Bansilal Ramnath Agarwal Charitable Trust's
Vishwakarma Institute of Information Technology

Department of
Artificial Intelligence and Data Science

Student Name: **Sahil Dilip Sonawane**		
Class: **TY**	Division: **C**	Roll No.: **373057**
Semester: **5**	Academic Year: **2023-24**	
Subject Name & Code: **Design and Analysis of Algorithm: ADUA31202**		
Title of Assignment: **Implement N- queen Problem using Back tracking method. A suitable message is to be displayed if the given Problem instance doesn't have a solution.**		
Date of Performance:	Date of Submission:	

Assignment No. 6

Aim:

To implement N- queen Problem using Back tracking method.

Problem Statement:

Implement N- queen Problem using Back tracking method. A suitable message is to be displayed if the given problem instance does not have a solution.

Theory:

Backtracking is a technique based on algorithm to solve problem. It uses recursive calling to find the solution by building a solution step by step increasing values with time. It removes the solutions that does not give rise to the solution of the problem based on the constraints given to solve the problem.

Backtracking algorithm is applied to some specific types of problems-

- Decision problem used to find a feasible solution of the problem.
- Optimisation problem used to find the best solution that can be applied.
- Enumeration problem used to find the set of all feasible solutions of the problem.

In backtracking problem, the algorithm tries to find a sequence path to the solution which has some small checkpoints from where the problem can backtrack if no feasible solution is found for the problem.

Algorithm:

Step 1 –

Start from 1st position in the array.

Step 2 –

Place queens in the board and check.
Do,
After placing the queen, mark the position as a part of the solution and then recursively check if this will lead to a solution.
Now, if placing the queen does not lead to a solution and trackback and go to step (a) and place queens to other rows.
If placing queen returns a lead to solution return TRUE.

Step 3 –

If all queens are placed return TRUE.

Step 4 –

If all rows are tried and no solution is found, return FALSE.

Complexity Analysis:

Time Complexity:

O(N!), For the first column we will have N choices, then for the next column, we will have N-1 choices, and so on. Therefore the total time taken will be N*(N-1)*(N-2)...., which makes the time complexity to be O(N!).

Space Complexity:

O(N^2), we can see that the initial space needed is N^2, then N^2-1, then N^2-2 and so on. Thus, making the space complexity N^2N, but we don't need all the N^2 options each time. Therefore, the overall space complexity is O(N^2).

Software Requirements:

Text Editor: VSCode, Online GDB Compiler
Environment: Python

Program Code:

```
def is_safe(board, row, col, n):
    # Check for queens in the same row on the left side
    for i in range(col):
        if board[row][i] == 1:
            return False

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

    # Check for queens in the lower diagonal on the 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 solve_nqueens_util(board, col, n):
    if col >= n:
        return True

    for i in range(n):
        if is_safe(board, i, col, n):
            board[i][col] = 1

            if solve_nqueens_util(board, col + 1, n):
                return True

            board[i][col] = 0  # Backtrack if placing a queen at board[i][col] doesn't lead to a solution

    return False
```

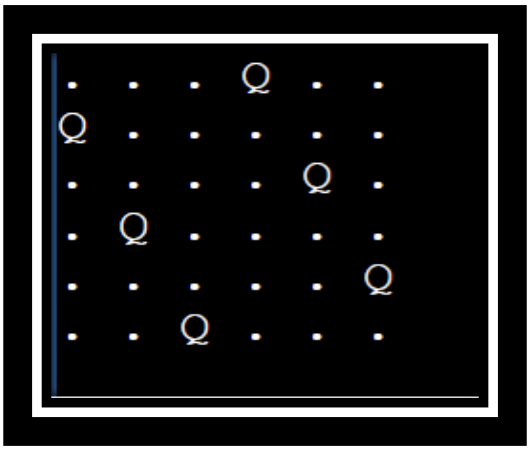
```
def solve_nqueens(n):
    # Initialize the chessboard
    board = [[0 for _ in range(n)] for _ in range(n)]

    if not solve_nqueens_util(board, 0, n):
        print("No solution exists.")
        return

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

n = 6
solve_nqueens(n)
```

Output:



Conclusion:

Implemented N- queen Problem using Back tracking method. A suitable message is also displayed if the given Problem instance doesn't have a solution.
