

**人 工 智 能 技 术 实 验**

**实 验 报 告**

**实验名称：八皇后问题求解**

**姓 名：李岚琦**

**学 号：2111078**

**专 业：智能科学与技术**

**人工智能学院**

**2023年9月**

1. **问题简述**

八皇后问题，是一个古老而著名的问题：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。

最早是由国际西洋棋棋手马克斯·贝瑟尔于1848年提出，之后陆续有数学家对其进行研究，其中包括高斯和康托。

八皇后问题可以推广为更一般的N皇后摆放问题：这时棋盘的大小变为N×N，而皇后个数也变成N。

1. **实验目的**
2. 通过求解皇后问题，熟悉深度优先搜索法技术；
3. 理解递归回溯算法思想，进而推广到n皇后问题；（选做）
4. 对实验进行图形化界面设计，实现按步或按解的展示。（选做）
5. **实验内容**
   1. 实现八皇后问题的解法统计；
   2. 将八皇后问题推广到N皇后；（选做）
   3. 图形化界面设计。（选做）
6. **编译环境**

**使用了3.7.16版本的python语言**

**使用的工具包是1.10.1版本的pytorch(为了进行cpu与cuda的对比)**

1. **实验步骤**

**代码文件是：backtrack.py**

* 1. **定义全局变量：**

import torch as nn  
##用回溯法进行查找  
  
to\_num = 0  
board\_size = 8  
bs = board\_size  
board = nn.zeros(bs,bs)  
board[0,0]=1

**其中，to\_num是对total\_number的缩写，用来对所有的解法进行计数。**

**Board\_size（即bs）是棋盘的大小，此时先设置为8**

**然后将board设置成bs \*bs大小的一个张量（此时是8x8），并且将[0,0]位置设为1，表示在这里先放一个皇后。**

**另外，在本棋盘中，0代表没有皇后，1代表这个位置放了一个皇后。**

* 1. **定义关键函数**

**我主要定义了三个函数，分别是：**

**Singel\_detect: 检测某个位置的皇后是不是“米”字形里唯一且位于中心：**

def single\_detect(x: nn.Tensor, i, j):  
 row\_sum = x[i,:].sum()  
 col\_sum = x[:,j].sum()  
 dia\_sum = x.flip(0).diagonal(offset = i+j -bs+1).sum()  
 idia\_sum = x.diagonal(offset = j-i).sum()  
 num = row\_sum + col\_sum + dia\_sum + idia\_sum  
 if x[i,j]==1 and num == 4:  
 return True  
 else :  
 return False

**思路是：如果是米字型唯一且中心的皇后，则每行每列以及俩对角线元素 的和各个都是1，也就是加起来是4，而且中心位置的值是1。**

**All\_detect: 用来检测整个棋盘上现有的皇后们（小于等于8），是否都满足题意。（小于等于8的原因是，在这里放置皇后是一个一个放的）**

def all\_detect(x: nn.Tensor):  
 for i in range(bs):  
 for j in range(bs):  
 if x[i,j]==1:  
 if not single\_detect(x,i,j):  
 return False  
  
 else:  
 return True

**思路是：如果某个位置是1，则对其进行single检测。Single检测全通过则整体通过，只要有不通过的，就整体不通过。**

**Put :来放置单个皇后。其中参数i表示这是放置的第i+1个皇后（因为从0开始索引）。**

def put(board:nn.Tensor,i):  
 global num  
 for j in range(bs):  
  
 board[i,j]=1  
 if all\_detect(board) and i<bs-1 :  
 put(board,i+1)  
 if all\_detect(board) and i==bs-1:  
 num+=1  
 print('这是第：',num,'种解法')  
 print(board)  
 board[i,j]=0  
 if j==bs-1 and not all\_detect(board):  
 for k in range(bs):  
 if board[k]==1 and k!=0:  
 board[k-1]=1  
 board[k]=0  
 put(board,i)

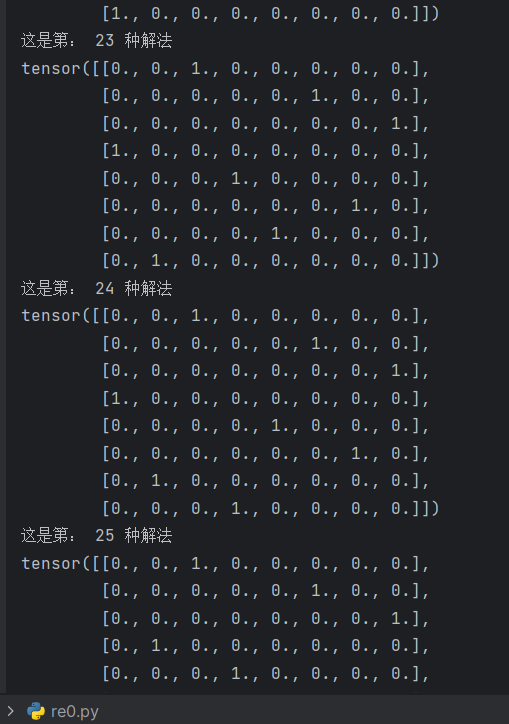
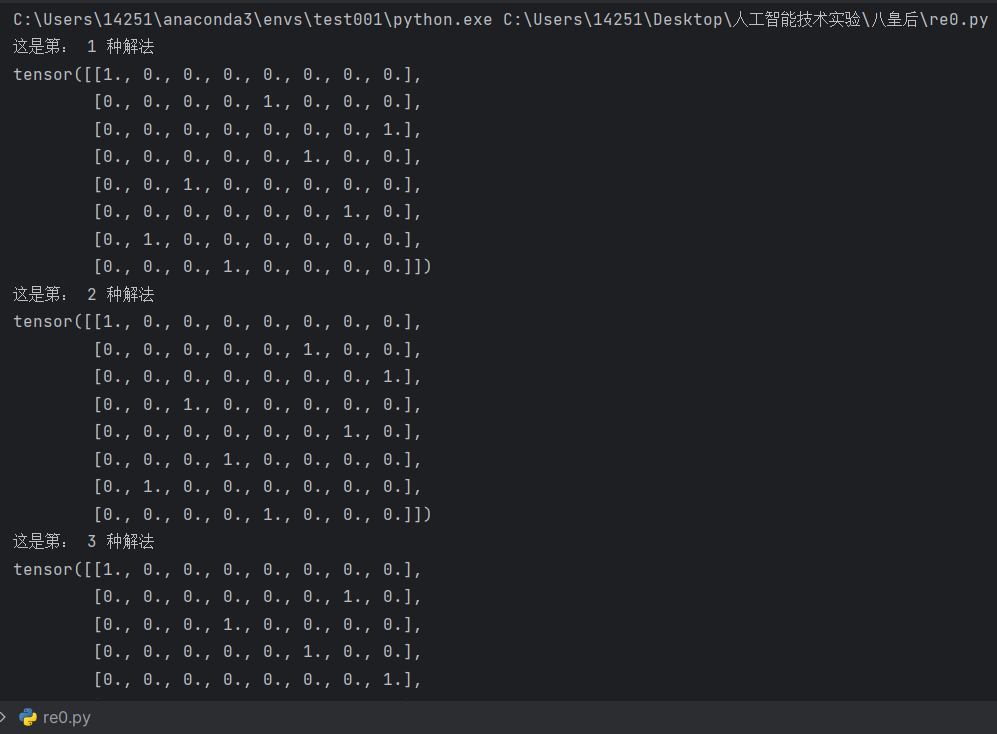
**思路就是回溯法，第i个皇后位置能行，那就put(i+1)，放下一个；如果第i个皇后找遍了也不行，那就让上一个皇后倒退一格，然后重新put(i)**

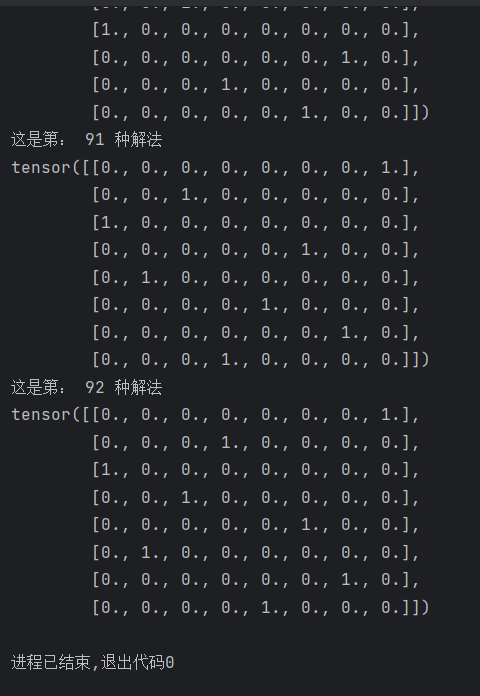
* 1. **运行程序并查看结果**

**调用主体：**

put(board,0)

1. **实验结果**

**查看结果：**

****

**可以观察到，完成了对92种解法的查找，并且以棋盘的形象形式表达了出来。**

1. **分析总结**

**使用递归法大大缩短了程序运行时间，同时代码也更加简洁，而且能够通过调整**

**Board\_size的大小来求解N皇后问题。更具体的分析请见另外一个word文档。**