

TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PURWANCHAL CAMPUS
DHARAN



ARTIFICIAL INTELLIGENCE

Lab Report II

Submitted by:

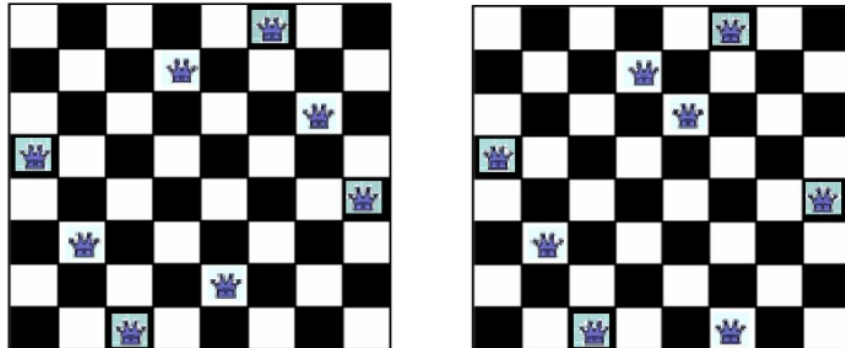
Gokarna Baskota
PUR075BEI013

Submitted To:

Department of Electronics and Computer
Engineering

Lab 2: Problem Solving by Searching: N Queens Problem

Problem Statement: The problem is to place 8 queens on a chessboard so that no two queens are in the same row, column or diagonal. The picture below on the left shows a solution of the 8-queens problem. The picture on the right is not a correct solution, because some of the queens are attacking each other.



Problem Formulation:

- State Space: Any arrangement of k queens in the first k rows such that none are attacked
- Initial state: 0 queens on the board
- Successor function: Add a queen to the $(k+1)$ th row so that none are attacked.
- Goal test: 8 queens on the board, none are attacked

Program:

```
import pprint

class grid:
    def __init__(self, nfquen:int):
        self.queens = nfquen
        self.grid = [[' ' for i in range(self.queens)] for j in range(self.queens)]

    def is_safe(self, posx, posy):
        for row in range(self.queens):
            if(self.grid[row][posy] == 'Q'):
                return False
        for col in range(self.queens):
            if(self.grid[posx][col] == 'Q'):
                return False

        row = posx
        col = posy
        while(row>=0 and col>=0):
            if(self.grid[row][col] == 'Q'):
                return False
            row -= 1
            col -= 1
        row = posx
        col = posy
        while(row<self.queens and col<self.queens):
            if(self.grid[row][col] == 'Q'):
```

```

        return False
        row += 1
        col += 1
    row = posx
    col = posy
    while(row<self.queens and col>=0):
        if(self.grid[row][col] == 'Q'):
            return False
        row += 1
        col -= 1
    row = posx
    col = posy
    while(row>=0 and col<self.queens):
        if(self.grid[row][col] == 'Q'):
            return False
        row -= 1
        col += 1

    return True

def palce_queen(self, posx, posy):
    self.grid[posx][posy] = 'Q'

def reset(self, posx, posy):
    self.grid[posx][posy] = ' '

def show(self):
    pprint.pprint(self.grid)

def nqueen(board, index, queens):
    if(index >= queens):
        return True

    for col in range(queens):
        if(board.is_safe(index, col)):
            board.palce_queen(index, col)
            if(nqueen(board, index+1, queens)):
                return True
            board.reset(index, col)
    return False

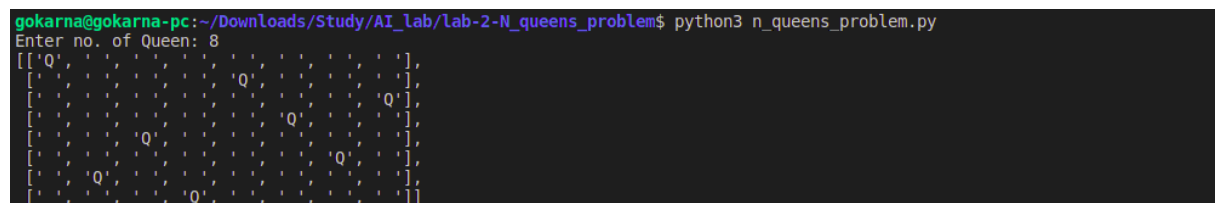
nfqueen = int(input("Enter no. of Queen: "))
board = grid(nfqueen)

done = nqueen(board, 0, nfqueen)

if(done):
    board.show()
else:
    print("No solutions!");

```

Output:



```

gokarna@gokarna-pc:~/Downloads/Study/AI_lab/lab-2-N_queens_problem$ python3 n_queens_problem.py
Enter no. of Queen: 8
[['Q', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', 'Q', ' ', ' ', ' '],
 [' ', ' ', ' ', 'Q', ' ', ' ', ' ', ' '],
 [' ', ' ', 'Q', ' ', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', ' ', 'Q', ' ', ' '],
 [' ', 'Q', ' ', ' ', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', 'Q', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', ' ', ' ', 'Q', ' ']]

```