

Exp No 1

Date: 24-8-24

N-Queen Problem

AM

→ To place N Queen on a $N \times N$ chessboard
Such that no two queen be in same row,
Column or diagonal

Algorithm

- * create $n \times n$ with empty position
- * place queens recursively one on each column
- * Before placing check - no same row
- no left-right diagonal
- * if safe place & move to next column
- * if no solution backtrack
- * Continue upto final step

Code

```
def is-safe(board, row, col, n):  
    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), range(col, -1, -1)):  
        if board[i][j] == 1:  
            return False  
  
    return True
```

```
def solve_nqueen(board, col, n):
```

```
    if col >= n:
```

```
        return True
```

```
    for i in range(n):
```

```
        if is_safe(board, i, col, n):
```

```
            board[i][col] = 1
```

```
            if solve_nqueen(board, col+1, n):
```

```
                return True
```

```
            board[i][col] = 0
```

```
    return False
```

```
def print_board(board, n):
```

```
    for i in range(n):
```

```
        for j in range(n):
```

```
            print("@" if board[i][j] == 1
```

```
                else '.', end=" ")
```

```
        print()
```

```
def n_queens(n):
```

```
    board = [[0] * n for _ in range(n)]
```

```
    if not solve_nqueen(board, 0, n):
```

```
        print("solution does not exist")
```

```
        return False
```

```
    print_board(board, n)
```

```
    return True
```

```
n = int(input("Enter no of Queens: "))
```

```
n_queen(n)
```

Output

```
Enter no of Queens: 8
```

```
Q . . . . .
. . . . Q .
. . . Q . .
. . . . . Q
Q . . . . .
. . . Q . .
. . Q . . .
. Q . . . .
```

RESULT

→ Thus NQueen Problem
is Solved & executed
Successfully & output is
verified