Program

```
import copy

N = 8  # Size of the chessboard (8x8)
```

```
# Function to print the solution
def printSolution(board):
    for row in board:
        for i in range(N):
            print("Q" if row[i] else ".", end=" ")
        print()
    print()  # Add a newline for readability

# Function to check if a queen can be placed on board[row][col]
def isSafe(board, row, col):
    # Check the column
    for i in range(row):
        if board[i][col]:
            return False

    # Check the upper left diagonal
    for i, j in zip(range(row - 1, -1, -1), range(col - 1, -1, -1)):
        if board[i][j]:
            return False

    # Check the upper right diagonal
    for i, j in zip(range(row - 1, -1, -1), range(col + 1, N)):
        if board[i][j]:
            return False
    return True

# Function to solve the 8 Queens problem using backtracking
def solve(board, row, solutions):
    if row == N:
        solutions.append(copy.deepcopy(board))  # Deep copy of the board
        printSolution(board)
```

```
        return

    for col in range(N):
        if isSafe(board, row, col):
            board[row][col] = 1  # Place queen
            solve(board, row + 1, solutions)  # Recur to place next queen
            board[row][col] = 0  # Backtrack (remove queen)
# Main function to initialize the board and start solving the problem
def eightQueens():
    board = [[0 for _ in range(N)] for _ in range(N)]
    solutions = []  # Store all solutions
    solve(board, 0, solutions)
    print(f"Total solutions found: {len(solutions)}")
# Calling the function
eightQueens()
```

Output:

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

Result:

Thus the given case-based discussion program has been implemented successfully and the program has been uploaded in GitHub link