数据结构项目四文档

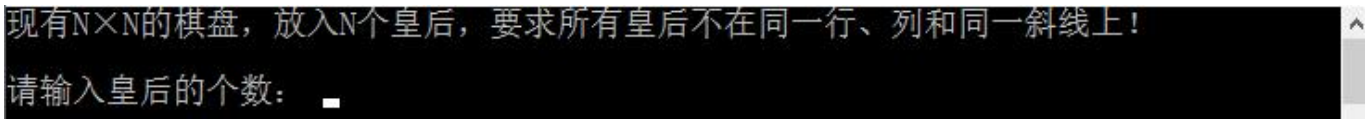
同济大学 软件学院 15级2班 1552651 王依睿

- 使用说明
  - 操作手册
  - 注意事项
- 概述
  - 项目功能要求
  - 程序设计目的
  - 算法思路
  - 数据结构
  - 文件目录
- 源代码

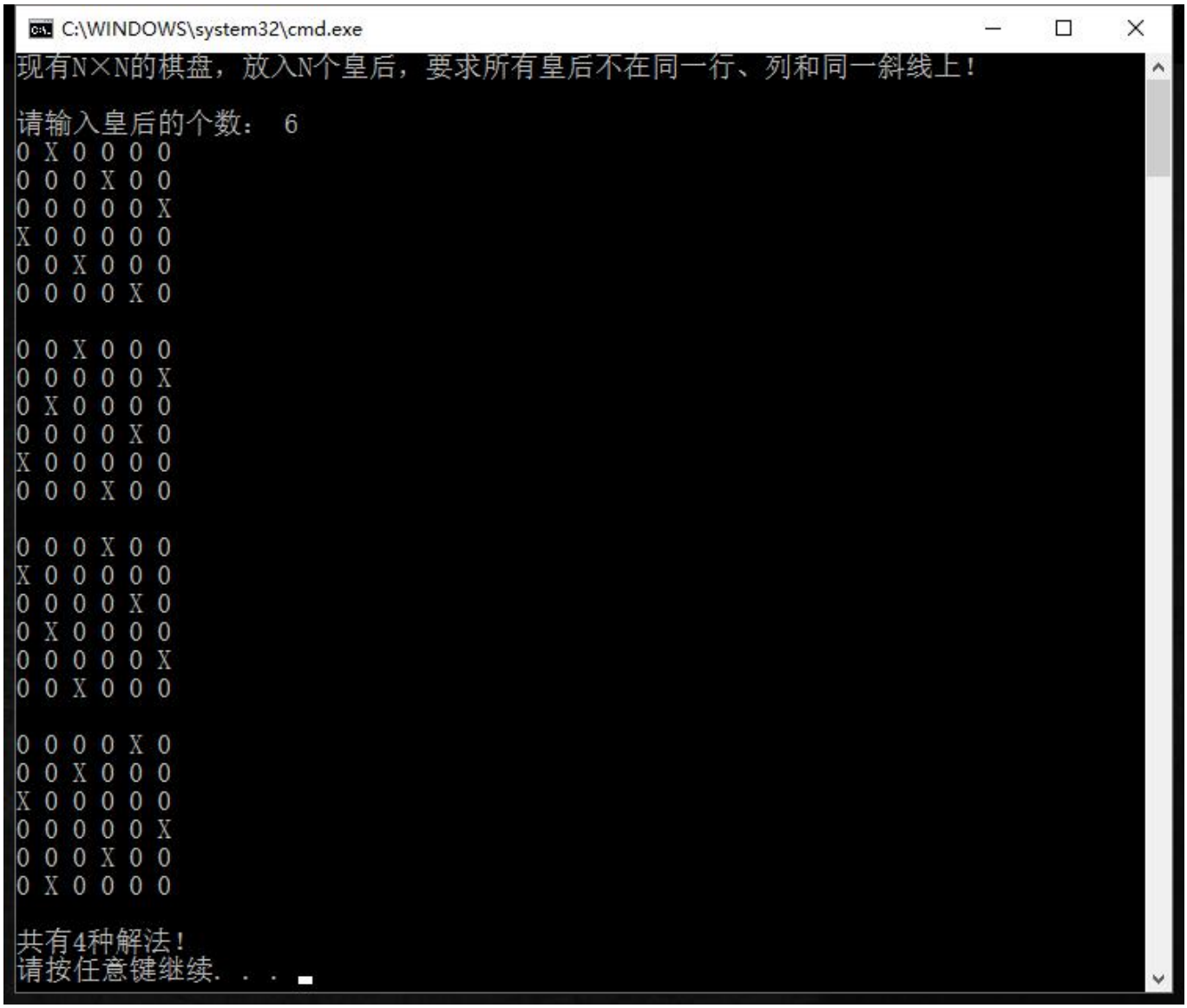
使用说明

操作手册

- 运行程序后，阐明游戏规则，请求用户输入皇后个数。



- 输出所有解法以及解法总数。



注意事项

- 当输入的皇后个数过大时，程序运行时间较长。
- 当输入的皇后个数达到一定大小时，由于深度优先搜索的函数递归调用会导致函数栈溢出。

概述

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中摆法。
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，都依次兑每一列进行检测，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。

程序设计目的

练习掌握DFS及回溯算法，编写一个能输出将N个皇后放在N*N的棋盘上并两两处于不同列和不同对角线位置上的程序。

算法思路

- 利用深度优先搜索算法。
- 利用回溯遍历所有解法。
- 分别利用两个一维数组记录已放置的每行皇后的位置以及每列是否已存在皇后。
- 用横向距离是否等于纵向距离判断是否处于对角线上。

数据结构

- 一维数组。
- 二维数组。

文件目录

- 4_1552651_wangyirui.cpp（主文件）
- 4_1552651_wangyirui.exe（可执行文件）
- 4_1552651_wangyirui.pdf（项目文档）

源代码

```
#include<iostream>
```

```
using namespace std;

char chessboard[50][50];                                                    //棋盘
int position[50];                                                           //记录每行皇后的位置
bool column_available[50];                                                  //记录此列是否已存在
皇后

int n;
int cnt;

bool diagonal_availabe(int ln, int col) {                                   //从对角线角度看是否
能在此位置放置皇后

    for (int i = 0; i < ln; ++i)
        if (ln - i == col - position[i] || ln - i == position[i] - col)
            return false;

    return true;
}

void DFS(int ln) {

    if (ln == n) {                                                          //全部放完后打印棋盘

        ++cnt;
        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < n; ++j)
                cout << chessboard[i][j] << ' ';
            cout << endl;
        }
        cout << endl;
        return;
    }

    for (int i = 0; i < n; ++i) {

        if (column_available[i] && diagonal_availabe(ln, i)) {              //列和对角线都符合要
求
            column_available[i] = false;
            chessboard[ln][i] = 'X';
            position[ln] = i;
            DFS(ln + 1);                                                    //递归
            column_available[i] = true;                                     //回溯
            chessboard[ln][i] = '0';
        }
    }
    return;
}
```

```
int main() {

    cout << "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！\n\n"
        << "请输入皇后的个数：  " << flush;
    cin >> n;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j)
            chessboard[i][j] = '0';

        column_available[i] = true;
    }
    DFS(0);
    cout << "共有" << cnt << "种解法！" << endl;

    return 0;
}
```