Backtracking:-

```
global N
N = 4


def printSolution(board):
    for i in range(N):
        for j in range(N):
            if board[i][j] == 1:
                print("1",end=" ")
            else:
                print("0",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]]

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

    printSolution(board)
    return True

if __name__ == '__main__':
    solveNQ()
```

Output:-

```
student@student:~$ python3 Backtraking.py
0   0   1   0
1   0   0   0
0   0   0   1
0   1   0   0
```

Branch and Bound:-

```
N = 8

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

def isSafe(row, col, slashCode, backslashCode,
           rowLookup, slashCodeLookup,
                    backslashCodeLookup):
    if (slashCodeLookup[slashCode[row][col]] or
        backslashCodeLookup[backslashCode[row][col]] or
        rowLookup[row]):
        return False
    return True


def solveNQueensUtil(board, col, slashCode, backslashCode,
                     rowLookup, slashCodeLookup,
                     backslashCodeLookup):

    if(col >= N):
        return True
    for i in range(N):
        if(isSafe(i, col, slashCode, backslashCode,
                  rowLookup, slashCodeLookup,
                  backslashCodeLookup)):

            board[i][col] = 1
            rowLookup[i] = True
            slashCodeLookup[slashCode[i][col]] = True
            backslashCodeLookup[backslashCode[i][col]] = True

            if(solveNQueensUtil(board, col + 1,
                                slashCode, backslashCode,
                                rowLookup, slashCodeLookup,
                                backslashCodeLookup)):
                return True

            board[i][col] = 0
            rowLookup[i] = False
            slashCodeLookup[slashCode[i][col]] = False
            backslashCodeLookup[backslashCode[i][col]] = False

    return False

def solveNQueens():
    board = [[0 for i in range(N)]
```

```
            for j in range(N)]

    slashCode = [[0 for i in range(N)]
                 for j in range(N)]
    backslashCode = [[0 for i in range(N)]
                     for j in range(N)]

    rowLookup = [False] * N

    x = 2 * N - 1
    slashCodeLookup = [False] * x
    backslashCodeLookup = [False] * x

    for rr in range(N):
        for cc in range(N):
            slashCode[rr][cc] = rr + cc
            backslashCode[rr][cc] = rr - cc + 7

    if(solveNQueensUtil(board, 0, slashCode, backslashCode,
                        rowLookup, slashCodeLookup,
                        backslashCodeLookup) == False):
        print("Solution does not exist")
        return False

    printSolution(board)
    return True

solveNQueens()
```

Output:-

```
student@student:~$ python3 BranchAndBound.py
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
```