Name Aoun Muhammad

Roll no 045

Task Queen Problem

class Solution:

    def solveNQueens(self, n: int) -> list[list[str]]:

        col = set()

        posDiag = set()

        negDiag = set()

        res = []

        board = [["."] \* n for \_ in range(n)]

        def backtrack(r):

            if r == n:

                copy = ["".join(row) for row in board]

                res.append(copy)

                return

            for c in range(n):

                if c in col or (r + c) in posDiag or (r - c) in negDiag:

                    continue

                col.add(c)

                posDiag.add(r + c)

                negDiag.add(r - c)

                board[r][c] = "q"

                backtrack(r + 1)

                col.remove(c)

                posDiag.remove(r + c)

                negDiag.remove(r - c)

                board[r][c] = "0"

        backtrack(0)

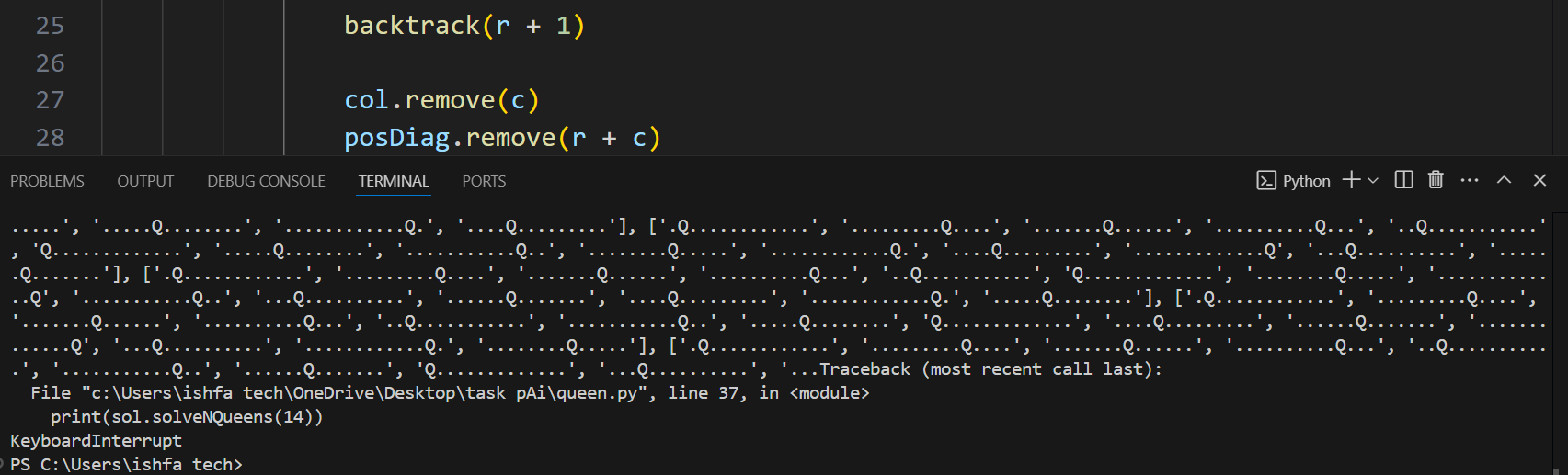
        return res

# Example usage

sol = Solution()

print(sol.solveNQueens(14))

Output



This Python code defines a solution for the N-Queens problem using back off. The solveNQueens function finds all possible ways to place n queens on an n × n chessboard so that no two crowned head attack each other.

It uses sets to track employed columns, positive diagonals, and bad diagonals. The backtrack function iterates row by row, ensuring valid queen placements.

If a full board shape is valid, it is added to the result list. The function prints solutions for n = 14. The board uses "q" for queens and "0" for empty spaces instead of dots