算法设计与分析第四次上机：八皇后问题

学号：21009200158　姓名：游霄童

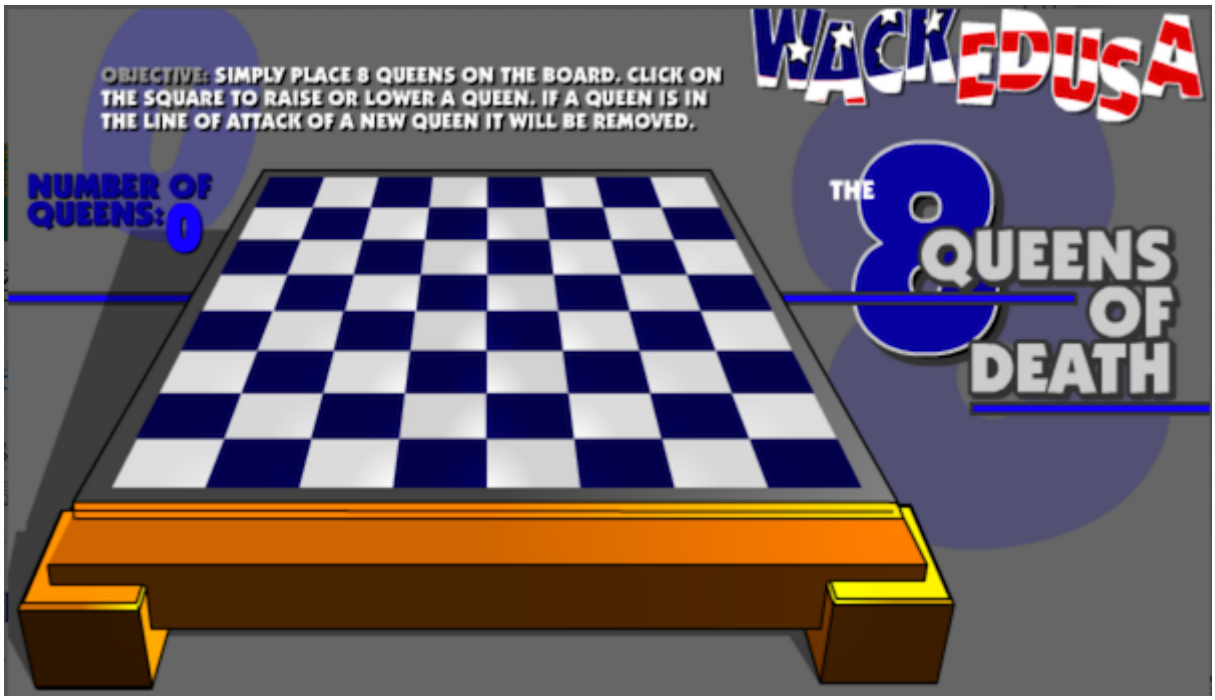
时间：2023/11/21

代码语言为Python

问题描述:

八皇后问题，是一个古老而著名的问题，是 **回溯算法** 的典型案例。

该问题是国际西洋棋棋手马克斯·贝瑟尔于 1848 年提出：在 8×8 格的国际象棋上摆放八个皇后，使其不能互相攻击，即：任意两个皇后都不能处于同一行、同一列或同一斜线上，问有多少种摆法。



使用回溯算法，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解出 **92 种结果**。计算机发明后，有多种计算机语言可以解决此问题

源代码如下:

```
# 导入 List 类型
from typing import List

class Solution:
    def solveNQueens(self, n: int) -> List[List[str]]:
        def generateBoard():
            board = list()
            for i in range(n):
```

```
                row[queens[i]] = "Q"
                board.append("".join(row))
                row[queens[i]] = "."
            return board

        def backtrack(row: int):
            if row == n:
                board = generateBoard()
                solutions.append(board)
            else:
                for i in range(n):
                    if i in columns or row - i in diagonal1 or row + i
in diagonal2:
                        continue
                    queens[row] = i
                    columns.add(i)
                    diagonal1.add(row - i)
                    diagonal2.add(row + i)
                    backtrack(row + 1)
                    columns.remove(i)
                    diagonal1.remove(row - i)
                    diagonal2.remove(row + i)

        solutions = list()
        queens = [-1] * n
        columns = set()
        diagonal1 = set()
        diagonal2 = set()
        row = ["."] * n
        backtrack(0)
        return solutions

# 创建解决器对象
solver = Solution()

# 解决8皇后问题
eight_queens_solutions = solver.solveNQueens(8)

# 打印所有解
i=1
for solution in eight_queens_solutions:
    print(f'第{i}种解法：')
    i = i + 1;
    for row in solution:
        print(row)
    print()
```

结果如下所示：

```
第1种解法：
Q.......
....Q...
.......Q
.....Q..
..Q.....
......Q.
.Q......
...Q....

第2种解法：
Q.......
.....Q..
.......Q
..Q.....
......Q.
...Q....
.Q......
....Q...

第3种解法：
Q.......
......Q.
...Q....
.....Q..
.......Q
.Q......
....Q...
..Q.....

第4种解法：
Q.......
......Q.
....Q...
.......Q
.Q......
...Q....
.....Q..
```

...

...

```
Q.......
......Q.
....Q...
..Q.....
.....Q..

第90种解法：
.......Q
.Q......
....Q...
..Q.....
Q.......
......Q.
...Q....
.....Q..

第91种解法：
.......Q
..Q.....
Q.......
.....Q..
.Q......
....Q...
......Q.
...Q....

第92种解法：
.......Q
...Q....
Q.......
..Q.....
.....Q..
.Q......
......Q.
....Q...


进程已结束，退出代码为 0
```

时间复杂度分析：

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

$$O(N!)$$