import sys

sys.setrecursionlimit(2000)

# --- Global Configuration ---

global N

N = 8

global INITIAL\_ROW

global INITIAL\_COL

INITIAL\_ROW = 1

INITIAL\_COL = 0

# --- Global Variables for Tracing ---

step\_counter = 0

first\_branch\_trace\_active = True

def print\_trace\_step(board, col\_index, action):

"""Prints a single step of the backtracking trace."""

global step\_counter

step\_counter += 1

print(f"\n--- Step {step\_counter}: Col {col\_index}. {action} ---")

for row in board:

# Use 'Q' for queens and '.' for empty squares for better readability

print(" ".join(['Q' if cell == 1 else '.' for cell in row]))

def print\_final\_board(board):

"""Prints the final solution board cleanly."""

for row in board:

print(" ".join(['Q'if cell == 1 else '.' for cell in row]))

def isSafe(board, row, col):

"""Checks if a queen can be safely placed at board[row][col]."""

# Check row on left side

for i in range(col):

if board[row][i] == 1:

return False

# Check upper diagonal on left side

for i, j in zip(range(row, -1, -1), range(col, -1, -1)):

if board[i][j] == 1:

return False

# Check lower diagonal on left side

for i, j in zip(range(row, N, 1), range(col, -1, -1)):

if board[i][j] == 1:

return False

return True

def solveNQUtil(board, col):

"""

Recursive utility to place queens.

'col' is the current column being processed.

"""

global first\_branch\_trace\_active

# Base case: If all queens are placed, we have a solution.

if col >= N:

return True

# Skip the column where the initial queen is placed

if col == INITIAL\_COL:

if first\_branch\_trace\_active:

print\_trace\_step(board, col, f"SKIP: Initial Queen is fixed at ({INITIAL\_ROW}, {INITIAL\_COL})")

return solveNQUtil(board, col + 1)

# Try placing a queen in all rows of the current column

for i in range(N):

if isSafe(board, i, col):

# 1. PLACE QUEEN (Tentative Decision)

board[i][col] = 1

if first\_branch\_trace\_active:

print\_trace\_step(board, col, f"Place Queen at ({i}, {col})")

# Recur to place the rest of the queens

if solveNQUtil(board, col + 1):

return True

# 2. BACKTRACK (Undo Decision)

# This part is reached only if the recursive call above failed.

board[i][col] = 0

if first\_branch\_trace\_active:

print\_trace\_step(board, col, f"BACKTRACK: Removing Queen at ({i}, {col})")

# CRITICAL: We have backtracked. Stop tracing any further attempts.

first\_branch\_trace\_active = False

# If all rows in this column fail, return False to backtrack further

return False

def solveNQ():

"""Initializes the board and starts the solver."""

global step\_counter, first\_branch\_trace\_active

step\_counter = 0

first\_branch\_trace\_active = True

print(f"--- N-Queens (N={N}) Backtracking Trace of First Branch ---")

print(f"Initial Queen Fixed at: (Row {INITIAL\_ROW}, Col {INITIAL\_COL})")

# Initialize the board with zeros

board = [[0] \* N for \_ in range(N)]

# Place the first queen

if not (0 <= INITIAL\_ROW < N and 0 <= INITIAL\_COL < N):

print("\nError: Initial position is out of bounds.")

return False

board[INITIAL\_ROW][INITIAL\_COL] = 1

# Start the solving process from column 0

if solveNQUtil(board, 0) == False:

print("\n" + "="\*40)

print("--- Final Result ---")

print(f"Solution does not exist for N={N} with initial queen at ({INITIAL\_ROW}, {INITIAL\_COL}).")

print("="\*40)

return False

# If a solution was found, print it cleanly at the end.

print("\n... (omitting further trace steps) ...")

print("\n" + "="\*40)

print("--- Final Result ---")

print("Solution Found!")

print\_final\_board(board)

print("="\*40)

return True

# --- Driver Code ---

solveNQ()

OUTPUT :

PS C:\Users\KJCOEMR\Desktop> python .\n-queens.py

--- N-Queens (N=8) Backtracking Trace of First Branch ---

Initial Queen Fixed at: (Row 1, Col 0)

--- Step 1: Col 0. SKIP: Initial Queen is fixed at (1, 0) ---

. . . . . . . .

Q . . . . . . .

. . . . . . . .

. . . . . . . .

. . . . . . . .

. . . . . . . .

. . . . . . . .

. . . . . . . .

--- Step 2: Col 1. Place Queen at (3, 1) ---

. . . . . . . .

Q . . . . . . .

. . . . . . . .

. Q . . . . . .

. . . . . . . .

. . . . . . . .

. . . . . . . .

. . . . . . . .

--- Step 3: Col 2. Place Queen at (0, 2) ---

. . Q . . . . .

Q . . . . . . .

. . . . . . . .

. Q . . . . . .

. . . . . . . .

. . . . . . . .

. . . . . . . .

. . . . . . . .

--- Step 4: Col 3. Place Queen at (2, 3) ---

. . Q . . . . .

Q . . . . . . .

. . . Q . . . .

. Q . . . . . .

. . . . . . . .

. . . . . . . .

. . . . . . . .

. . . . . . . .

--- Step 5: Col 4. Place Queen at (4, 4) ---

. . Q . . . . .

Q . . . . . . .

. . . Q . . . .

. Q . . . . . .

. . . . Q . . .

. . . . . . . .

. . . . . . . .

. . . . . . . .

--- Step 6: Col 4. BACKTRACK: Removing Queen at (4, 4) ---

. . Q . . . . .

Q . . . . . . .

. . . Q . . . .

. Q . . . . . .

. . . . . . . .

. . . . . . . .

. . . . . . . .

. . . . . . . .

... (omitting further trace steps) ...

========================================

--- Final Result ---

Solution Found!

. . . . . Q . .

Q . . . . . . .

. . . . Q . . .

. Q . . . . . .

. . . . . . . Q

. . Q . . . . .

. . . . . . Q .

. . . Q . . . .

========================================