|  |  |
| --- | --- |
| **EX.N0:1** | **8- QUEENS PROBLEM** |
| **DATE: 28/02/2024** |

**AIM:**

To solve 8Queen Problem using python

**ALGORITHM:**

Step1: Initiate an empty chess board of size8x8.

Step 2: Start with the left most column and place a queen in the first row of the column.

Step3: Move to the next column and place a queen in the first row of the column.

Step4: Repeat Step3 until either all 8queen has been placed or it is impossible to place a

Queen in the current column without violating the rule of the problem.

Step5: If it not possible to place a queen in the current column

**PROGRAM:**

N = 8  # Assuming N is defined elsewhere

def solveNQueens(board, col):

    # Base case: If all columns are filled, print the board and return True

    if col == N:

        printBoard(board)

        return True

    # Recursive case: Try placing queens in each row of the current column

    for i in range(N):

        # Check if it's safe to place a queen in this row and column

        if isSafe(board, i, col):

            # Place a queen in this position

            board[i][col] = 'Q'

            # Recursively solve the problem for the next column

            if solveNQueens(board, col + 1):

                return True

            # Backtrack: If placing a queen doesn't lead to a solution, reset the position

            board[i][col] = 0

    # If no solution is found for this column, return False

    return False

def isSafe(board, row, col):

    # Check if there's a Queen in the same row

    for x in range(col):

        if board[row][x] == 'Q':

            return False

    # Check upper diagonal on left side

    for x, y in zip(range(row, -1, -1), range(col, -1, -1)):

        if board[x][y] == 'Q':

            return False

    # Check lower diagonal on left side

    for x, y in zip(range(row, N, 1), range(col, -1, -1)):

        if board[x][y] == 'Q':

            return False

    # If no conflicting queens are found, return True

    return True

def printBoard(board):

    # Print the board configuration

    for i in range(N):

        for j in range(N):

            print(board[i][j], end=" ")

        print()

# Initialize the board with all cells set to 0

board = [[0 for x in range(N)] for y in range(N)]

# Start solving the N-Queens problem from column 0

if not solveNQueens(board, 0):

    print("No solution found")  # If no solution is found, print a message

**Output**

Q 0 0 0 0 0 0 0

0 0 0 0 0 0 Q 0

0 0 0 0 Q 0 0 0

0 0 0 0 0 0 0 Q

0 Q 0 0 0 0 0 0

0 0 0 Q 0 0 0 0

0 0 0 0 0 Q 0 0

**RESULT:**

Thus, the experiment to **solve the 8-Queen Problem** by using Python has been executed and verified successfully.