1. 首先，我证明了这个棋盘必然是可以被覆盖的

共有$2^k * 2^k = 2^{2k}$个方格，

$\mathrm{k} = 1$时，$2^{2k} - 1 = 3$。

假设 k=n-1 时成立，即$2^{2(n-1)} - 1$是三的倍数。

当 k=n 时，$2^{2n} - 1 = 2^{2(n-1)} * 3 + 2^{2(n-1)} - 1$，因为$2^{2(n-1)} - 1$是三的倍数，

所以$2^{2n} - 1$也是三的倍数

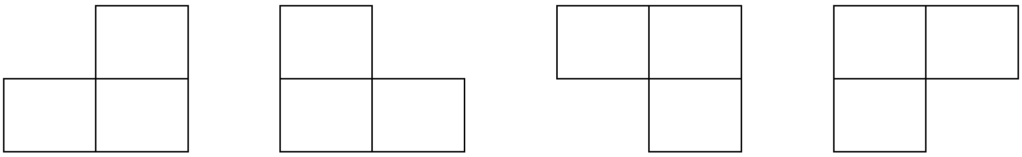
所以$2^k * 2^k$的棋盘必可以被覆盖

2. 我的第一个算法：棋盘问题 1.c

Step1：将棋盘均匀的分割为四部分，特殊方格必然属于其中一部分，递归的不断分割下去，直至分到仅有特殊方格。（find_block()函数）

Step2：递归返回，再返回的同时填充空格。

填充方法：当前的递归层次，必是如下形式之一：

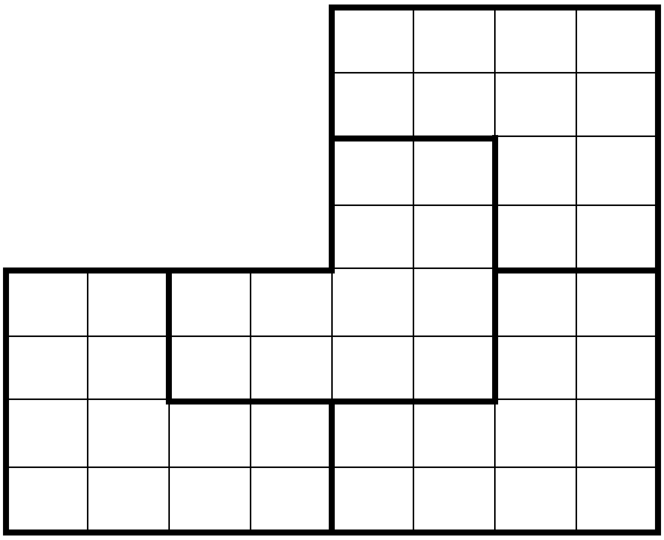


1 型　　2 型　　3 型　　4 型

（其中的一个方格可能是若干阶的方块阵）

其中缺的一块是之前分割棋盘时，特殊方格所在的那一部分，下面以 1 型为例，其他型用相似的方法

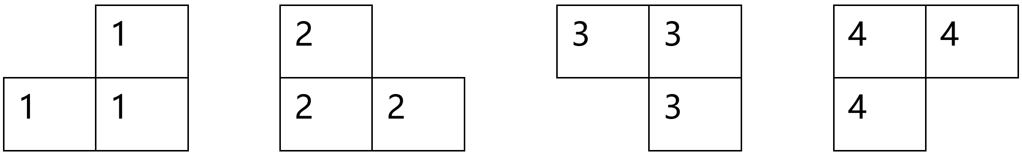
1 型必然可以继续分割，方法如下：



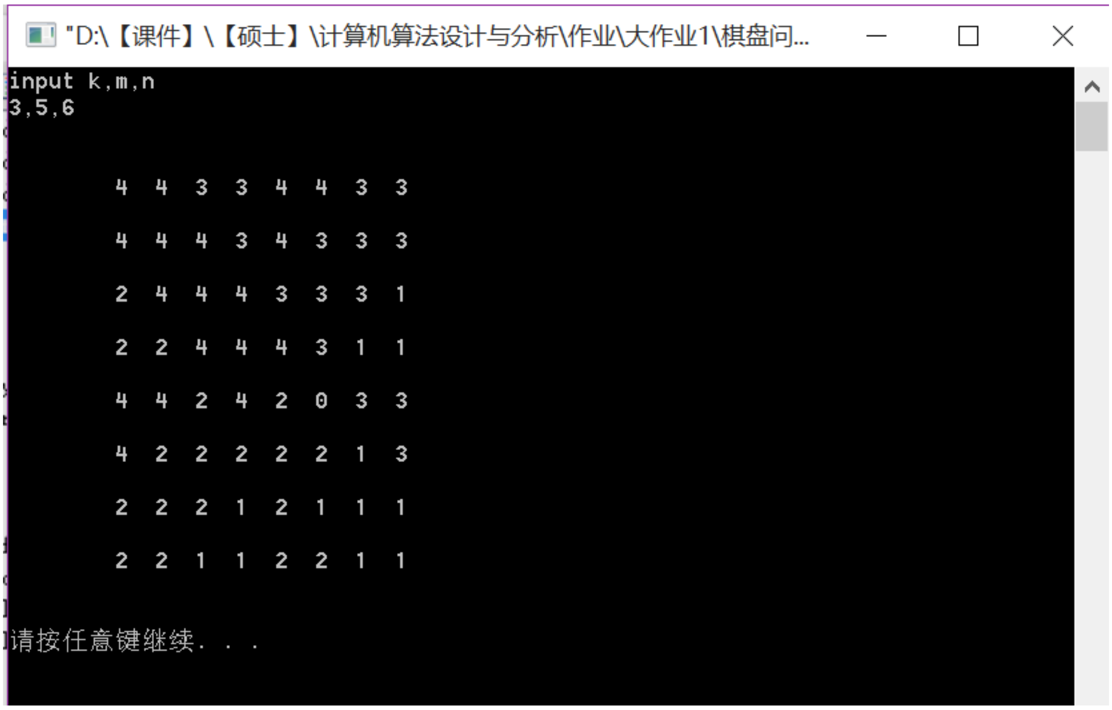
（以 8*8 为例，其他阶的类似）

这四块都是 1 型、2 型、3 型、4 型其中的一种，可按上述方法递归分割。

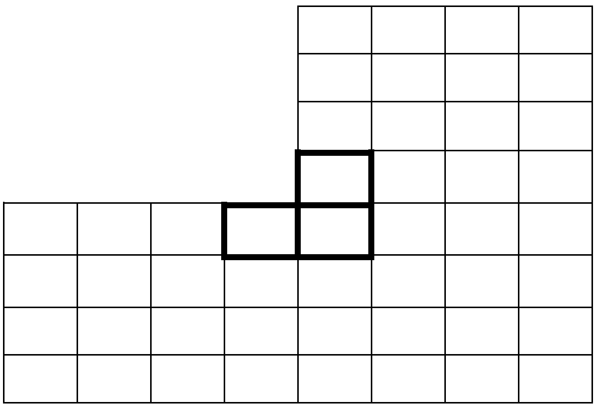
当分割到 2*2 的小方块时，则可以直接填充。(fill_block()函数)
为方便显示，是哪一型的就在方块中填上数字几（如下），而特殊方格填上 0。



实验截图：



3. 我的第二个算法：棋盘问题 2.c
这是我独立做完后，从网上查到的，与老师的提示思路一致。
同上，将棋盘均匀的分割为四部分，特殊方格必然属于其中一部分，则这一部分与原问题一致，复杂度降低，可以递归求解，递归出口是：不断分割下去，直至分到仅有特殊方格。
那分割剩下的三个部分怎么处理呢？



以上图为例，将剩余三部分最靠中间的三块填充（其实任选即可），则这三部分也与原问题一致，递归解决。

实验截图：