数据结构第一次上机实验文档

班级2021211302

代码测试文档：李俊达 2021210976

问题描述：

八皇后问题是一个以国际象棋为背景的问题：如何能够在8×8的国际象棋棋盘上放置八个皇后，使得任何一个皇后都无法直接吃掉其他的皇后？为了达到此目的，任两个皇后都不能处于同一条横行、纵行或斜线上。

算法思路：

递归：一个程序直接调用自己或通过其它程序调用自己就称为递归。

回溯法是一种满足一定条件的穷举搜索法：在求解过程中，不断利用可供选择的规则扩展部分解，一旦当前的部分解不成立，就停止扩展，退回到上一个较小的部分解继续试探，直到找到问题所有的解或无解为止。

尝试所有的可能，如果不符合条件就倒退回去原来的状态，重新放置皇后，统计解的数量，输出所有符合的结果

算法描述

把棋盘抽象成一个8\*8的数组，初始状态数组值均为0，使某一项赋值为1即为放置皇后。

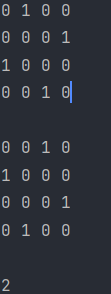
检测如果两个皇后在同一条横行或者纵行或者斜线上就倒退回去，重新更换放置位置。

程序核心：setQueen()

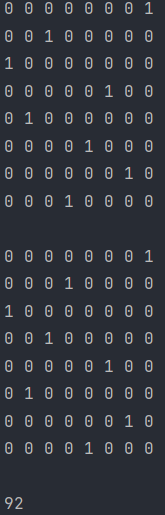
如果 k 等于 N，也就是说成功放置了8个皇后，就会进入条件分支，增加 res 的值，然后输出当前解。如果 k 不等于 N，就会进入 else 分支，程序会遍历当前行的每个位置 (i 从 0 到 N-1) 并尝试放置皇后。 在每个位置 (k, i)，程序调用 check(k, i) 函数来检查是否可以在这个位置放置皇后。如果 check 函数返回 true，表示当前位置是安全的，没有与其他皇后冲突，就会放置皇后在当前位置 boards[k][i] = 1; setQueen(k + 1); 递归调用 setQueen 来处理下一行，即尝试在下一行放置皇后。递归结束后boards[k][i] = 0; 开始回溯操作，重置当前位置，以便在后续的递归中重新使用这个位置，尝试其他可能的放置方式。

测试数据

N=4



N=8



答案过多，附在answer.txt。

结果分析和结论

以4x4为例：

0 0 0 0

0 0 0 0

0 0 0 0

0 0 0 0

（0，0）放置皇后

1 0 0 0

0 0 0 0

0 0 0 0

0 0 0 0

（2，1）放置皇后

1 0 0 0

0 0 1 0

0 0 0 0

0 0 0 0

第三行无法放置皇后，回溯

1 0 0 0

0 0 0 0

0 0 0 0

0 0 0 0

第二行重新放置

1 0 0 0

0 0 0 1

0 0 0 0

0 0 0 0

第三行放置

1 0 0 0

0 0 0 1

0 1 0 0

0 0 0 0

第四行无法放置，回溯到初始状态

0 0 0 0

0 0 0 0

0 0 0 0

0 0 0 0

第一行放置

0 1 0 0

0 0 0 0

0 0 0 0

第二行放置

0 1 0 0

0 0 0 1

0 0 0 0

0 0 0 0

第三行放置

0 1 0 0

0 0 0 1

1 0 0 0

0 0 0 0

第四行放置

0 1 0 0

0 0 0 1

1 0 0 0

0 0 1 0

皇后数量K=4,输出解 res++;

回溯继续重复上述过程略。

程序符合题意，能够输出所有正确结果和结果数

心得体会

通过解决八皇后问题我体会到了递归和回溯的意义。设计setqueen函数遇到很多问题，在调试中逐一解决了，锻炼思维和debug的能力。