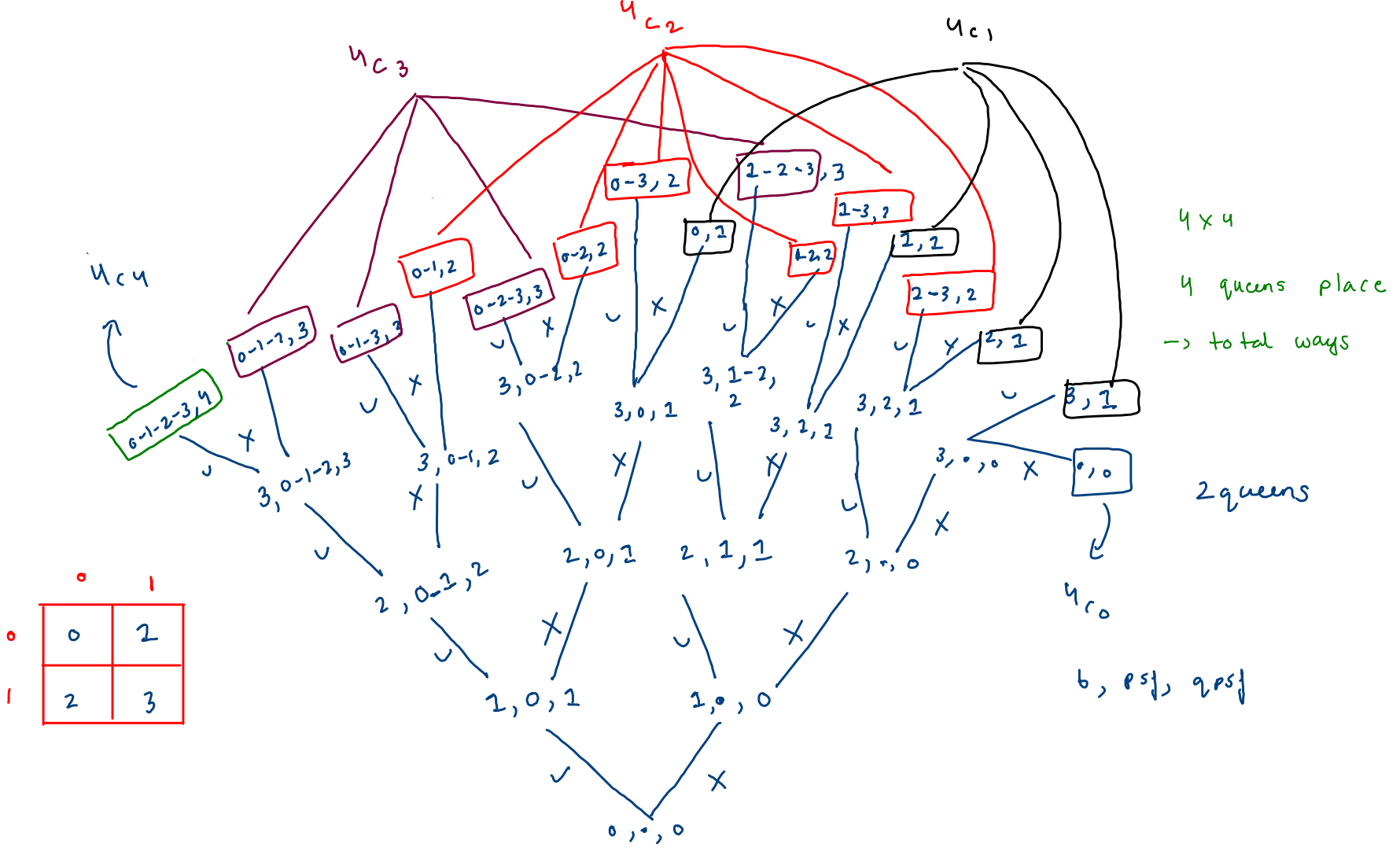

4 x 4
4 queens place
-> total ways
2 queens
b, psf, qsf
0 1
0 0 2
1 2 3

N x N matrix ,

boxes -> 4 = n

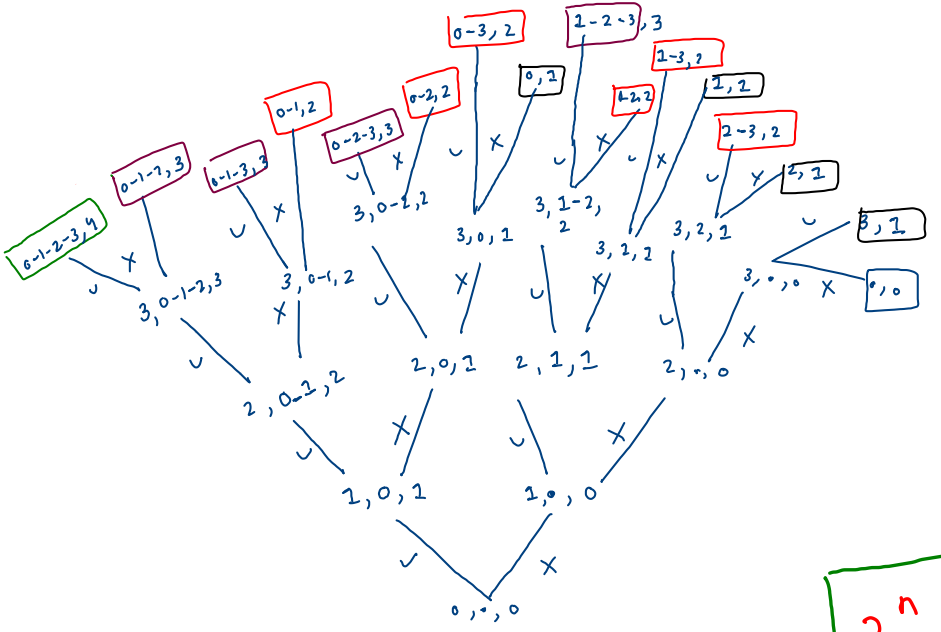
0 queens -> 1 = $^4C_0$

1 queens -> 4 = $^4C_1$

2 queens -> 6 = $^4C_2$

3 queen -> 4 = $^4C_3$

4 queen -> 1 = $^4C_4$



$2^n = {}^nC_0 + {}^nC_1 + {}^nC_2 + \cdots {}^nC_n$

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

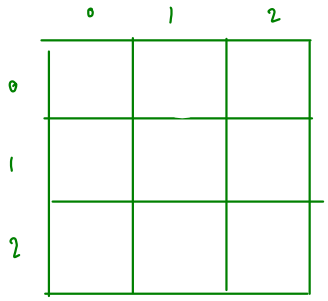
i, j to b.no.

$$b = i * n + j$$

b.no to i, j

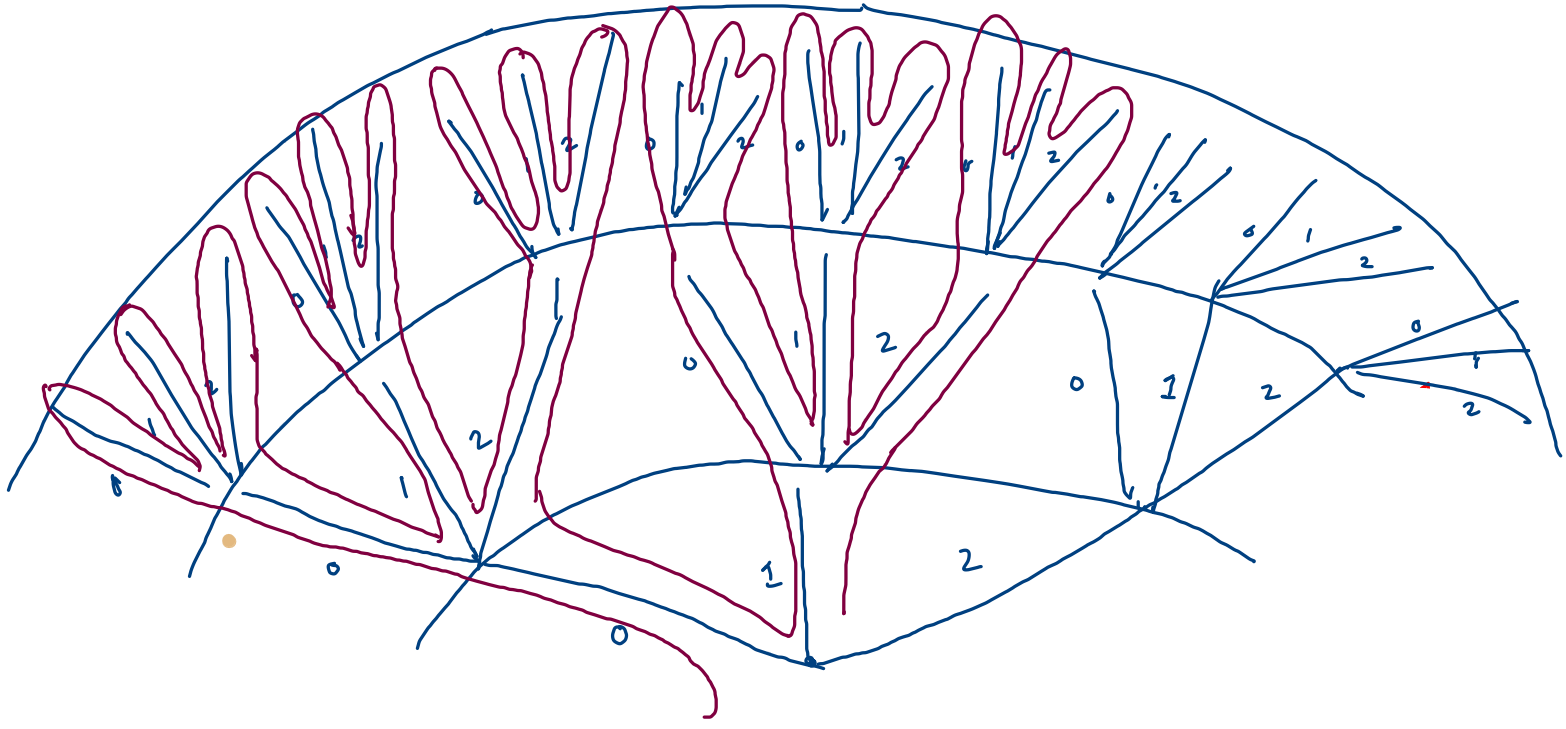
$$i = b / n$$

$$j = b \% n$$

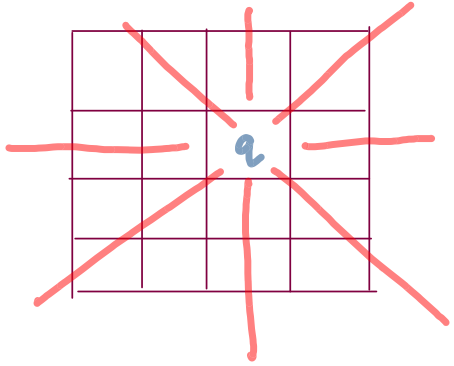


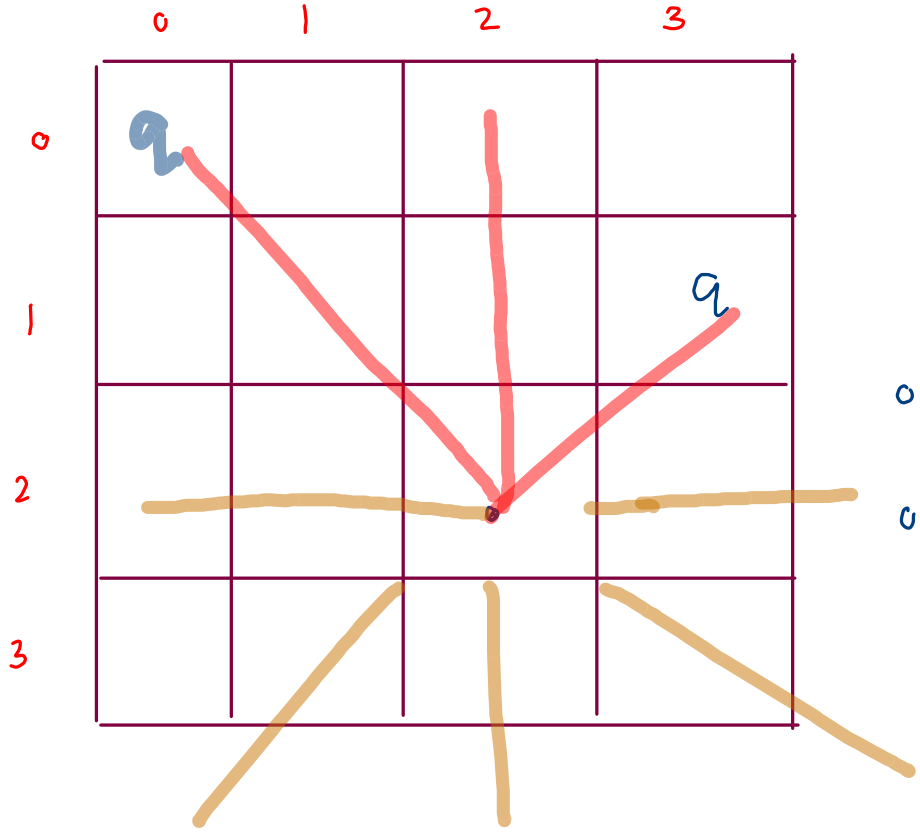
NxN, matrix N queen

each row -> one queen



N-queens





0-0, 1-2

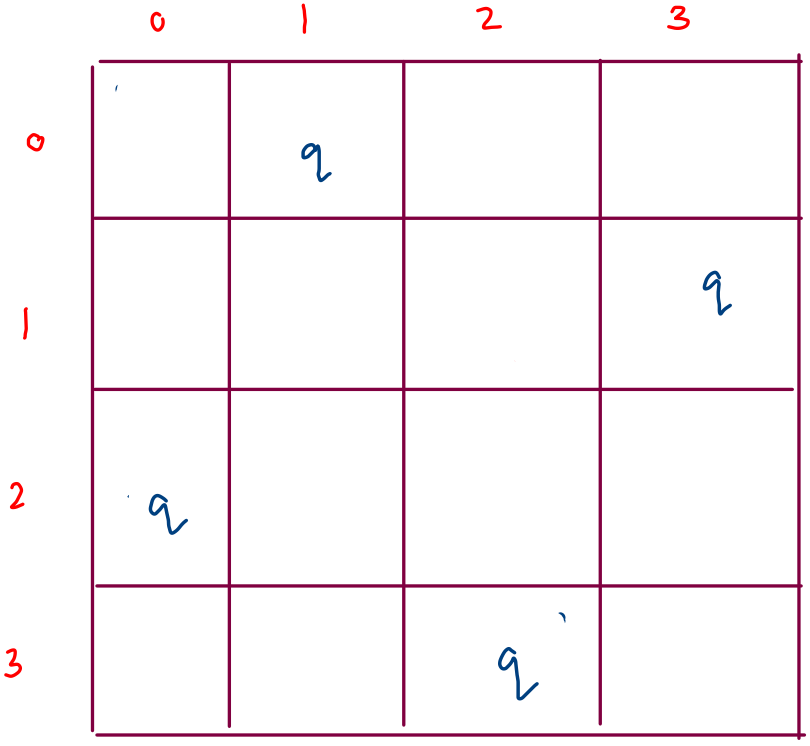
0-1, 1-3, 2-0, 3-2

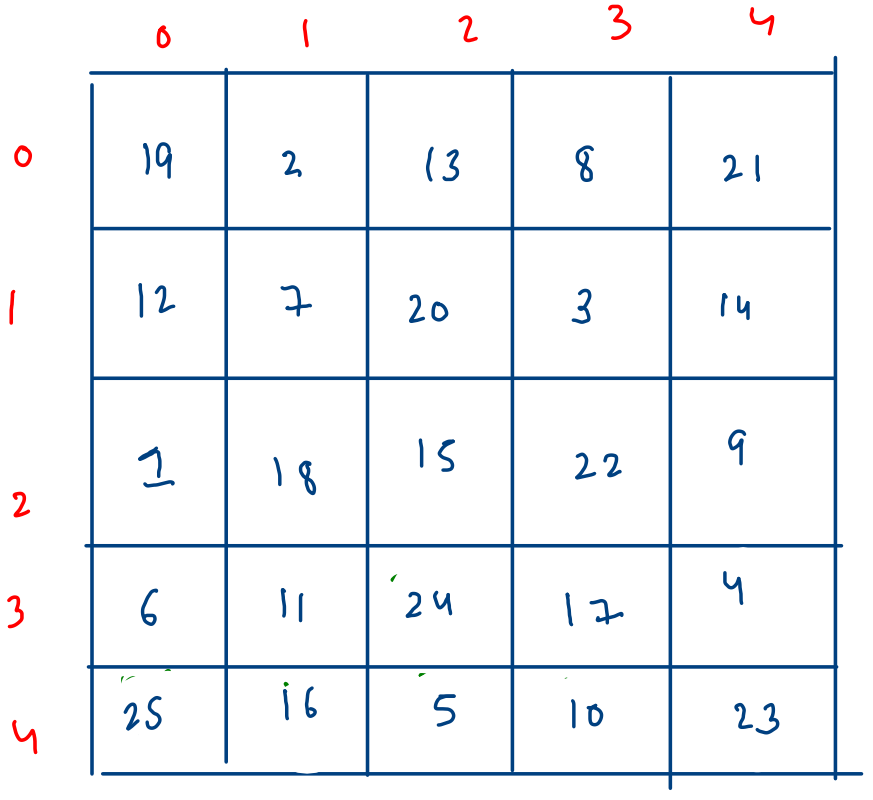
0-2, 1-0, 2-3, 3-1

```java
public static boolean isQueenSafe(int[][]chess,int r,int c) {
    //vertical
    for(int i = r-1; i >= 0;i--) {
        if(chess[i][c] == 1) {
            return false;
        }
    }

    //top right diagonal
    for(int i = r-1, j = c+1; i >= 0 && j < chess.length;i--,j++) {
        if(chess[i][j] == 1) {
            return false;
        }
    }

    //top left diagonal
    for(int i = r-1, j = c-1; i >= 0 && j >= 0; i--,j--) {
        if(chess[i][j] == 1) {
            return false;
        }
    }

    return true;
}
```

01 - 13 - 20 - 32

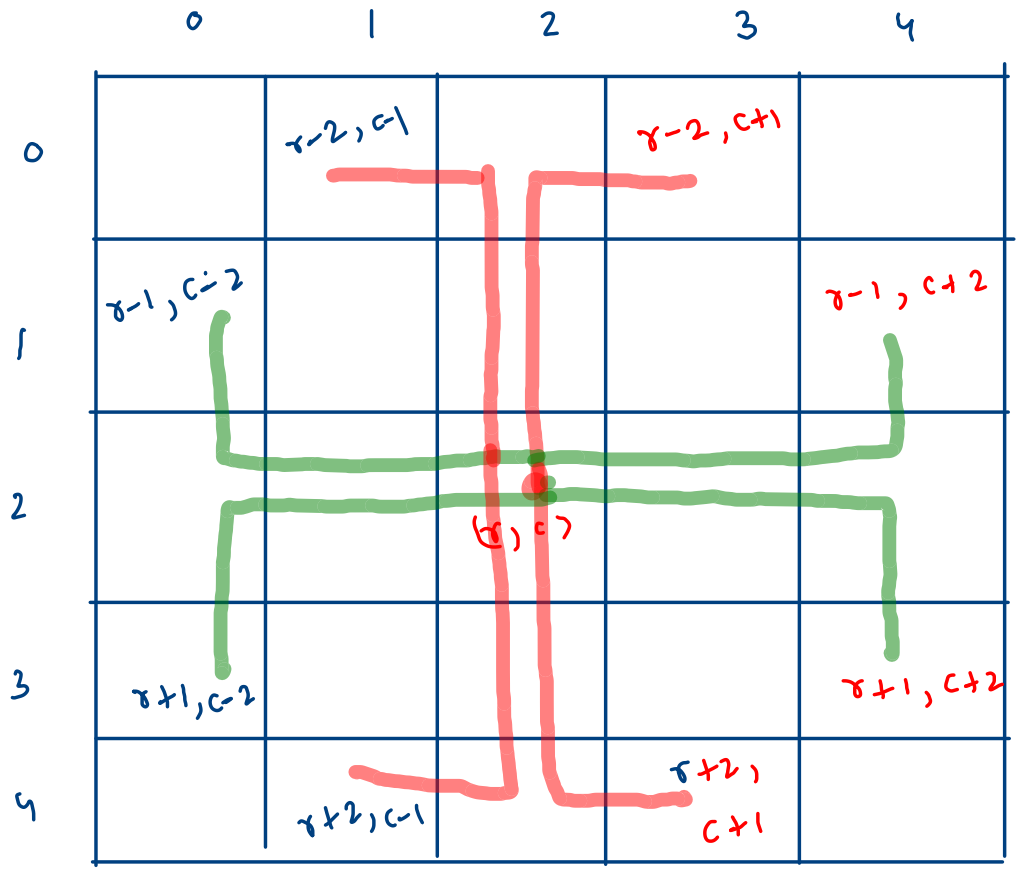
```java
public static void printNQueens(int[][] chess, String psf, int row) {
    if(row == chess.length) {
        System.out.println(psf + ".");
        return;
    }


    for(int col = 0; col < chess.length;col++) {
        if(isQueenSafe(chess,row,col) == true) {
            //place the queen at row,col
            chess[row][col] = 1;
            //make call to next row
            printNQueens(chess,psf + row + "-" + col + ", ",row + 1);
            //unplace the queen at row,col
            chess[row][col] = 0;
        }
    }
```






```
25 2 13 8 23
12 7 24 3 14
1 18 15 22 9
6 11 20 17 4
19 16 5 10 21
```

```java
public static void printKnightsTour(int[][] chess, int r, int c, int move) {
    if(r < 0 || c < 0 || r >= chess.length || c >= chess.length || chess[r][c] != 0) {
        return;
    }

    if(move == chess.length * chess.length) {
        chess[r][c] = move;
        displayBoard(chess);
        chess[r][c] = 0;
        return;
    }


    chess[r][c] = move;

    printKnightsTour(chess,r-2,c+1,move+1);
    printKnightsTour(chess,r-1,c+2,move+1);
    printKnightsTour(chess,r+1,c+2,move+1);
    printKnightsTour(chess,r+2,c+1,move+1);
    printKnightsTour(chess,r+2,c-1,move+1);
    printKnightsTour(chess,r+1,c-2,move+1);
    printKnightsTour(chess,r-1,c-2,move+1);
    printKnightsTour(chess,r-2,c-1,move+1);

    chess[r][c] = 0;

}
```


0
1
2
3
4
0
1
2
3
4
r-2, c-1
r-2, c+1
r-1, c-2
r-1, c+2
(r, c)
r+1, c-2
r+1, c+2
r+2, c-1
r+2, c+1