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

1000

0000

0001

0100

Output:

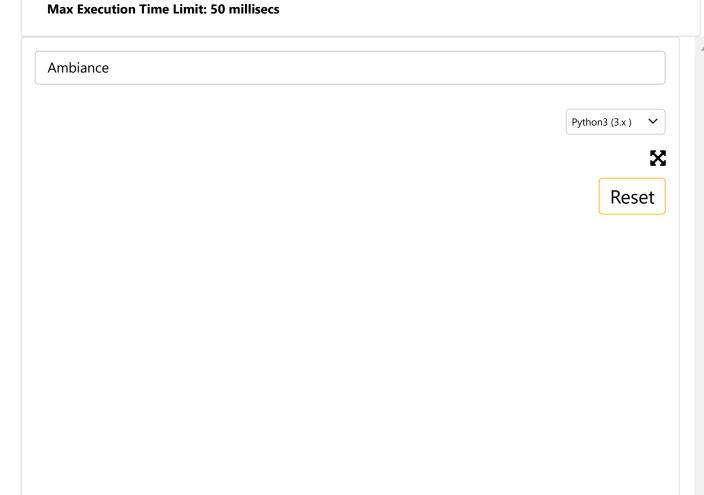
1000

1000

0001

0100

Example Input/Output 2: Input: 3 000 000 000 Output: 100 100 100 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 1000 0000 0100 0000 Output: NotPossible



```
def canPlace(N,row,col,bishopRow,swDiagonal,nwDiagonal):
 1
 2
        if(row == N):
 3
            return True
 4
        if(bishopRow[row] == True):
            return canPlace(N, row+1,col,bishopRow,swDiagonal,nwDiagonal
 5
 6
        for col in range(N):
 7
            if(nwDiagonal[row + col]==False and swDiagonal[col-row+N-
 8
                board[row][col] = 1
9
                bishopRow[row] = True
                nwDiagonal[row + col] = True
10
                swDiagonal[col - row + N-1] = True
11
                if(canPlace(N, row+1,col,bishopRow,swDiagonal,nwDiagonal)
12
                     return True
13
14
                else:
15
                     board[row][col] = 0
                     bishopRow[row] = False
16
                     nwDiagonal[row + col] = False
17
                     swDiagonal[col-row+N-1] = False
18
19
20
        return False
21
22
    N = int(input())
23
24
   board = []
25
   bishopRow = [False]*N
    swDiagonal = [False]*(2*N-1)
26
    nwDiagonal = [False]*(2*N-1)
27
    for row in range(N):
28
29
        row = list(map(int , input().split()))
        board.append(row)
30
31
32
    for row in range(N):
33
        for col in range(N):
34
            if(board[row][col] == 1):
                bishopRow[row] = True
35
                swDiagonal[col - row + N - 1] = True
36
                nwDiagonal[col + row] = True
37
38
39
    if(canPlace(N, 0,col,bishopRow,swDiagonal,nwDiagonal)):
        for row in range(N):
40
41
            for col in range(N):
42
                print(board[row][col], end=" ")
43
            print()
44
    else:
        print("NotPossible")
45
46
```

