例题. RQNOJ 幻方

给定N\*N个数，把它们填入N\*N的方格中，使每行每列和两个斜对角线里数的和都相等

对于30%的测试数据：N<=3

存在20%的测试数据：互不相同的数最多9个

对于100%的测试数据：N<=4，所有出现的数值的绝对值<=10^9

如果有多组解，将他们按照如下方法排序： 如果两组解第一行第一个数一样，那么就去比较第一行的第二个数，否则第一行第一个数较 小的一方更小。如果还是一样，就去比较第一行的第三个数...如果第一行全部一样，就去比 较第二行.....最后比较第N行的第N个数。 在这种排序方式下，输出最小的一组解

Solution1：按常规的搜索顺序，一行一行一列一列枚举+若干剪枝，然后最后一行可以直接算出来

这样思路比较简单，显然也能很容易地求出字典序最小的解，但是N=4过不去

那么怎么办呢？

首先回顾一下WCX写的《Dancing Links》

***Donald Knuth将这一类问题总结为Algorithm X，并提出了一个十分有效的优化方法——Dancing Links。我对其的理解是在尽量靠近递归树根的位置减少问题的规模。***

***我们假设有两种空位总数相同而分布不同的情况：***

***1.空位置数量随着行增加而增加***

***2.空位数量随着行增加而减少***

***看上去此二者复杂度相同，其实不然。前者回溯时避免了后面更多数据的判断，从而加速了算法。为此，我们可以将行按空位置递增排序，并在函数中手动指定搜索顺序。***

然后再看一下USACO上的一个题：Prime3

题目要求计算出一个每行、每列、两个斜对角线上的数都能组成素数的5\*5矩阵。

考虑Dancing Links的思想：**每一次枚举都要尽可能多地为后面的枚举制造约束条件**。

对于一个矩阵来说，那就是两条对角线！它可以控制尽可能多的行和列。

那么我们的枚举顺序就是：

***先枚举两条对角线，因为他们控制的行列是最多的，而且左上角的数已经确定了。接着枚举中间的一竖列,因为它同样控制的行列最多,接着枚举最上面一行和最下面一行。此时第3行的2，4两个数也可以通过减法得到了。最后枚举第2，4行，第3行减法得到。于是所有格子都已经确定了，验证一下就OK。***

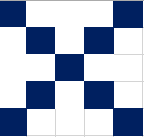
那么对于幻方这道题，我们不妨采用相同的思路：

枚举两条对角线，N<=3时可以直接计算出剩余的空格，判断并输出

N>=4时可以再枚举一些点

拓展：如果N=5呢？

枚举完对角线之后是这种情况：

 然后我们可以枚举第3行和第3列。(同Prime3)

采用这样的搜索方法后，如何保证字典序最小？

因为搜索顺序发生了变化，所以只能**搜出所有的解，再排序**

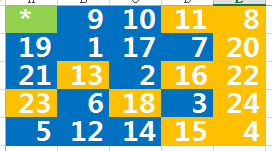
但WCX曰：”**可行解较少，所以速度不会太慢**。”

程序实现方法：

***拓展阅读：NOI专刊 2009.5 P31***

首先我们需要定下搜索顺序，以及那些点需要枚举、那些点可以计算出。

比如Prime3可以设计如下搜索顺序



蓝色是需要枚举的点，橙色是计算的点

绿色为定点，不需操作

我们可以**设f[I,j].x、f[I,j].y，表示[I,j]点之后下一个要枚举的点为[i+f[I,j].x,j+f[I,j].y]**，

并且设cal[I,j]表示[I,j]点是枚举还是计算：

Cal[I,j]的值：

0：[I,j]点需要枚举

1：可以通过横向的一行计算

2：可以通过纵向的一列计算

3：可以通过”\”形状的对角线计算

4：可以通过”/”形状的对角线计算

然后从第一个点开始按顺序dfs，如果当前点需要枚举就枚举，需要计算就计算。

所有的点都确定之后再进行相应的判断（有时甚至已经不用判断了，搜索+计算出来的已经是可行解）

另外在判断的时候可以进行优化：因为搜索过程中有些点是算出来的，所以有一些行\列\对角线已经是可行的了（对于”幻方”这道题），那么我们可以设数组h[i]=true\false，记录第i行是否还需要检查。列\对角线同理

总结：

遇到这种枚举类问题，**应该让先开始的枚举能够尽可能多的约束后面的行为，并且能先计算出的点一定先计算出来。**

比如上面的问题，如果按顺序枚举行和列，可以说，当前枚举的某一行对后面的行丝毫没有约束作用。可是枚举对角线呢？情况就大不相同了。