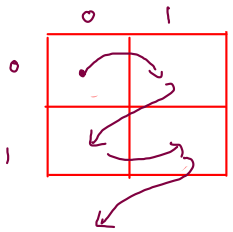
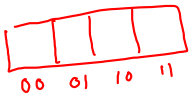
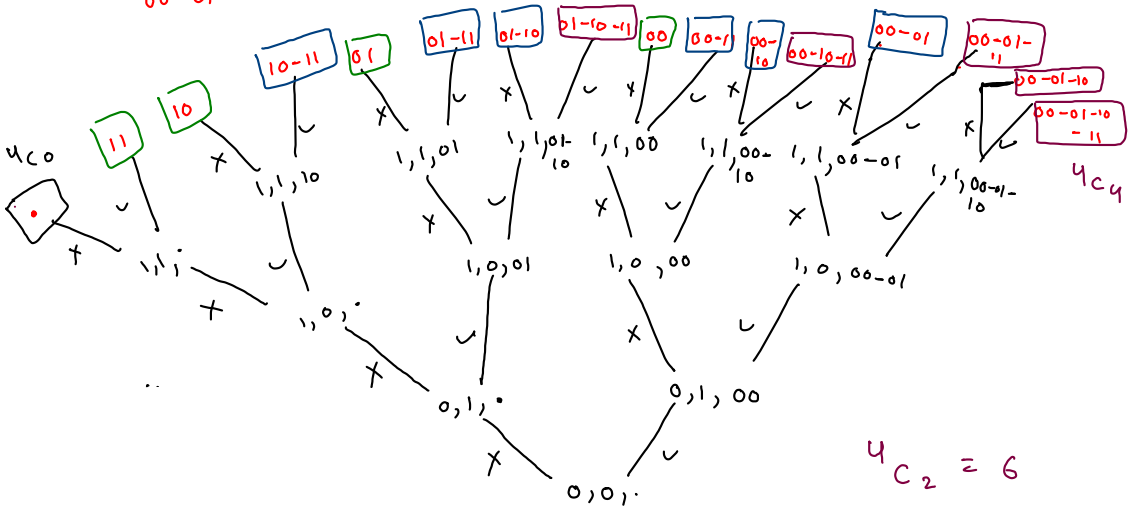
N - queens





subsets



N x N matrix

boxes = 4

queens = 2

$^4C_2 = 6$

$^4C_0, {}^4C_1, {}^4C_2, {}^4C_3, {}^4C_4$

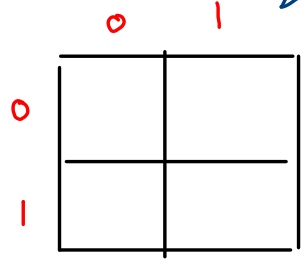
N -> order $2^n = {}^nC_0 + {}^nC_1 + {}^nC_2 + \cdots {}^nC_n$

n -> N x N (boxes) $2^4 = {}^4C_0 + {}^4C_1 + {}^4C_2 + {}^4C_3 + {}^4C_4$

$= 1 + 4 + 6 + 4 + 1$

$= 16$

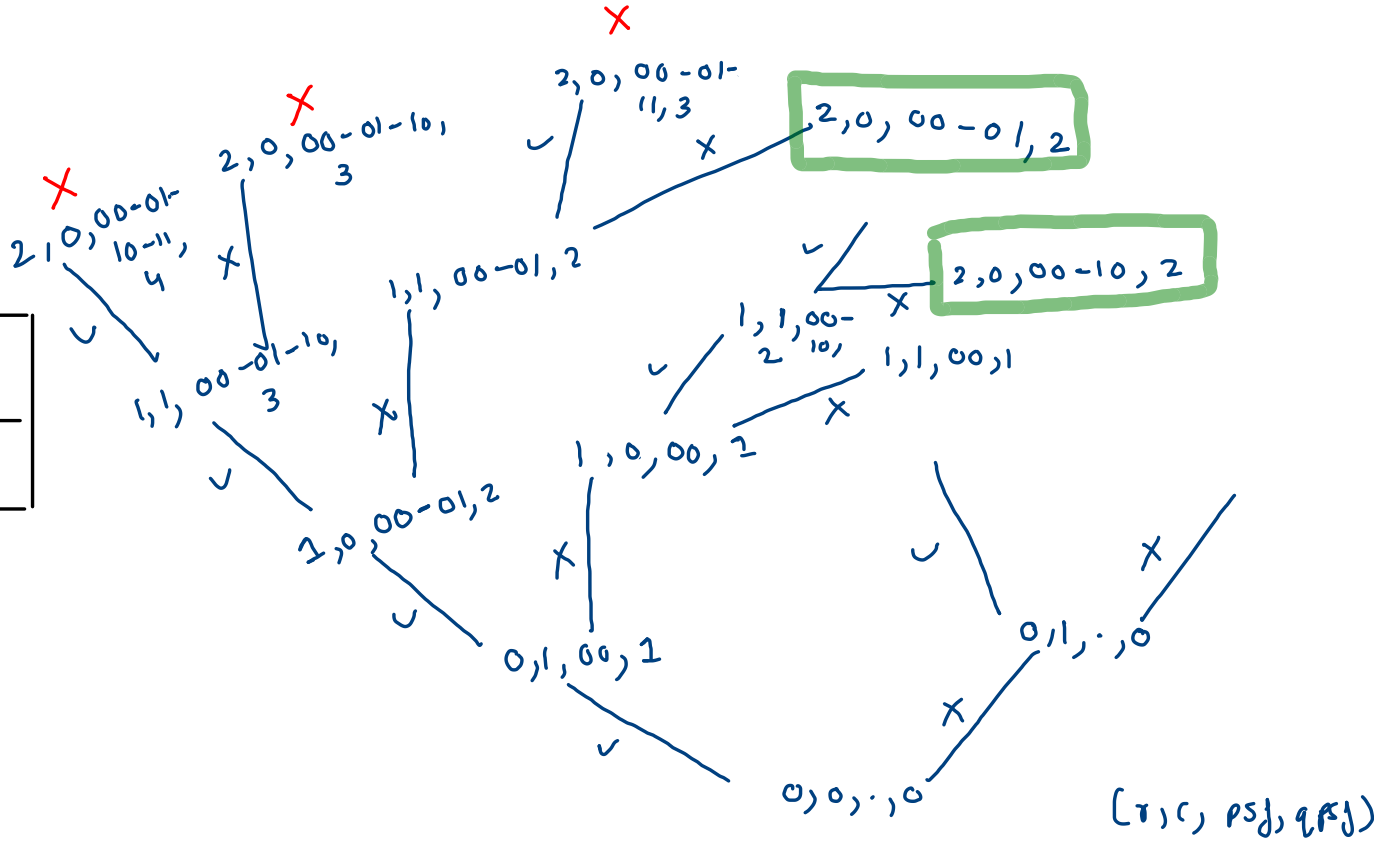
```java
public static void nqueens(int r,int c,String psf,int qpsf,int N) {
    if(c == N) {
        r = r+1;
        c = 0;
    }

    if(r == N) {
        if(qpsf == N) {
            System.out.println(psf);
        }
        return;
    }

    //yes
    nqueens(r,c+1,psf + r + c + "-",qpsf + 1,N);

    //no
    nqueens(r,c+1,psf,qpsf,N);
}
```

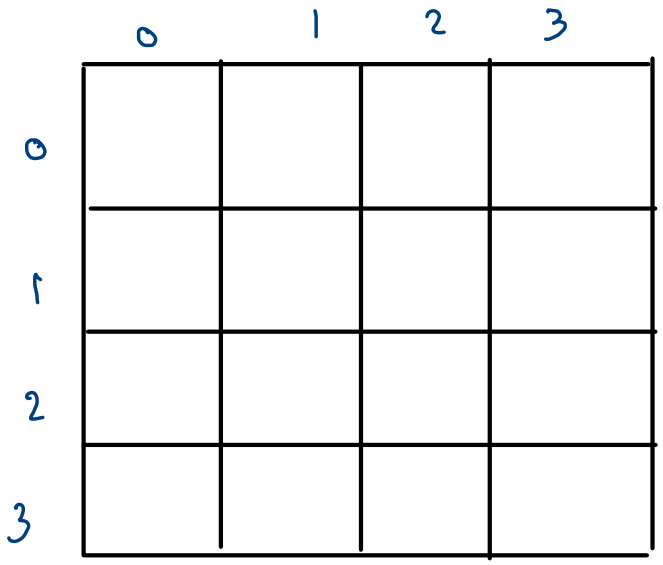


N = 2

n = 4 (boxes)

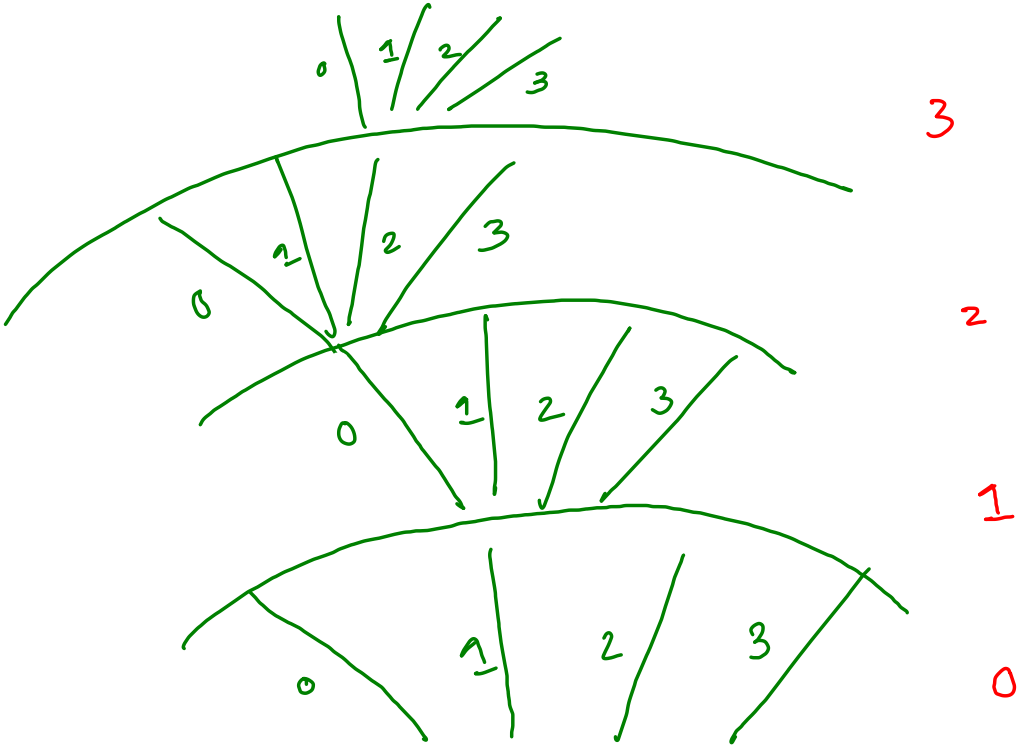
NxN box , N queen



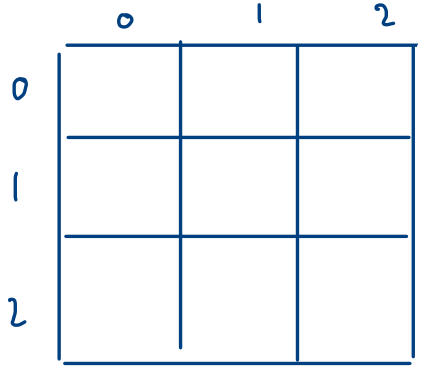


NXN boxes

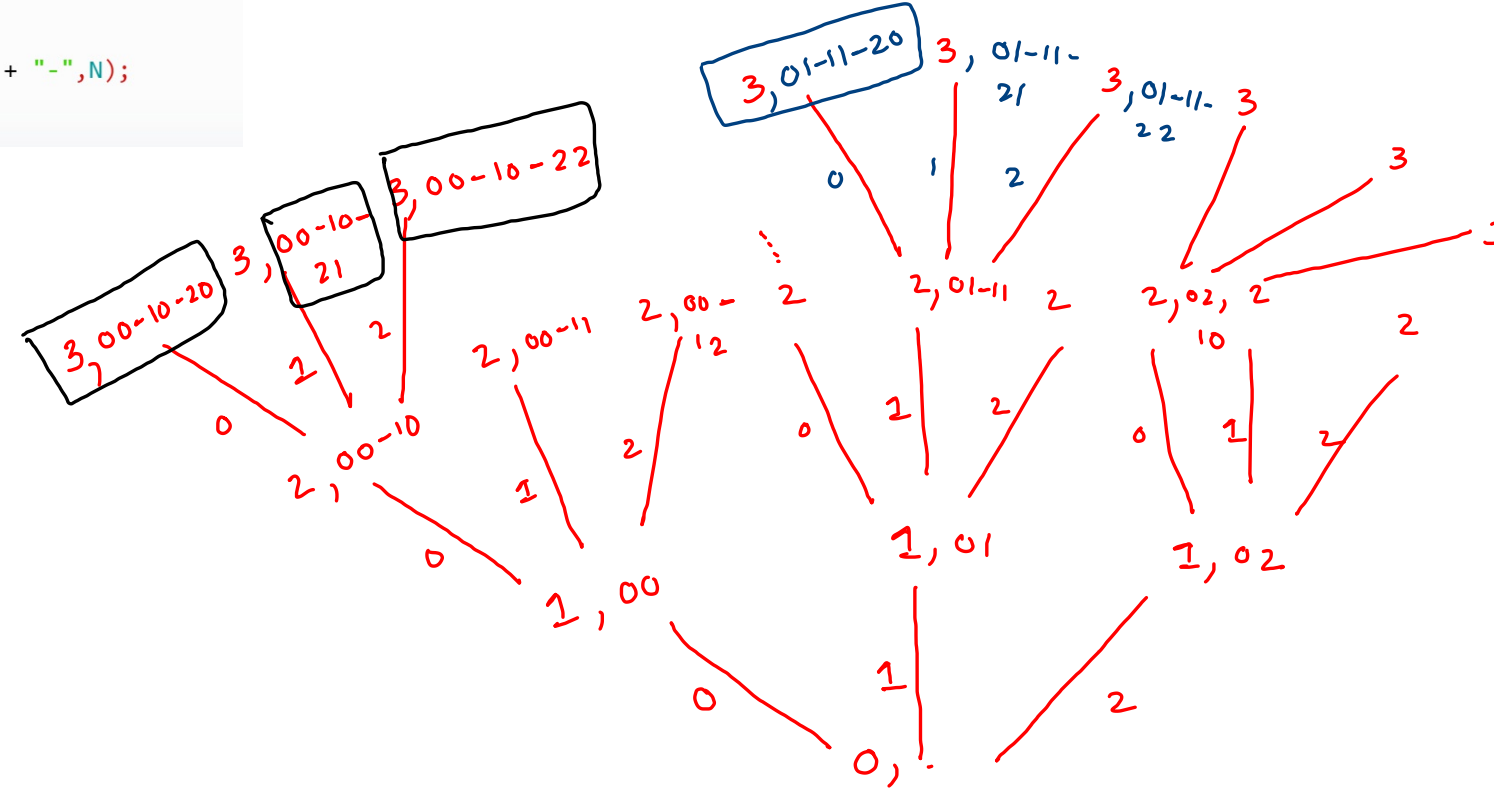
N queens, thre is a single queen each row



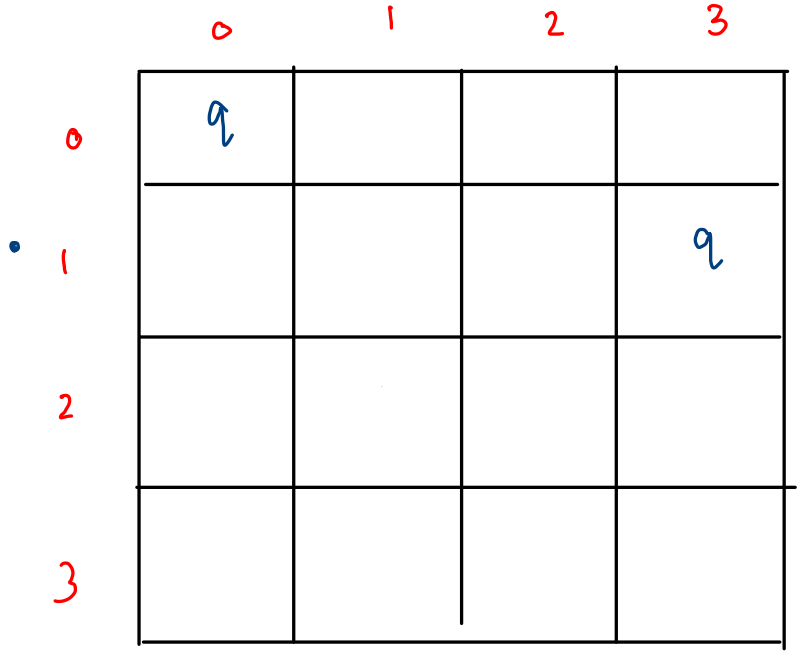
```java
public static void nqueens(int row,String psf,int N) {
    if(row == N) {
        System.out.println(psf);
        return;
    }

    for(int col = 0; col < N;col++) {
        nqueens(row+1,psf + row + col + "-",N);
    }
}
```



N = 3



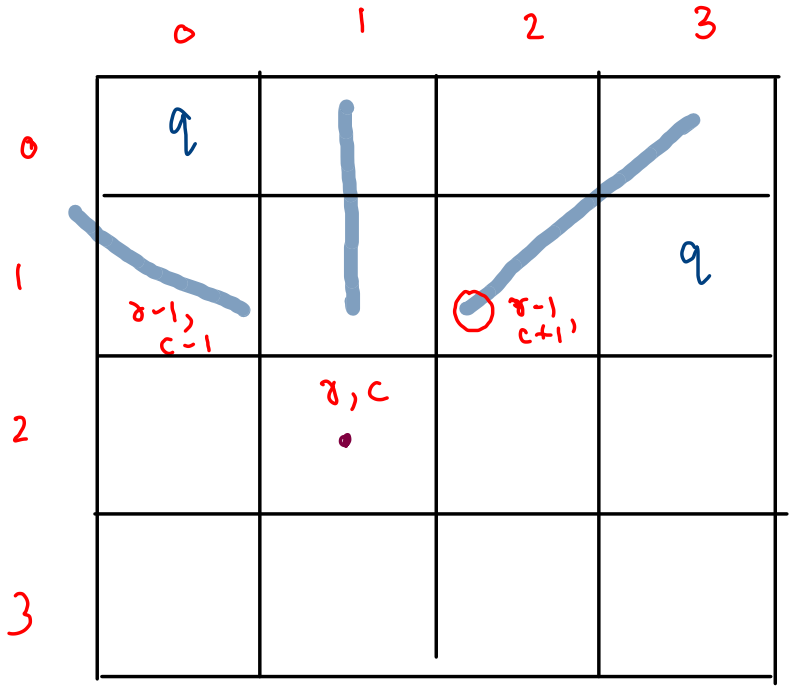
with safety

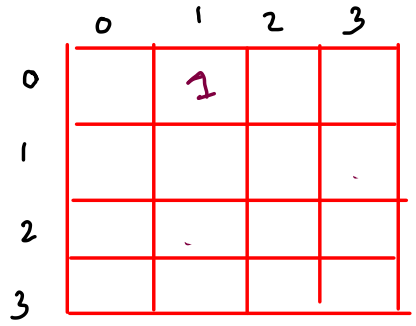



```java
public static void nqueens(int row,String psf,int N) {
    if(row == N) {
        System.out.println(psf);
        return;
    }

    for(int col = 0; col < N;col++) {
        nqueens(row+1,psf + row + col + "-",N);
    }
}
```

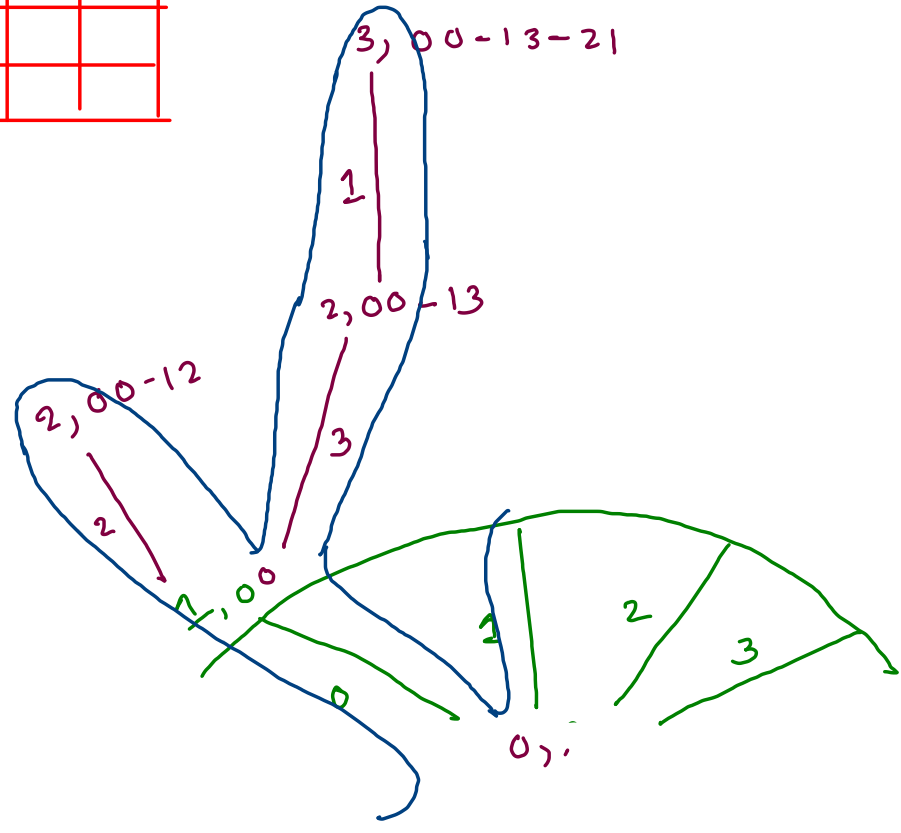
01-13-20-32

02-10-23-31

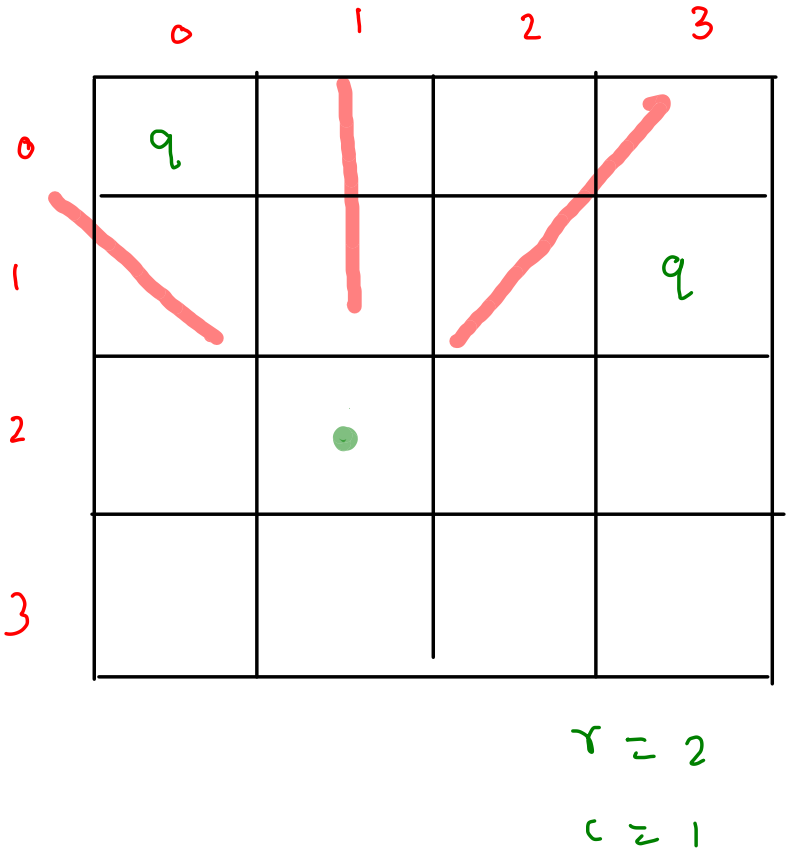

0
1
2
3
0
1
2
3
q
q
r-1,
c-1
r-1,
c+1
r, c




```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
            chess[row][col] = 1;

            printNQueens(chess,psf + row + "-" + col + ", ",row+1);

            //unplace
            chess[row][col] = 0;
        }
    }
```

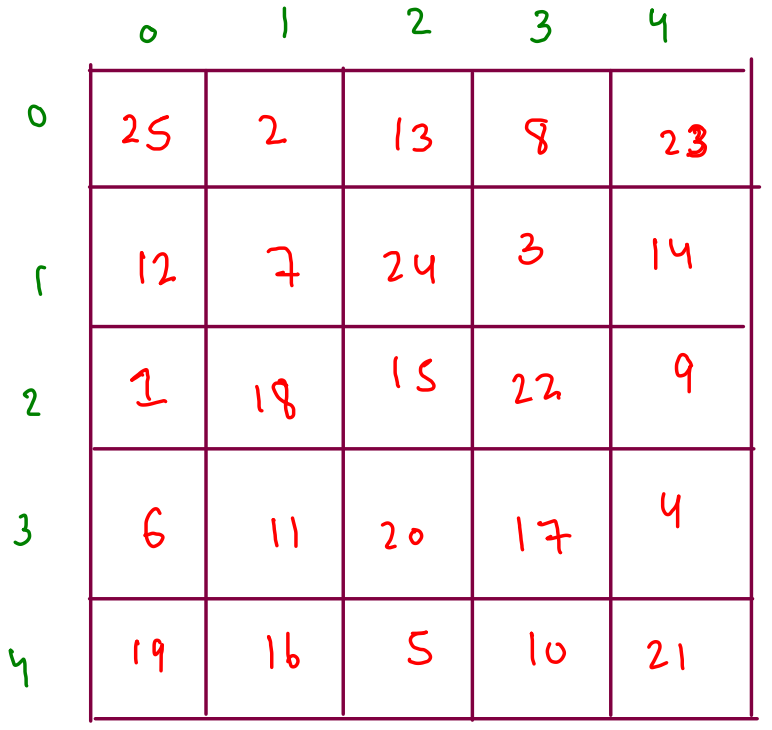


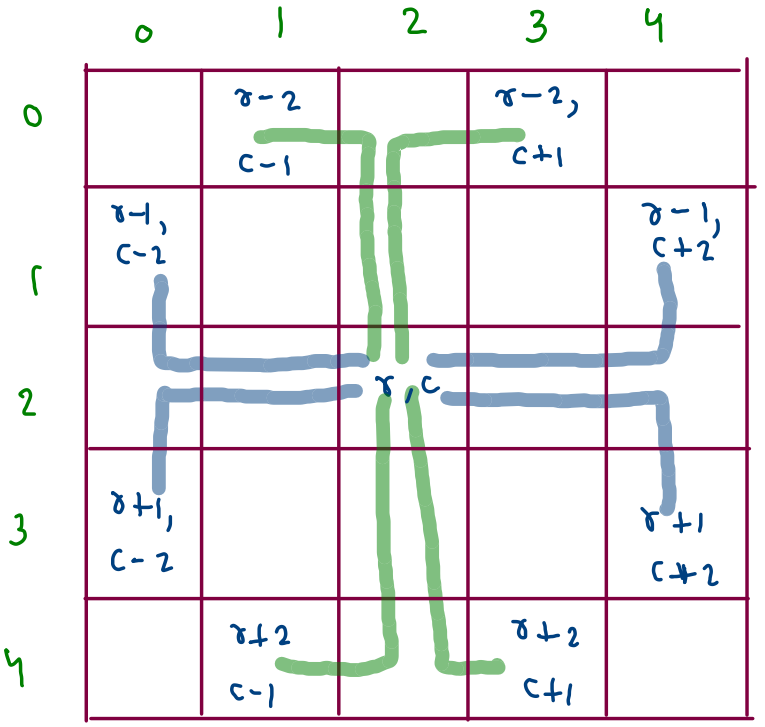
```java
public static boolean isQueenSafe(int[][]chess,int r,int c) {
    //col check
    for(int i = r-1; i >= 0; i--) {
        if(chess[i][c] == 1) {
            return false;
        }
    }

    //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;
}
```



```
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
```

```
5
2
0
```




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