SRM INSTITUTE OF SCIENCE & TECHNOLOGY

DEPARTMENT OF NETWORKING & COMMUNICATIONS

18CSC305J-ARTIFICIAL INTELLIGENCE

SEMESTER – 6

BATCH-2

REG. NO.	**RA1911028010016**
NAME	**NISHANT SAGAR**

INDEX

Exercise: 1

Date : 06/01/2022

N-Queens Problem

Problem Statement : The N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other.

The idea is to place queens one by one in different columns, starting from the leftmost column. When we place a queen in a column, we check for clashes with already placed queens. In the current column, if we find a row for which there is no clash, we mark this row and column as part of the solution. If we do not find such a row due to clashes then we backtrack and return false.

Algorithm :

1) Start in the leftmost column

2) If all queens are placed return true

3) Try all rows in the current column.

Do the 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
The 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.

4) If all rows have been tried and nothing worked,

return false to trigger backtracking.

Optimization technique :

The idea is not to check every element in right and left diagonal instead use property of diagonals:

1.The sum of i and j is constant and unique for each right diagonal where i is the row of elements and j is the column of elements.

2.The difference of i and j is constant and unique for each left diagonal where i and j are row and column of element respectively.

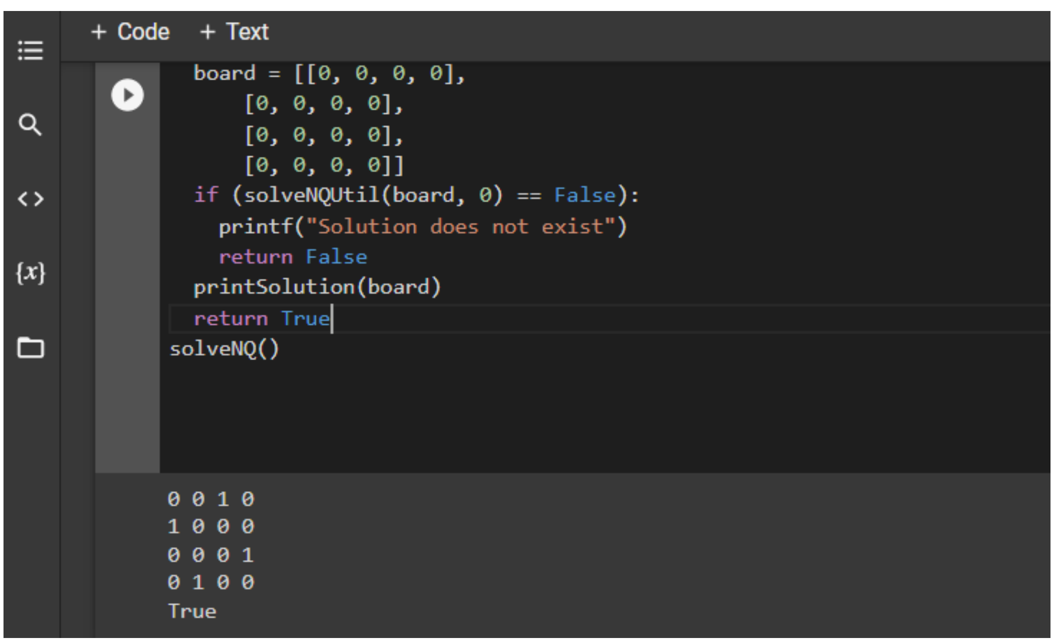
Tool : VS Code and Python 3.9.0

Programming code :

```
N = 4

ld = [0] * 30

rd = [0] * 30

cl = [0] * 30

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

def solveNQUtil(board, col):

		if (col >= N):
		return True

		for i in range(N):

				if ((ld[i - col + N - 1] != 1 and
			rd[i + col] != 1) and cl[i] != 1):


			board[i][col] = 1
			ld[i - col + N - 1] = rd[i + col] = cl[i] = 1


			if (solveNQUtil(board, col + 1)):
```

```
                        return True

                board[i][col] = 0 # BACKTRACK
                ld[i - col + N - 1] = rd[i + col] = cl[i] = 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):
                printf("Solution does not exist")
                return False
        printSolution(board)
        return True


solveNQ()
```

Output screen shots :



Result : Successfully found out the positions where the queens can be placed represented by 1 in the matrix.