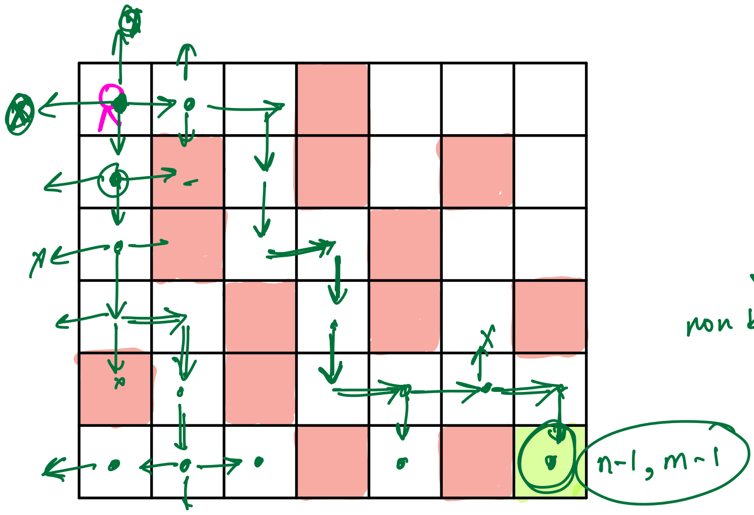
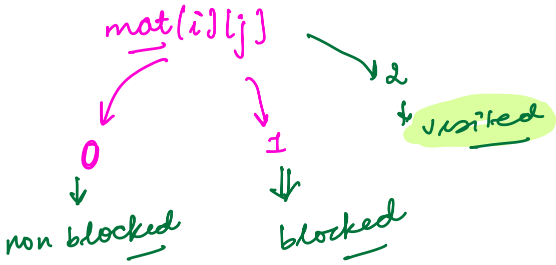
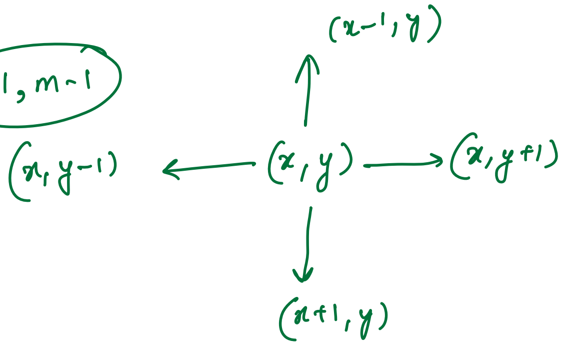
- Rat in a maze





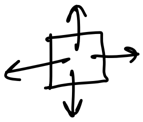


don't visit an already visited cell

parameter

i, j, mat, n, m

target (n-1, m-1)

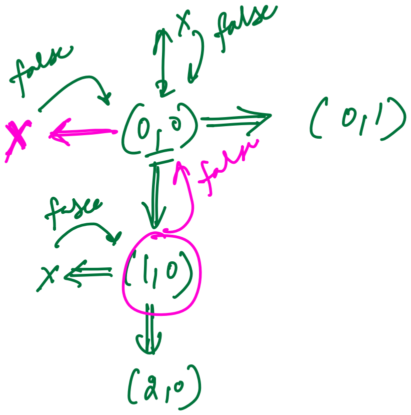


```
bool check ( i, j, mat, n, m)
{
        if( i==n-1 && j==m-1) return true;
        //out of boundary
        if( i<0 || i>=n || j<0 || j>=m)
                                        return false;
        if( mat[i][j]==1 || mat[i][j]==2) return false;

        mat[i][j]=2;

        return check( i-1, j, ...) || check( i, j-1) ||
check ( i, j+1 . . ) || check ( i+1, j );

}
```



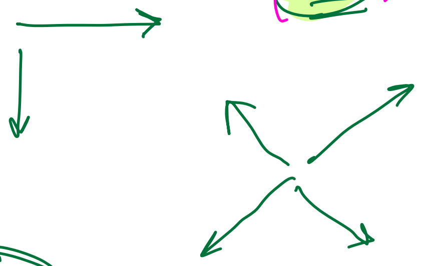
T·C: n*m

S·C: O(n*m)

o

N queens
|
chess

queen



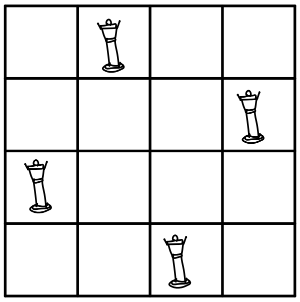
O(n!)

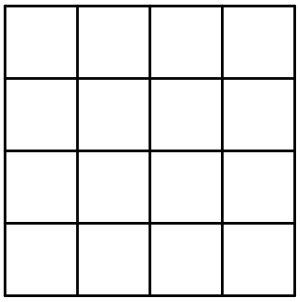
N*N

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

N queens

No queen should be reachable by other → single movement



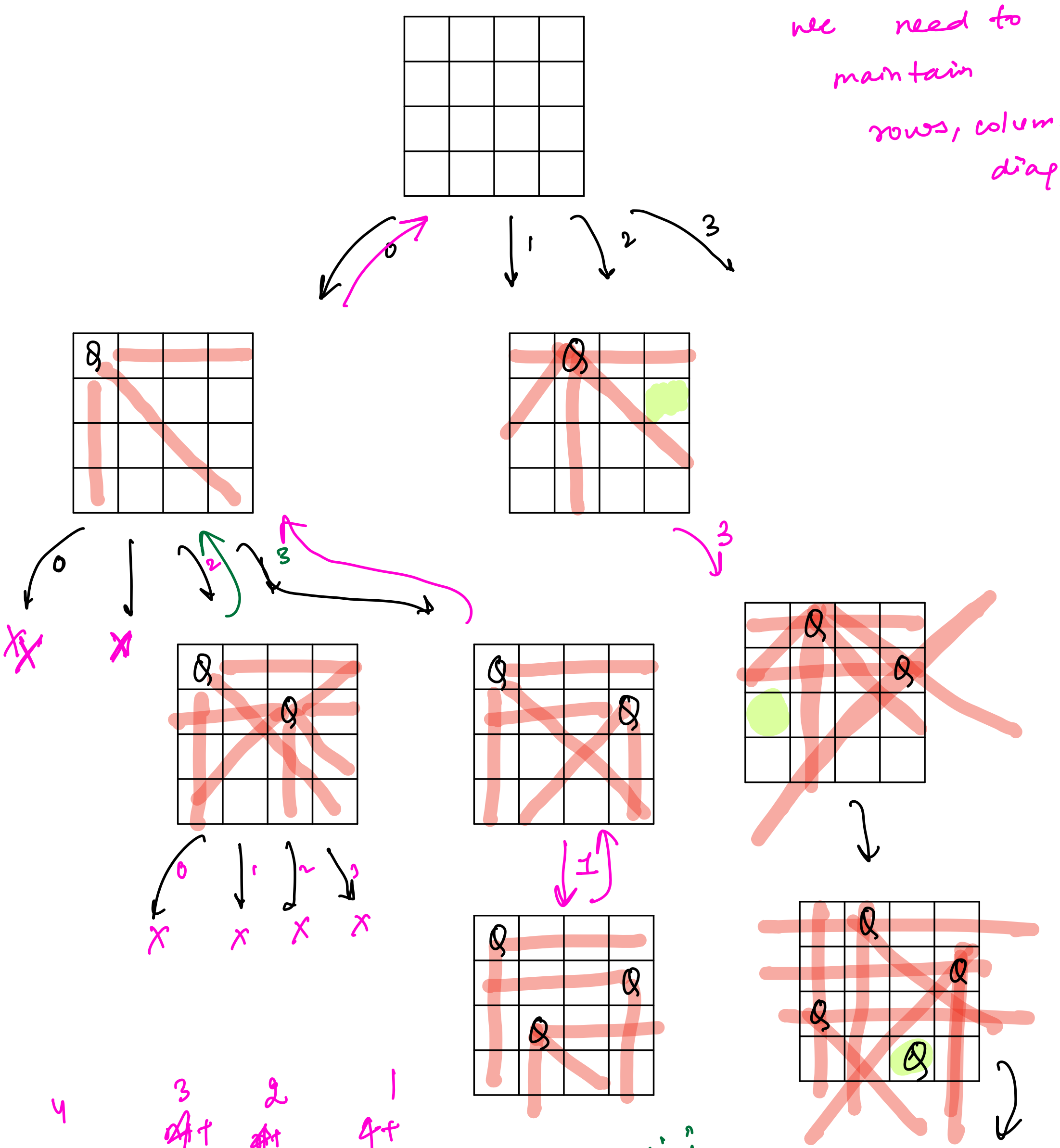


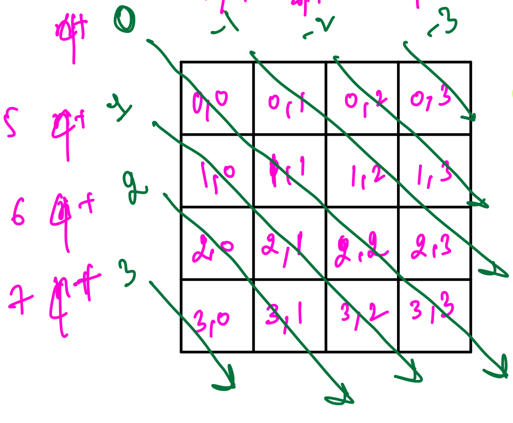
N queen

Every row will contain one queen

0	1	2	3
col 1	col 2	col 3	col 4
⇓	⇓	⇓	⇓
0,1,2,3	0,1,2,3	0,1,2,3	0,1,2,3

we need to maintain rows, columns, diagonals





i, j

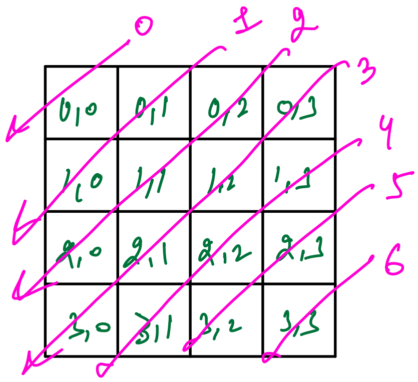
i - j

left diagonal = N + i - j

leftdiagonal[2*n] = {0};

row[n] = {0};
col [n] = {0};
leftdiagonal[ ]

ld[n+i-j] = 1



i+j

right diagonal [2n];

rd[i+j]=1;

parameters

```
Nqueens (index , col[], ld[], rd[], mat[][] , N)
{
        if(index==n) { //print    return; }

        for( j=0; j<=N-1; j++)
        {
              if( col[j]==1 ||
                  ld[N+index-j] ==1 ||
                  rd[indx+j]==1)
                   continue;
             mat[index][j]=1;
              col[j]=1;
              ld[ N+index-j]=1
              rd[ index +j]=1
               Nqueeen( index+1 , ....);
            mat[index][j]=0;
             col[j]=0;
             ld[ N+index-j]=0
             rd[ index +j]=0
        }
}
```

ij

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

sudoku

9*9

1-9

no no in col / row should repeat twice

solve!

mat[ ][ ]

index

r = index/9
c = index%9

sr = r - r%3
sc = c - c%3

```
sudoku( int index, int mat[][] )
{
        if( index==81) { // got your ans return};
          int r= index / 9;
              c= index % 9;
         if ( mat[r][c] != 0) { sudoku( index+1, mat); }
                                          return
         for( int x=1; x<=9; x++)
          {
                  if( x  can be placed)
                   {     mat[r][c]=x;
```

row r if it has x

col c if has x

cube → r-r%3, c-c%3

```
sudoku( index+1, mat ));
mat[r][c] = 0;
        }
    }
```

T.C: $O\left(9^{\underline{n*n}}\right)$ 9^{81} $\underline{n=9}$