

EX.NO:01

DATE:09.08.2024

Reg.no:220701054

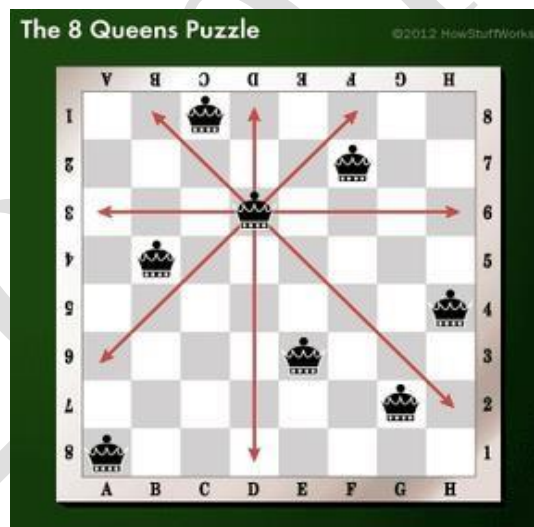
8- QUEENS PROBLEM

AIM:

To implement an 8-Queens problem using Python.

You are given an 8x8 board; find a way to place 8 queens such that no queen can attack any other queen on the chessboard. A queen can only be attacked if it lies on the same row, same column, or the same diagonal as any other queen. Print all the possible configurations.

To solve this problem, we will make use of the Backtracking algorithm. The backtracking algorithm, in general, checks all possible configurations and tests whether the required result is obtained or not. For the given problem, we will explore all possible positions the queens can be relatively placed at. The solution will be correct when the number of placed queens = 8.



Code:

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

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

    return True

def solve_n_queens_util(board, col, n):
    # If all queens are placed, return True
    if col >= n:
        return True

    # Try placing the queen in each row in this column
    for i in range(n):
        if is_safe(board, i, col, n):
            # Place the queen
            board[i][col] = 1

            # Recur to place the rest of the queens
            if solve_n_queens_util(board, col + 1, n):
```

```

        # Backtrack and remove the queen
        board[i][col] = 0

    return False

def print_board(board, n):
    print("\nSolution:")
    for row in board:
        for cell in row:
            if cell == 1:
                print('Q', end=' ')
            else:
                print('.', end=' ')
        print() # New line after each row

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

    if not solve_n_queens_util(board, 0, n):
        print("No solution exists")
        return False

    # Print the solution in a nice format
    print_board(board, n)
    return True

# Driver code to get user input
n = int(input("Enter the value of N: "))
solve_n_queens(n)

```

Output:

The screenshot shows a Jupyter Notebook window titled "220701054.ipynb". The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar with buttons for adding code or text cells and running the current cell. The active cell is a code cell containing the input "Enter the value of N: 8". Below the input, the output of the code is displayed, showing the solution for N=8. The output is a grid of 8 rows and 8 columns, where 'Q' represents a queen and '.' represents an empty cell. The queens are placed at (0,7), (1,3), (2,2), (3,5), (4,1), (5,6), (6,4), and (7,0). The output ends with "True".

```

Enter the value of N: 8

Solution:
Q . . . . . . .
. . . . . Q .
. . . . Q . .
. . . . . . Q
. Q . . . . .
. . . Q . . .
. . . . . Q .
. . Q . . . .
True

```

220701054