N皇后问题

#### 软件工程20级2班 2053300 胡锦晖

# 一、项目概述

## 1.1 项目背景

首先来介绍著名的8皇后问题：在8\*8的国际象棋棋盘上，放置8个皇后，按照国际象棋的规则，皇后的走法是：可以横直斜走，格数不限。因此要求皇后彼此之间不能相互攻击，等价于要求任何两个皇后都不能在同一行、同一列以及同一条斜线上，求解有多少种放置方法使得8个皇后相互不会攻击。

八皇后问题最早是由国际象棋棋手马克斯·贝瑟尔于1848年提出。第一个解在1850年由弗朗兹·诺克给出。并且将其推广为更一般的N皇后问题。N皇后问题中棋盘大小变为n\*n，此时皇后个数也为n个，求解有多少种放置方法使得n个皇后不会相互攻击。

## 1.2 项目目标

要求通过输入的N值求解N皇后问题，若有解则需输出解法个数以及每种解法的具体放置方法，若无解则需给出提示。

# 二、实现思路

## 2.1 数据的存储结构

本题的数据存储较为简单，虽然是一个N\*N的方形棋盘，但由于皇后放置规则的限制，我们可以用一个一维数组来记录每个皇后的放置信息。选择行优先排列，由于每一行只有一个皇后，数组中就存这个皇后所在的列号。

此外，还需要用两个int型数据记录输入的皇后个数以及解法个数。

如下图所示：

## 文本 描述已自动生成2.2 功能实现

### 2.2.1 基本思路

可以考虑回溯的方法：即依次在每一行放置一个皇后。每次新放置的皇后都不能和已经放置的皇后之间有攻击：即新放置的皇后不能和任何一个已经放置的皇后在同一列以及同一条斜线上，并更新数组中的当前行的皇后列下标。当 N 个皇后都放置完毕，则找到一个可能的解，进行记录并输出。

### 2.2.2 解法输出

解法输出的过程即是将存储皇后信息的一维数组重新转换为二维棋盘的过程，借助两层循环的嵌套即可轻易实现。需要说明的是，输出结果中0代表此处没有放置棋子，1代表此处应当放置一个皇后。

### 2.2.3 错误处理

本项目只有一个输入N，按照题目描述，合法的输入范围应是[1, INT\_MAX]范围内的整数。但由于算法时间复杂度较高，输入的N值过大会使得程序的执行时间过长，所以这里人为将输入范围限制在了[1, 15]范围内的整数，其余包括使cin状态出错的输入均判定为不合法的输入。本项目中采用的错误处理方式与Problem2约瑟夫生者死者游戏中相同：

电脑萤幕

描述已自动生成即将用input函数处理待输入数据，当且仅当cin状态正确且data处于由low、high限定的合法范围内时才结束一次输入。

## 2.3 核心算法及流程图

## 2.4 核心代码

文本

描述已自动生成放置皇后：

文本

描述已自动生成判断位置是否允许放置皇后：

# 三、性能分析

## 3.1 时间复杂度

由于每个皇后必须位于不同列，因此已经放置的皇后所在的列不能放置别的皇后。第一个皇后有N列可以选择，第二个皇后最多有N−1列可以选择，第三个皇后最多有N−2列可以选择（如果考虑到不能在同一条斜线上，可能的选择数量更少），因此所有可能的情况不会超过N! 种，遍历这些情况的时间复杂度是O(N!)，其中N为皇后数量

## 3.2 空间复杂度

使用了存储及皇后信息的一维数组，空间复杂度为O(N)，其中N为皇后数量。

## 3.3 运行时间

为了更加直观的理解O(N!)时间复杂度下的问题规模增长速度，程序在开始和结束时分别进行了一次时间记录，二者的差值即为整个程序的运行时间。

电脑的屏幕

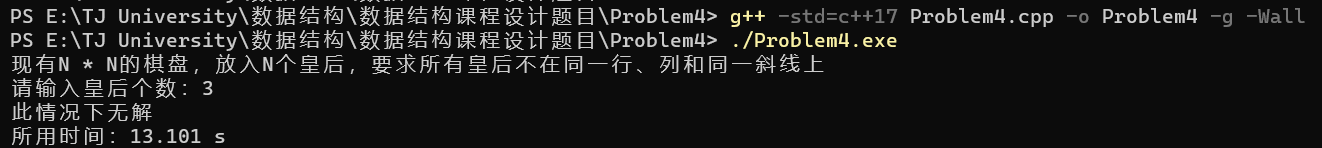
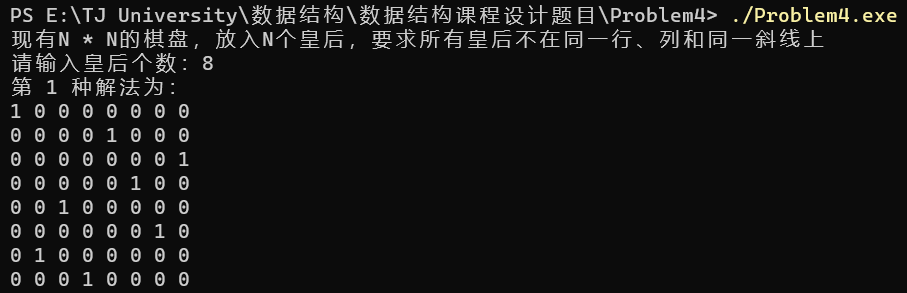
中度可信度描述已自动生成图为13皇后（左）和14皇后（右）的运行时间：

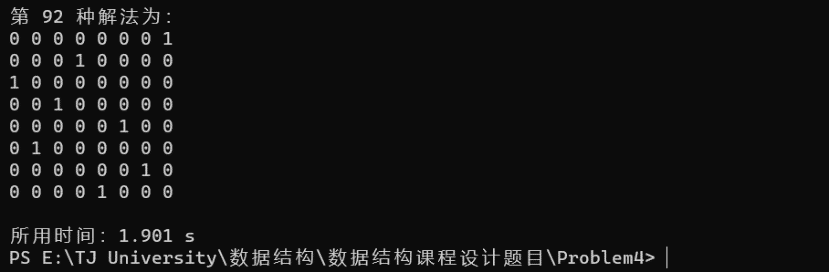
# 四、测试结果

## 4.1 Windows平台

操作系统：Windows11 专业版 21H2

编译器：g++ (GCC) 8.1.0

C++标准：c++17

···

## 4.2 Linux平台

操作系统：CentOS Linux release 8.5.2111

编译器：g++ (GCC) 8.5.0-4

C++标准：c++17

## 4.3 Mac平台

操作系统：macOS Monterey 12.0.1

编译器：clang++ 13.0.0

C++标准：std=c++17

图片包含 形状

描述已自动生成文本

描述已自动生成 ……