ARTIFICIAL INTELLIGENCE (18CSC305J) LAB

EXPERIMENT-1: TOY PROBLEM

Harsh Goel
RA1811003010185
CSE-C1

Aim:

To solve N Queens problem and verify it's test cases.

Problem Description:

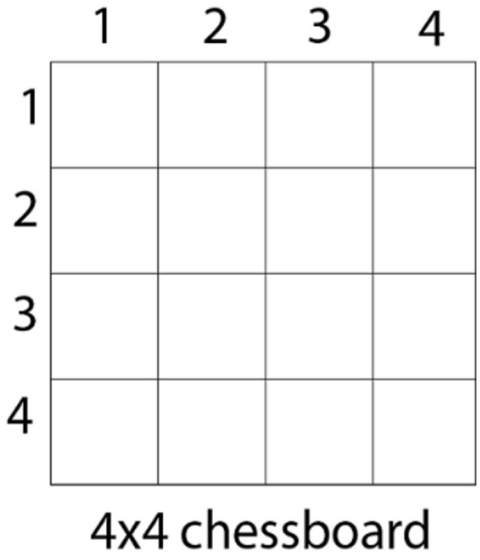
N - Queens problem is to place n - queens in such a manner on an n x n chessboard that no queens attack each other by being in the same row, column or diagonal.

Problem Formulation:

It can be seen that for n =1, the problem has a trivial solution, and no solution exists for n =2 and n =3. So first we will consider the 4 queens problem and then generate it to n - queens problem.

Given a 4 x 4 chessboard and number the rows and column of the chessboard 1 through 4.

Since, we have to place 4 queens such as q1 q2 q3 and q4 on the chessboard, such that no two queens attack each other. In such a conditional each queen must be placed on a different row, i.e., we put queen "i" on row "i."



4x4 chessboard

Now, we place queen q1 in the very first acceptable position (1, 1). Next, we put queen q2 so that both these queens do not attack each other. We find that if we place q2 in column 1 and 2, then the dead end is encountered. Thus the first acceptable position for q2 in column 3, i.e. (2, 3) but then no position is left for placing queen 'q3' safely. So we backtrack one step and place the queen 'q2' in (2, 4), the next best possible solution. Then we obtain the position for placing 'q3' which is (3, 2). But later this position also leads to a dead end, and no place is found where 'q4' can be placed safely. Then we have to backtrack till 'q1' and place it to (1, 2) and then all other queens are placed safely by moving q2 to (2, 4), q3 to (3, 1) and q4 to (4, 3). That is, we get the solution (2, 4, 1, 3). This is one possible solution for the 4-queens problem. For another possible solution, the whole method is repeated for all partial solutions. The other solutions for 4 - queens problems is (3, 1, 4, 2

Algorithm:

STEP 1. Start in the leftmost column

STEP 2. If all queens are placed

return true

STEP 3. Try all rows in the current column.

Do following for every tried row.

a) If the queen can be placed safely in this row then mark this [row, column] as part of the solution and recursively check if placing queen here leads to a solution.

b) If placing the queen in [row, column] leads to a solution then return true.

c) If placing queen doesn't lead to a solution then unmark this [row, column] (Backtrack) and go to step (a) to try other rows.

STEP 4. If all rows have been tried and nothing worked, return false to trigger backtracking.

Source Code:

Language-**PYTHON**

```
global N
N = 7
```

```
def checkSafe(board, row, col):
    # STEP 1 row left side
    for i in range(col):
        if board[row][i] == 1:
            return False
    # upper diagonal left side
    for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    # lower diagonal left side
    for i, j in zip(range(row, N, 1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False
    #default yes
    return True

def solveNQueen(board, col):
    # STEP 1(a) if all queens are placed
    if col >= N:
        return True

    # placing this queen in all rows one by one
    for i in range(N):
        #STEP 2(a) Check if can be placed safely
        if checkSafe(board, i, col):
            board[i][col] = 1
            if solveNQueen(board, col + 1) == True:
                #STEP 2(b) if placed, return true
                return True
            board[i][col] = 0
    #STEP 1(b) default, if queen not placed
    return False

def solveNQueenProblem(board):
    if solveNQueen(board, 0) == False:
        print ("Solution does not exist")
        return False
    # Print the solution
    for i in range(N):
        for j in range(N):
            print (board[i][j], end=" ")
        print("")
    return True
```

```
board = [ [0, 0, 0, 0,0,0],[0, 0, 0, 0,0,0], [0, 0, 0, 0,0,0],[0, 0, 0,
0,0,0], [0, 0, 0, 0,0,0], [0, 0, 0, 0,0,0] ] #initialize the board

solveNQueenProblem(board)
```

TEST CASE:

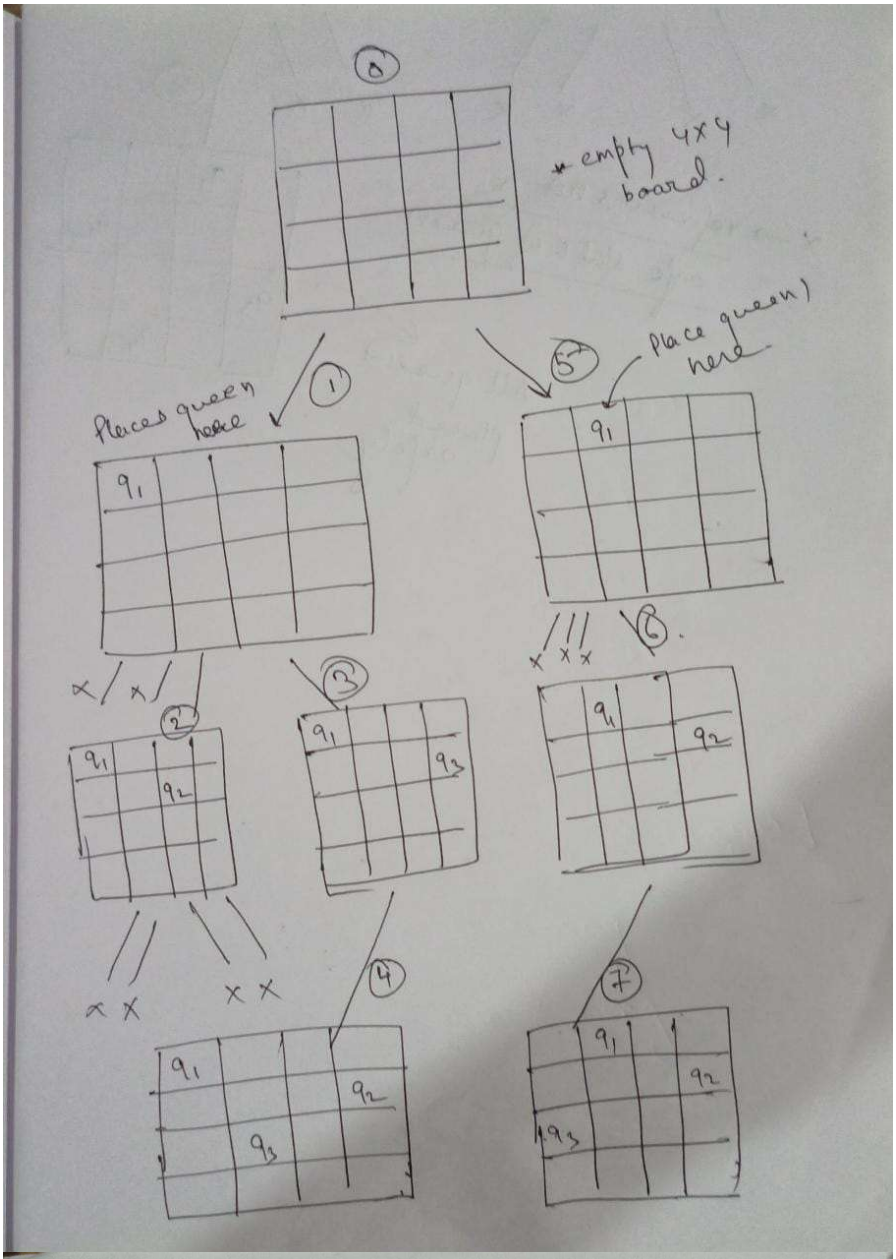
1. **Input :**
    a. **6*6 CHESS BOARD**

```
PS C:\Users\HARSH-PC\Desktop\college\AI\EXP_1> py .\n_queen.py
0 0 0 1 0 0
1 0 0 0 0 0
0 0 0 0 1 0
0 1 0 0 0 0
0 0 0 0 0 1
0 0 1 0 0 0
PS C:\Users\HARSH-PC\Desktop\college\AI\EXP_1>
```

    b. **4*4 CHESS BOARD**

```
PS C:\Users\HARSH-PC\Desktop\college\AI\EXP_1> py .\n_queen.py
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
PS C:\Users\HARSH-PC\Desktop\college\AI\EXP_1>
```

Verification:



We verify the steps according the input for 4*4 board with 4 queens and we verify that on step 8 all queens are placed safely.

IsSafe() function is called each time we place a queen and it returns a Boolean accordingly. Thus we keep placing the queens using backtracking one by one

Result: We have successfully solved the N queen problem using Python and verified the output and test cases.