项目文档

谈瑞

项目四：N 皇后问题

专业班级：软件工程 4 班 学号：1452775 课目：数据结构课程设计
电话：18936361545 电子邮件：tanrui106@163.com 个人网站：http://guitoubing.top

目录

项目简介

项目概要

八皇后问题是一个古老而著名的问题，是回溯算法的经典问题。该问题是十九世纪著名的数学家高斯在 1850 年提出的：在 8*8 的国际象棋棋盘上，安放 8 个皇后，要求没有一个皇后能够“吃掉”任何其它一个皇后，即任意两个皇后不能处于同一行，同一列或者同一条对角线上，求解有多少种摆法。

高斯认为有 76 种方案。1854 年在柏林的象棋杂志上不同的作者发表了 40 种不同的解，后来有人用图论的方法得到结论，有 92 种摆法。

本实验拓展了 N 皇后问题，即皇后个数由用户输入。

项目功能及要求

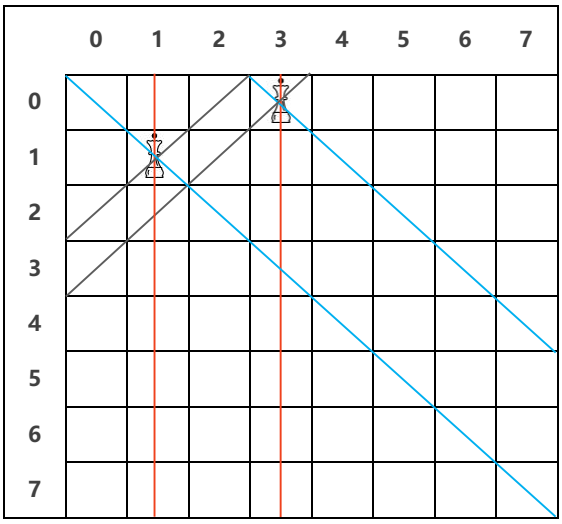
八皇后在棋盘上分布的各种可能的格局数目非常大，约等于 2 的 32 次方种，但是，可以将一些明显不满足问题要求的格局排除掉。由于任意两个皇后不能同行，即每行只能放置一个皇后，因此将第 i 个皇后放在第 i 行上，这样在放置第 i 个皇后时，只要考虑它与前 i-1 个皇后处于不同列和不同对角线位置上即可。

解决这个问题采用回溯法，首先将第一个皇后放置在第一行第一列，然后，依次在下一行上放置一个皇后，直到八个皇后全部放置安全。在放置每个皇后时，都依次对每一列进行检测，首先检测放在第一列是否与已放置的皇后冲突，如不冲突，则将皇后放置在该列，否则，选择改行的下一列进行检测。如整行的八列都冲突，则回到上一行，重新选择位置，依次类推。

项目结构

N 皇后采用回溯法的思想，即试错，对于每一行的一个可能的位置进行错误检测，若符合条件则递归进行下一行的查找，直到找到一种可能结果或者中断某一个不可能结果。

本项目中采用三个数组 valid_left、valid_right、valid_line 来标记不能被使用的位置集合，如下图所示：



每放置一个皇后，其所在的左斜线、右斜线、列三者均被标记为不可用状态，即只能从剩下的未标记位置找皇后位置。易知，其整个棋盘左斜线与右斜线条数均为 2N-1，且对于任意一个点（a,b）来说，其列号为：b、左斜线号为：a+b、右斜线号为：N+a-b-1（行号列号均是从 0 开始），这样在每次确定所查看皇后位置是否为可用时，只需要查看这三个数组对应位置是否被标记即可。而在回溯前又需要将此次位置的三个标记消除。这样避免了确定第 i 个皇后位置时需要对前 i-1 个已经放置好的皇后再进行一次遍历检测，大大节省了运行时间。下面看下 16 皇后问题以下的排列数量及运行时间：

N	1	2	3	4	5	6	7	8
摆法/种	1	0	0	2	10	4	40	92
时间/s	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	0.001
N	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
摆法/种	352	724	2680	14200	73712	365596	2279184	14772512
时间/s	0.001	0.006	0.027	0.14	0.79	4.813	31.445	214.363

项目类的实现

Chessboard 类

类成员	作用
Chessboard();	构造函数
~Chessboard();	析构函数
Chessboard(int _amount);	带参数的构造函数，构造指定大小的（amount*amount）棋盘
void findQueen(int row);	在第row行寻找皇后的递归函数
void printQueen();	输出皇后
inline int getCount();	获取所有摆法的数量
inline double getTime();	获取某一棋盘获取摆法结束花费的时间
vector<Queen*> queen_vec;	存放符合条件的皇后的数组
~~int *present_queen;~~	原用于存放符合条件皇后的列号，与 queen_vec 有重复，但在项目中未删除
bool *valid_line, *valid_left, *valid_right;	标记列、左斜线、右斜线可用位置的数组
int queens_amount;	皇后的数量
int count;	所有摆法的数量
bool if_print;	是否输出（输出所有棋盘会大大降低运行时间）
clock_t time_begin, time_end;	计时器，开始时间和结束时间

Queen 类

类成员	作用
Queen();	构造函数
Queen(int _x, int _y);	带参数的构造函数，创建一个横坐标为_x，纵坐标为_y 的 Queen
~Queen();	析构函数
int index_x, index_y;	存放皇后的横纵坐标

主要代码分析

寻找皇后

```
void Chessboard::findQueen(int row){
    //寻找皇后的函数是一个利用编译器自身回溯机制的递归函数，编译器在某次递归结束后会自动返回上一次递归位置，并将结果带回，这种特性与 N 皇后问题恰好相符合。
    //回溯过程中，如何确定某一位置的皇后是否符合条件是 N 皇后问题所要解决的最主要的方面，这里采用三个标记数组，对每次添加的皇后更新相应位置的标记，使得下一次选择皇后的位置更具体，降低了暴力搜索的花费。
    for (int i = 0; i < queens_amount; i++){
        if (valid_left[row + i] || valid_right[queens_amount+row-i-1] || valid_line[i]){
            //如果该位置已经被前 i-1 个皇后标记过，则直接跳到下一个位置查找
            continue;
        }else{
            //否则将其加入可用皇后数组中
            Queen* temp = new Queen(row+1, i+1);
            queen_vec.push_back(temp);
            valid_left[row+i] = valid_right[queens_amount+row-i-1] = valid_line[i] = true;
            //将此时的列、左斜线、右斜线三个位置添加标记
            if (row == queens_amount-1){
                //如果已经找到最后一行，则说明找到了一种摆法，摆法数量加 1，并输出之（需要的话）
                count++;
                if (if_print)   printQueen();
            }
            findQueen(row+1);
            //递归对下一行开始查找
            valid_left[row+i] = valid_right[queens_amount+row-i-1] = valid_line[i] = false;
            //回溯时将三个标记取消
            queen_vec.pop_back();
            //将该皇后弹出，以查找下一个皇后
            free(temp);
        }
    }
}
```

运行测试

正常运行

```
C:\Users\Administrator\Documents\homework\DataStructure\4_1452775_tanrui>4_1452775_tanrui.exe
现有N*N的棋盘，放入N个皇后，要求所有皇后不在同一行、列和同一斜线上!

请输入皇后的个数：1
是否将所有结果打印出来？（打印结果将会大大降低程序运行速度）y or n: n
1皇后的摆法共有：1种!

这次寻找共用了0s!

C:\Users\Administrator\Documents\homework\DataStructure\4_1452775_tanrui>4_1452775_tanrui.exe
现有N*N的棋盘，放入N个皇后，要求所有皇后不在同一行、列和同一斜线上!

请输入皇后的个数：2
是否将所有结果打印出来？（打印结果将会大大降低程序运行速度）y or n: n
2皇后的摆法共有：0种!

这次寻找共用了0s!

C:\Users\Administrator\Documents\homework\DataStructure\4_1452775_tanrui>4_1452775_tanrui.exe
现有N*N的棋盘，放入N个皇后，要求所有皇后不在同一行、列和同一斜线上!

请输入皇后的个数：3
是否将所有结果打印出来？（打印结果将会大大降低程序运行速度）y or n: n
3皇后的摆法共有：0种!

这次寻找共用了0s!

C:\Users\Administrator\Documents\homework\DataStructure\4_1452775_tanrui>4_1452775_tanrui.exe
现有N*N的棋盘，放入N个皇后，要求所有皇后不在同一行、列和同一斜线上!

请输入皇后的个数：4
是否将所有结果打印出来？（打印结果将会大大降低程序运行速度）y or n: n\
4皇后的摆法共有：2种!

这次寻找共用了0s!

C:\Users\Administrator\Documents\homework\DataStructure\4_1452775_tanrui>4_1452775_tanrui.exe
现有N*N的棋盘，放入N个皇后，要求所有皇后不在同一行、列和同一斜线上!

请输入皇后的个数：5
是否将所有结果打印出来？（打印结果将会大大降低程序运行速度）y or n: n
5皇后的摆法共有：10种!

这次寻找共用了0s!

C:\Users\Administrator\Documents\homework\DataStructure\4_1452775_tanrui>4_1452775_tanrui.exe
现有N*N的棋盘，放入N个皇后，要求所有皇后不在同一行、列和同一斜线上!

请输入皇后的个数：6
是否将所有结果打印出来？（打印结果将会大大降低程序运行速度）y or n: n
6皇后的摆法共有：4种!

这次寻找共用了0s!

C:\Users\Administrator\Documents\homework\DataStructure\4_1452775_tanrui>4_1452775_tanrui.exe
现有N*N的棋盘，放入N个皇后，要求所有皇后不在同一行、列和同一斜线上!

请输入皇后的个数：7
是否将所有结果打印出来？（打印结果将会大大降低程序运行速度）y or n: n
7皇后的摆法共有：40种!

这次寻找共用了0s!

C:\Users\Administrator\Documents\homework\DataStructure\4_1452775_tanrui>4_1452775_tanrui.exe
现有N*N的棋盘，放入N个皇后，要求所有皇后不在同一行、列和同一斜线上!

请输入皇后的个数：8
是否将所有结果打印出来？（打印结果将会大大降低程序运行速度）y or n: n
8皇后的摆法共有：92种!

这次寻找共用了0.001s!

C:\Users\Administrator\Documents\homework\DataStructure\4_1452775_tanrui>4_1452775_tanrui.exe
现有N*N的棋盘，放入N个皇后，要求所有皇后不在同一行、列和同一斜线上!

请输入皇后的个数：9
是否将所有结果打印出来？（打印结果将会大大降低程序运行速度）y or n: n
9皇后的摆法共有：352种!

这次寻找共用了0.001s!
```

运行测试

```
C:\Users\Administrator\Documents\homework\DataStructure\4_1452775_tanrui>4_1452775_tanrui.exe
现有N*N的棋盘，放入N个皇后，要求所有皇后不在同一行、列和同一斜线上!

请输入皇后的个数：10
是否将所有结果打印出来？（打印结果将会大大降低程序运行速度）y or n: n
10皇后的摆法共有：724种!

这次寻找共用了0.006s!

C:\Users\Administrator\Documents\homework\DataStructure\4_1452775_tanrui>4_1452775_tanrui.exe
现有N*N的棋盘，放入N个皇后，要求所有皇后不在同一行、列和同一斜线上!

请输入皇后的个数：11
是否将所有结果打印出来？（打印结果将会大大降低程序运行速度）y or n: n
11皇后的摆法共有：2680种!

这次寻找共用了0.027s!

C:\Users\Administrator\Documents\homework\DataStructure\4_1452775_tanrui>4_1452775_tanrui.exe
现有N*N的棋盘，放入N个皇后，要求所有皇后不在同一行、列和同一斜线上!

请输入皇后的个数：12
是否将所有结果打印出来？（打印结果将会大大降低程序运行速度）y or n: n
12皇后的摆法共有：14200种!

这次寻找共用了0.14s!

C:\Users\Administrator\Documents\homework\DataStructure\4_1452775_tanrui>4_1452775_tanrui.exe
现有N*N的棋盘，放入N个皇后，要求所有皇后不在同一行、列和同一斜线上!

请输入皇后的个数：13
是否将所有结果打印出来？（打印结果将会大大降低程序运行速度）y or n: n
13皇后的摆法共有：73712种!

这次寻找共用了0.79s!

C:\Users\Administrator\Documents\homework\DataStructure\4_1452775_tanrui>4_1452775_tanrui.exe
现有N*N的棋盘，放入N个皇后，要求所有皇后不在同一行、列和同一斜线上!

请输入皇后的个数：14
是否将所有结果打印出来？（打印结果将会大大降低程序运行速度）y or n: n
14皇后的摆法共有：365596种!

这次寻找共用了4.813s!

C:\Users\Administrator\Documents\homework\DataStructure\4_1452775_tanrui>4_1452775_tanrui.exe
现有N*N的棋盘，放入N个皇后，要求所有皇后不在同一行、列和同一斜线上!

请输入皇后的个数：15
是否将所有结果打印出来？（打印结果将会大大降低程序运行速度）y or n: n
15皇后的摆法共有：2279184种!

这次寻找共用了31.445s!

C:\Users\Administrator\Documents\homework\DataStructure\4_1452775_tanrui>4_1452775_tanrui.exe
现有N*N的棋盘，放入N个皇后，要求所有皇后不在同一行、列和同一斜线上!

请输入皇后的个数：16
是否将所有结果打印出来？（打印结果将会大大降低程序运行速度）y or n: n
16皇后的摆法共有：14772512种!

这次寻找共用了214.363s!
```

错误检测

```
C:\Users\Administrator\Documents\homework\DataStructure\4_1452775_tanrui\4_1452775_tanrui.exe
现有N*N的棋盘，放入N个皇后，要求所有皇后不在同一行、列和同一斜线上!

请输入皇后的个数：0
输入错误，请输入一个正整数：-1
输入错误，请输入一个正整数：17
输入错误，请输入小于等于16的正整数：
```