LACS-Elite-Part009

Solved Challenges 0/1

Back To Challenges List

N Bishops - Fill Remaining

ID:12942 Solved By 199 Users

In a N*N square chessboard, certain number of bishops are placed. The program must accept the positions of B bishops(marked by 1). The empty squares are marked by the value 0. The program must fill in the remaining N-B bishops with the condition that only one bishop must be in a row and one bishop must be in a column so that the bishops do not attack each other. The program must finally print the arrangement of the bishops in the chessboard (if multiple arrangements are possible print the possibility which occurs with the left most column filled starting from the top row). If such an arrangement is not possible, then the program must print NotPossible as the output.
Note: In Chess, a bishop can move diagonally. The movement can happen till the end of the board is reached or another piece (like Rook, Knight, Bishop, Pawn etc is encountered). But in this program only the N bishops are placed and no other pieces will be present on the board.

Boundary Condition(s):
2 <= N <= 20
0 <= B <= N-1

Input Format:
The first line contains N.
The next N lines contain N values (0 or 1) each separated by a space.

Output Format:
The first N lines contain N values (0 or 1) each separated by a space.

Example Input/Output 1:
Input:
4
1 0 0 0
0 0 0 0
0 0 0 1
0 1 0 0

Output:
1 0 0 0
1 0 0 0
0 0 0 1
0 1 0 0

Example Input/Output 2:
Input:
3
0 0 0
0 0 0
0 0 0

Output:
1 0 0
1 0 0
1 0 0

Explanation:
Here multiple arrangements are possible. But starting from the top row, we consider the possibility once the left most column is filled.

Example Input/Output 3:
Input:
4
1 0 0 0
0 0 0 0
0 1 0 0
0 0 0 0

Output:
NotPossible

Max Execution Time Limit: 50 millisecs

Ambiance

Python3 (3.x)

Reset

```
def canPlace(N,row,col,bishopRow,swDiagonal,nwDiagonal):
    if(row == N):
        return True
    if(bishopRow[row] == True):
        return canPlace(N, row+1,col,bishopRow,swDiagonal,nwDiag
    for col in range(N):
        if(nwDiagonal[row + col]==False and swDiagonal[col-row+N
            board[row][col] = 1
            bishopRow[row] = True
            nwDiagonal[row + col] = True
            swDiagonal[col - row + N-1] = True
            if(canPlace(N, row+1,col,bishopRow,swDiagonal,nwDiag
                return True
            else:
                board[row][col] = 0
                bishopRow[row] = False
                nwDiagonal[row + col] = False
                swDiagonal[col-row+N-1] = False

    return False

N = int(input())

board = []
bishopRow = [False]*N
swDiagonal = [False]*(2*N-1)
nwDiagonal = [False]*(2*N-1)
for row in range(N):
    row = list(map(int , input().split()))
    board.append(row)

for row in range(N):
    for col in range(N):
        if(board[row][col] == 1):
            bishopRow[row] = True
            swDiagonal[col - row + N - 1] = True
            nwDiagonal[col + row] = True

if(canPlace(N, 0,col,bishopRow,swDiagonal,nwDiagonal)):
    for row in range(N):
        for col in range(N):
            print(board[row][col], end=" ")
        print()
else:
    print("NotPossible")

```

Code did not pass the execution

TestCase ID: 86135

Input:

```
4
1 0 0 0
0 0 0 0
0 1 0 0
0 0 0 0
```

Expected Output:

```
NotPossible
```

Your Program Output:

```
Not possible
```

Save Run

Run with a custom test case (Input/Output)