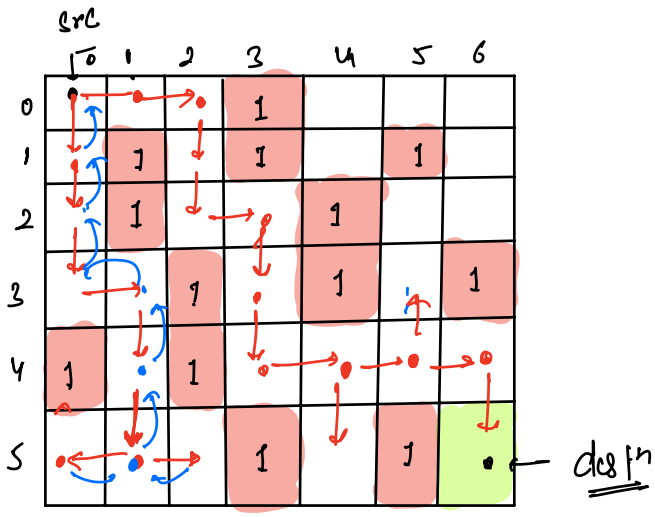
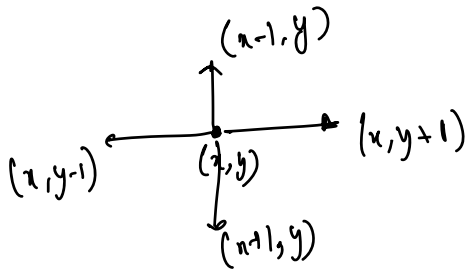
Rat in a Maze.

$N=6,\ m=7$



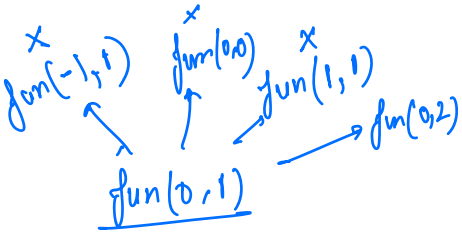
src → (0,0)

dst → (N-1, m-1)



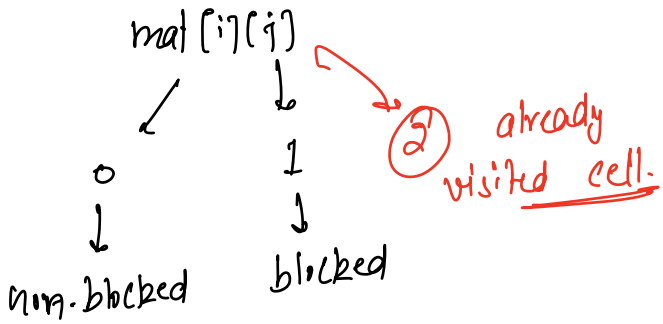
1 → blocker cell.

BF → try out all the possibilities.

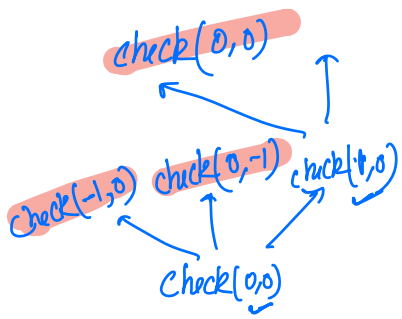


fun(arr, x-1, y)
fun(arr, x, y-1)
fun(arr, x+1, y)
fun(arr, x, y+1)

don't visit the already visited cells.

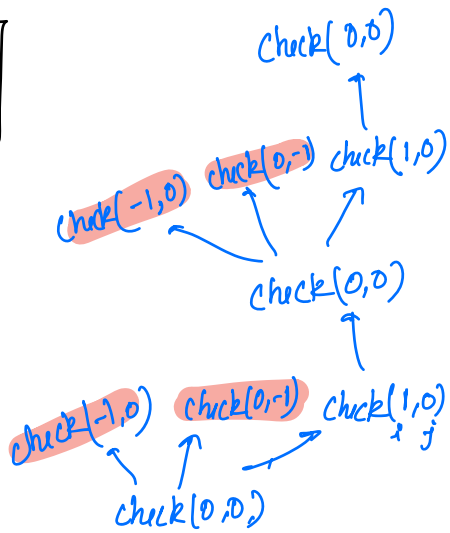


```
boolean check( i, j, mat[n][m], n, m) {
    if ( i == n-1 && j == m-1) {return true}
    if( i<0 || j<0 || i ≥n || j ≥m ) {return false}
    if ( mat[i][j] == 1 || mat[i][j] ==2) { return false }

        mat[i][j] = 2 :
return      check( i-1, j , mat, n, m) ||
            check( i, j-1, mat, n, m) ||
            check( i+1, j, mat, n, m) ||
            check( i, j+1 , mat, n, m)

}
```



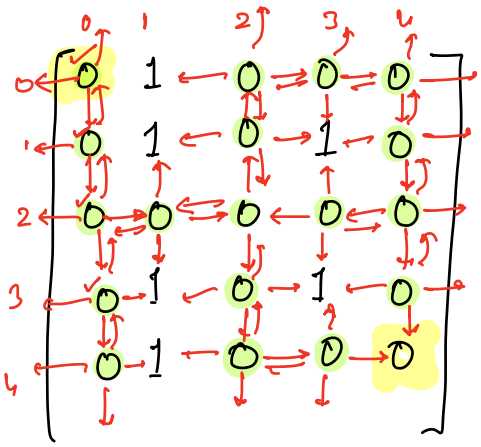
$$\boxed{\begin{array}{l} T.C \rightarrow O(N*M) \\ S.C \rightarrow O(N*M) \end{array}}$$

↑ recursive stack



For all the paths from src to destⁿ ⇒ undo the change.

For a single path from src to destⁿ ⇒ undo is not required.



1 → up
2 → left
3 → down
4 → right

N Queens.

chessboard → N * N

N Queens. → such that no queen is killing another queen.

N=4.

	0	1	2	3
0		Q		
1				Q
2	Q			
3			Q	

⇓
Yes.

N=5.

	0	1	2	3	4
0	Q				
1			Q		
2					Q
3		Q			
4				Q	

Yes.

multiple ans are also possible

N=4.

	0	1	2	3
0			Q	
1	Q			
2				Q
3		Q		

N=3.

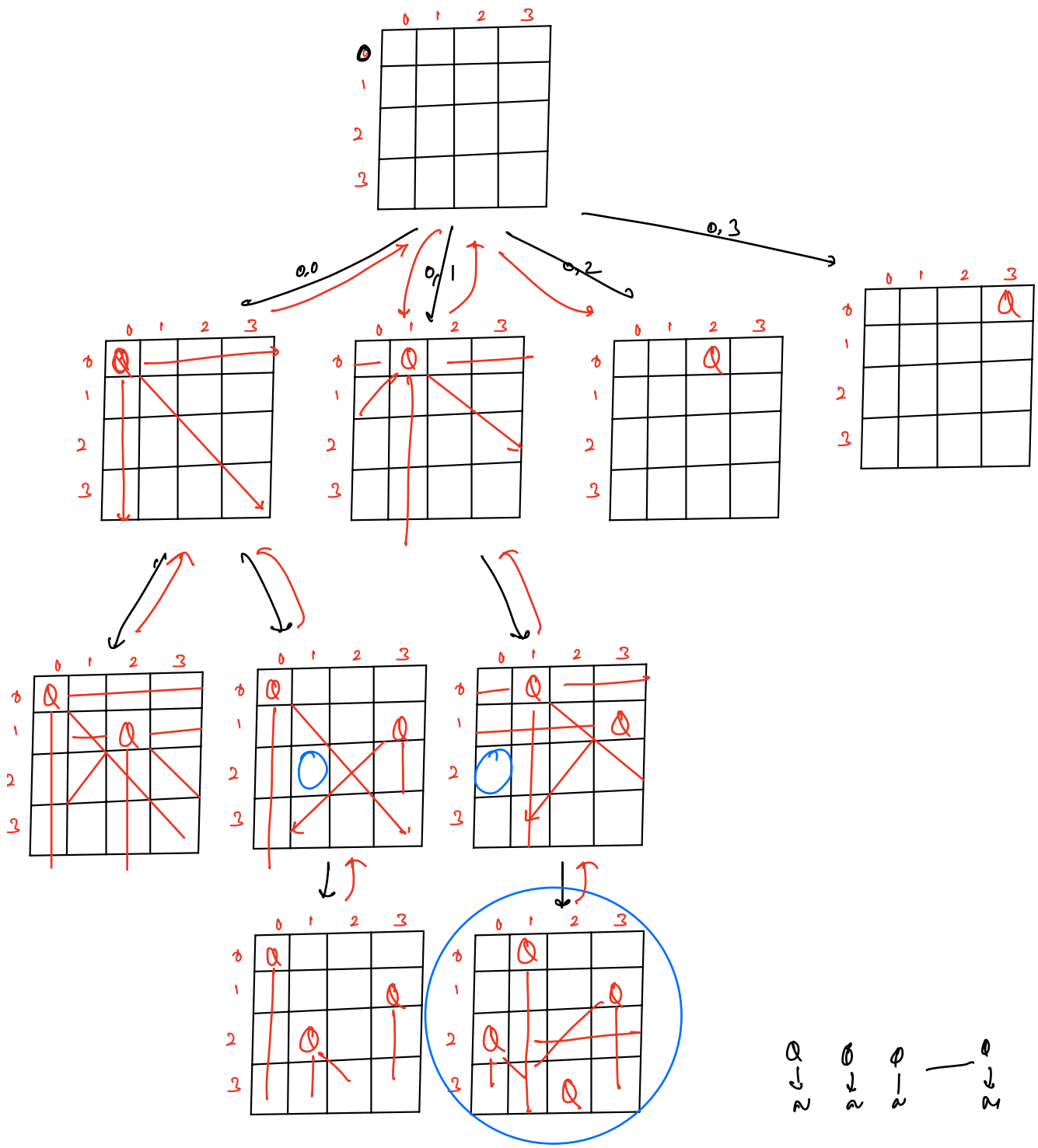
Q		
		Q

⇓
No.

obs → We can't have more than 1 queen in a row.

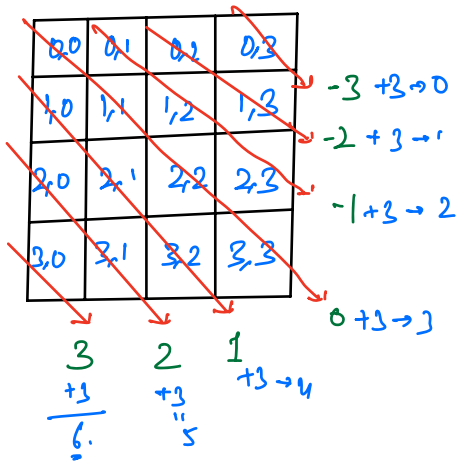
	0	1	2	3
0				
1				
2				
3				

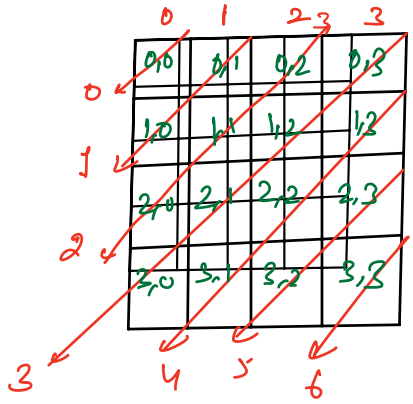
For Every queen, we have N columns available as options.



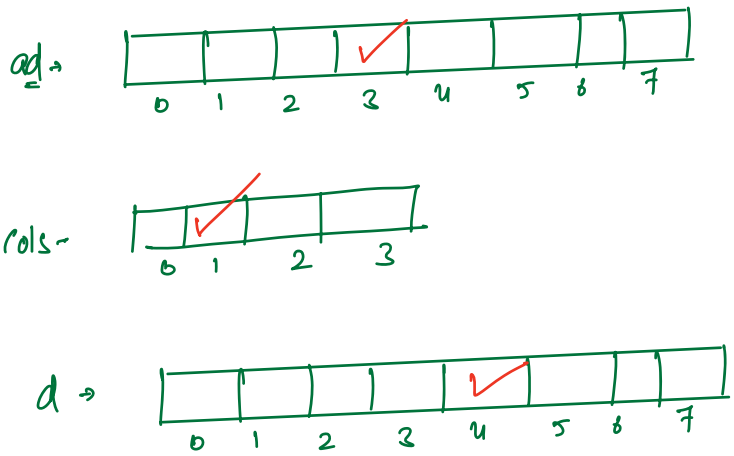
$i, j \rightarrow$ col[j] = true;

$(i-j)$


0,0 0,1 0,2 0,3
1,0 1,1 1,2 1,3
2,0 2,1 2,2 2,3
3,0 3,1 3,2 3,3
-3 +3 → 0
-2 + 3 → 1
-1 + 3 → 2
0 + 3 → 3
3 2 1
+3 +3 +3 → 4
6. 5


0 1 2 3 3
0,0 0,1 0,2 0,3
1,0 1,1 1,2 1,3
2,0 2,1 2,2 2,3
3,0 3,1 3,2 3,3
0 1 2 3
4 5 6
(i+j)

$i-j+(N-1)$


ad →
0 1 2 3 4 5 6 7
cols →
0 1 2 3
d →
0 1 2 3 4 5 6 7

```
NQueens( i , mat[][], cols[], anti-diag[], diag[]){
    if(row == N) { //print the current config. of matrix, return }
        for( j = 0 ; j < N ; j++){
            if( cols[j] == true || anti-diag[i+j] == true ||
                                   diag[i-j+N-1] == true ){
                continue ;
            }
            mat[i][j] = 1   //put a queen at i,j
              cols[j] = true
              anti-diag[i+j] = true
              diag[i-j+N-1] = true
            NQueens( i+1, mat, cols, antidiag, diag)
            mat[i][j] = 0
              cols[j] = false
              anti-diag[i+j] = false
              diag[i-j+N-1] = false
        }
}
```

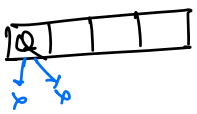
N^N

0th row → N

1st row → N-2

2nd row → N-4

i



T.C → O(N!)

Sudoku.

	0	1	2	3	4	5	6	7	8
0	5	3			7				
1	6			1	9	5			
2		9	8					6	
3	8				6				3
4	4			8		2			1
5	7				3				6
6		6					2	8	
7				4	1	9			5
8					8			7	9

Solve Sudoku

isvalid.

1. row [1-9]
2. col [1-9]
3. 3*3 grid [1-9]

No duplicacy.