Exp-6.1

Title:

Visualization of the N-Queens Problem Solutions

Aim:

To understand the placement of queens on an N×N chessboard through graphical representations for different values of N (4, 5, and 8) and to demonstrate how visualization aids in debugging and gaining insights into the problem's complexity.

Algorithm

1. Initialize an empty N×N board with '.' representing empty cells.
2. Start placing queens row by row, beginning from the first row.
3. For the current row, iterate over each column to find a safe position.
4. Check if placing a queen at the current cell does not conflict with previously placed queens in:
    1. The same column
    2. The upper left diagonal
    3. The upper right diagonal
5. If safe, place the queen ('Q') at the current cell and move to the next row recursively.
6. If a conflict occurs or no valid placements are possible in a row, backtrack by removing the queen from the previous row and try different positions.
7. Repeat steps 3-6 until all queens are placed or all possible placements are exhausted.

Input:

N = 4

N = 5

Output:

.Q..

...Q

Q...

..Q.

Q....

..Q..

....Q

.Q...

...Q.

Performance Analysis:

Time Complexity: O(N!)

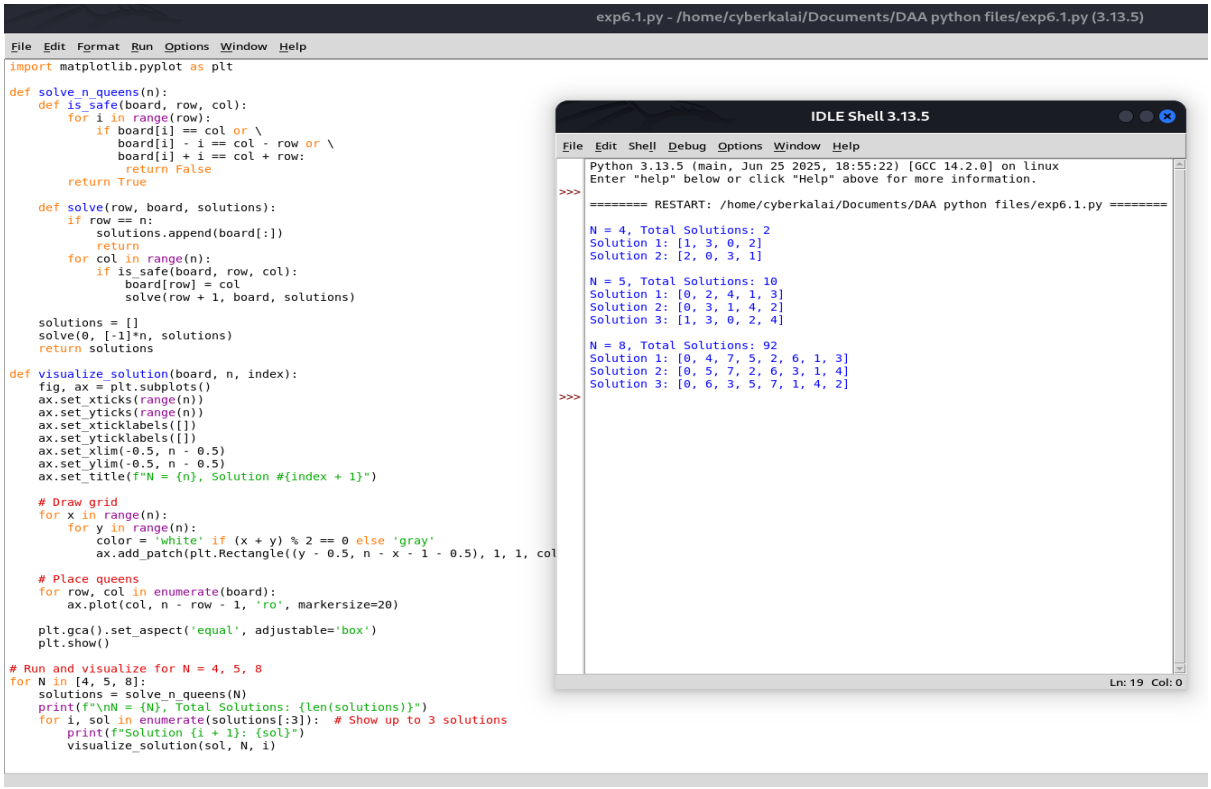
Space Complexity: O($N^{2)}$

Program output :

```
def solveNQueens(N):
    board = [['.' for _ in range(N)] for _ in range(N)]

    def isSafe(row, col):
        # Check column conflicts
        for i in range(row):
            if board[i][col] == 'Q':
                return False
        # Check upper left diagonal
        i, j = row - 1, col - 1
        while i >= 0 and j >= 0:
            if board[i][j] == 'Q':
                return False
            i -= 1
            j -= 1
        # Check upper right diagonal
        i, j = row - 1, col + 1
        while i >= 0 and j < N:
            if board[i][j] == 'Q':
                return False
            i -= 1
            j += 1
        return True

    def backtrack(row):
        if row == N:
```

```
            printBoard(board)
            return True
        for col in range(N):
            if isSafe(row, col):
                board[row][col] = 'Q'
                if backtrack(row + 1):
                    return True
                board[row][col] = '.'
        return False

    backtrack(0)
```

Program Output:



Result:

Thus, the Nqueens program is implemented and got output executed successfully.