四皇后问题

解决四皇后问题的方法就是构建一个解空间树，第一个棋子在第一列有4种摆放方法，同样在第二列、第三列、第四列都有4种摆放方法，则共有4×4×4×4=256种方法 ，遍历所有结果就可以找出符合四皇后问题的解。但是这样做会在已经出现错误的解上继续遍历，为了避免出现这样的情况，就采用深度优先搜索的方法，当判断某一种解一定不是正确的解后（出现任意两个或两个以上的棋子在同一行、同一列、对角线位置时），它向下派生出的节点也一定不是正确的解，所以返回上一层，派生下一个节点。但是再解决四皇后问题时，并不需要构建出一个解空间树，可以通过递归算法来模拟。

Completeness：Yes（搜索树空间的宽度、深度是有限的）

当扩展到某一节点，会先判断是否包含该问题的解，如果包含，再继续往下扩展，所以算法一定满足完整性。

Optimally：Yes

在四皇后问题中存在的解都是最优解

Time complexity：对于四皇后问题，搜索树有1+4+16+64+256个节点，每个节点处，要判断此位置的皇后与已经放置的皇后是否互相攻击，最多看3×4个位置，故四皇后问题的该算法在最坏情形下的时间复杂度为O（4^5）

Space complexity:空间复杂度较小

具体例子：

用坐标（X，Y）来表示棋盘上一个格的位置，X表示行，Y表示列

（2，3）

假设在（2，1）位置放棋子

则（2，1）入栈，同一行、同一列、对角线的格上值为零

0

0

0

0

0

0

0

0

0

（2，1）

再从第二列中值非0的格中选择一个放入下一个棋子，比如在（4，2）中放入下一个棋子，则（4，2）入栈，在（4，2）同一行、同一列、对角线上的格的值置为0

0

0

0

0

0

0

0

0

（4，2）

0

0

（2，1）

0

再从第三列中选择（1，3）放入棋子，则（1，3）入栈，在（3，1）同一行、同一列、对角线上的格的值置为0

0

0

0

0

0

0

（1，3）

0

0

0

（4，2）

（2，1）

0

0

0

再从第四列中选择（3，4）放入棋子，

（3，4）

（1，3）

（4，2）

（2，1）