Amit Srivastav

RA1911003010633

Artificial Intelligence Lab

LAB-1(A)

Aim: Implementation of Toy Problem - 8 Queen's Problem

Initial State

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	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

Final State

1	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0
0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0

(1-denotes that the Queen is placed in the grid, 0 - denotes no Queen is placed in that grid square)

Problem Formulation:

To place 8 queens in such a manner on a 8x8 chessboard such that no queens attack each other by being in the same row, column or diagonal.

Display one of the possible configurations.

Problem Solving:

To solve this problem backtracking algorithm is applied.

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	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

→

1	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	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

→

1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	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	0	0
0	0	0	0	0	0	0	0

↓

1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	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

←

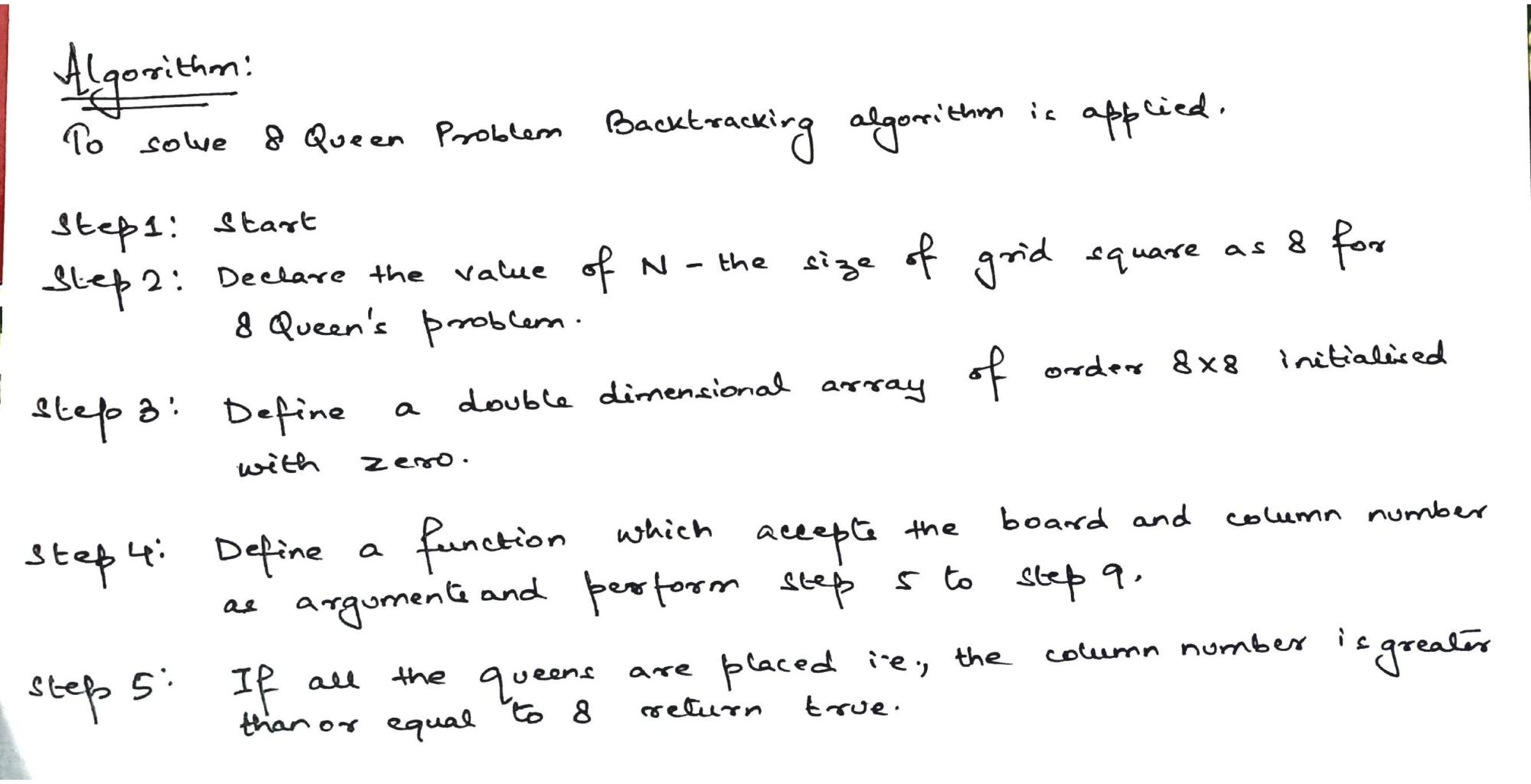
1	0	0	0	0	0	0	0
0	0	0	~~1~~	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

←

1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	1	0	0	0
0	0	1	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

And it tries for rest of the queens and place them safely using backtracking technique.

Scanned by TapScanner



Algorithm:

To solve 8 Queen Problem Backtracking algorithm is applied.

Step 1: Start

Step 2: Declare the value of N - the size of grid square as 8 for 8 Queen's problem.

Step 3: Define a double dimensional array of order 8x8 initialised with zero.

Step 4: Define a function which accepts the board and column number as arguments and perform step 5 to step 9.

Step 5: If all the queens are placed i.e., the column number is greater than or equal to 8 return true.

Step 6: Set $i=0$ and repeat step 7 to 8 till $i<N$.

Step 7: If the queen can be placed safely in that grid then mark it with 1 and go to step 4 with the column number incremented by 1.

Step 8: If the queen placed cannot lead to a solution then unmark the grid with 0 and goto step 6 (Backtrack) with i incremented by 1.

Step 9: If all grids have been tried and nothing worked, return false.

Step 10: Define a function to check whether the queen is in attack position or safe.

Step 11: Check for same column, left diagonal and right diagonal.

Step 12: Print the solution matrix.

Step 13: Stop.

Result:

Hence, the implementation of 8 Queen's Problem is successfully executed.

AMIT SRIVASTAV

RA1911003010633

ARTIFICIAL INTELLIGENCE LAB

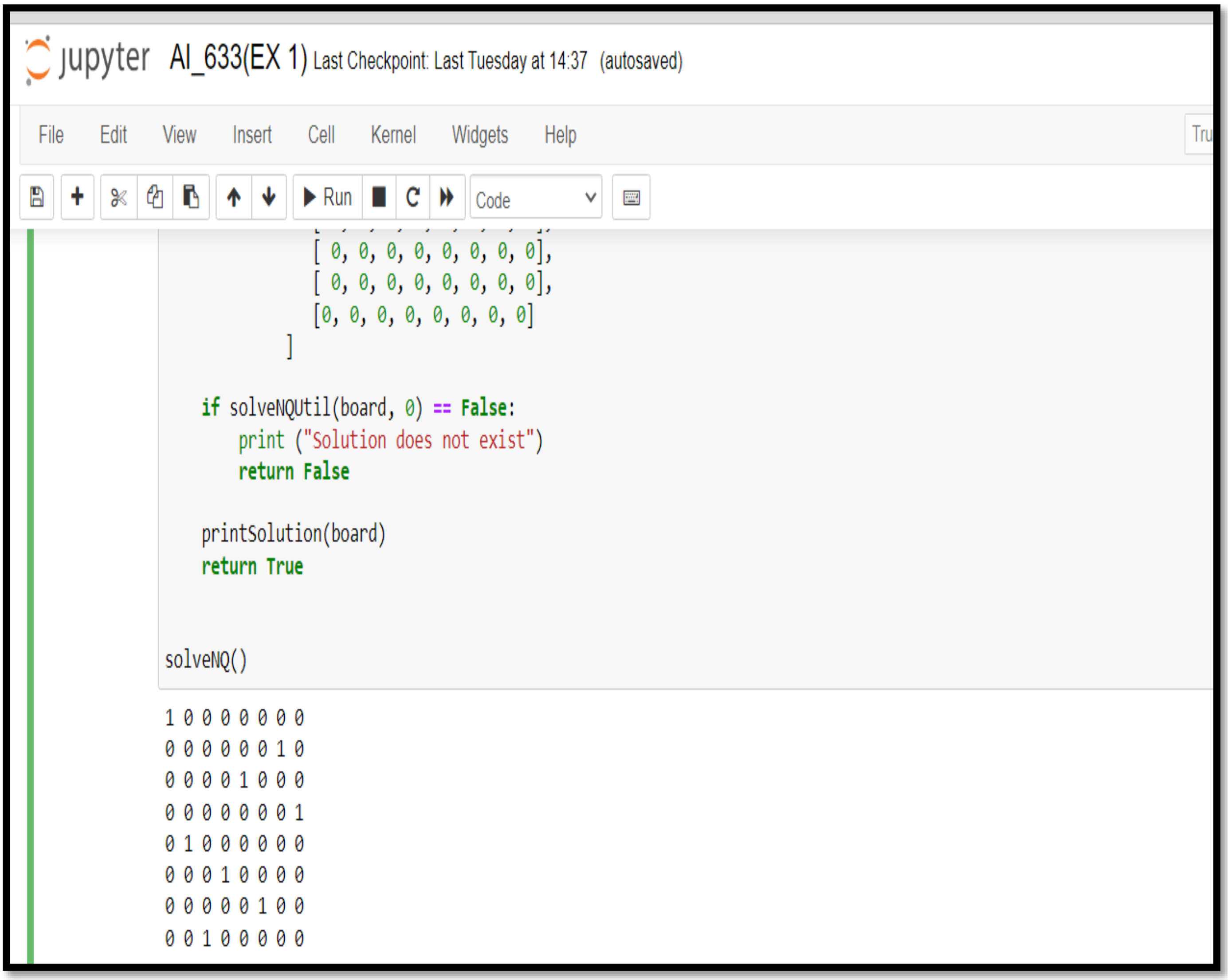
EXPERIMENT NO: 1(A)

IMPLEMENTATION OF TOY PROBLEM (8 QUEEN'S PROBLEM)

Source code:

```
# Python program to solve N Queen
# Problem using backtracking

global N
N = 8

def printSolution(board):
    for i in range(N):
        for j in range(N):
            print (board[i][j],end=' ')
        print()



def isSafe(board, row, col):

    for i in range(col):
        if board[row][i] == 1:
            return False

    for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    for i, j in zip(range(row, N, 1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    return True

def solveNQUtil(board, col):
    if col >= N:
        return True


    for i in range(N):

        if isSafe(board, i, col):
            board[i][col] = 1

            if solveNQUtil(board, col + 1) == True:
                return True


            board[i][col] = 0


    return False


def solveNQ():
    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, 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]
             ]

    if solveNQUtil(board, 0) == False:
        print ("Solution does not exist")
        return False
    printSolution(board)
    return True


solveNQ()
```

Output:


jupyter
AI_633(EX 1) Last Checkpoint: Last Tuesday at 14:37 (autosaved)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Tru
Run
Code

```
                [ 0, 0, 0, 0, 0, 0, 0, 0],
                [ 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0]
             ]

    if solveNQUtil(board, 0) == False:
        print ("Solution does not exist")
        return False

    printSolution(board)
    return True


solveNQ()
```

```
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0
```

Result:

Hence, the implementation of 8 Queen’s Problem is done successfully.