据以实现程序的正确运作？
2. checknumberblack 和 checknumberblack2 函数中，由于格子数组和数字数组大小不一样，容易越界。

3. checkcontinue 函数中，如何找出一个完整的连通分量：利用队列，通过广度遍历实现。
4. tryMakeBlack 函数中，如何将尝试变黑的连通分量还原：另外设置一个队列，保存原来的相关信息。
5. 实现伪代码（三）时，在执行 recursion 函数中，由于按照概率进行排序尝试，递归的过程不像一般的每次从头开始而是需要不断更新数组，但递归的过程中只能在一次循环中用到，而且每次尝试失败即 return &&chargeRecursion()==false 都需要将数组复原，导致按照这种思路写的递归函数在回溯的过程中出现了意想不到的跳跃，检查不全面，所以有些题目得不出正确答案。

使用的数据结构：

结构体、数组、队列。