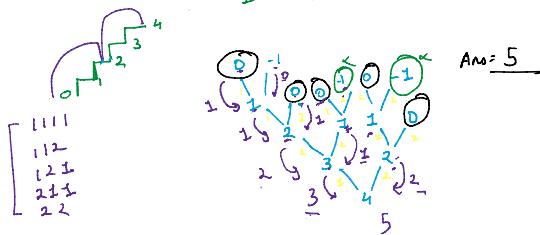


Climbing Stairs :- [https://leetcode.com/problems/climbing_stairs/](https://leetcode.com/problems/climbing-stairs/)

$n = \text{No. of Stairs} = 4$
1 or 2 steps



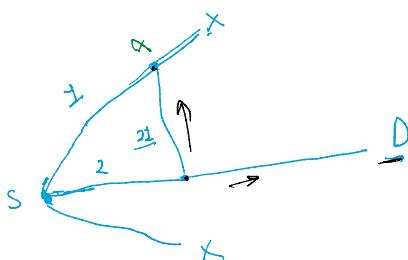
```
int CS(n) {
    if (n < 0) return 0;
    if (n == 0) return 1;
    oneStep = CS(n-1);
    twoStep = CS(n-2);
    return oneStep + twoStep;
}
```

n	oneStep	twoStep
0	0	0
1	1	0
2	2	1
3	3	2
4	5	3
5	8	5
6	13	8
7	21	13
8	34	21
9	55	34
10	89	55
11	144	89

$n = 4$

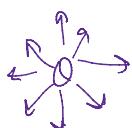
```
public static int climbingStairs(int n) {
    1. if (n == 0) return 1;
    2. if (n < 0) return 0;
    3. int oneStep = climbingStairs(n - 1);
    4. int twoSteps = climbingStairs(n - 2);
    5. return oneStep + twoSteps;
}
```

Backtracking



N-Queens

n -steps



1, 1

+ (-1, 0)

0, 1

(-1, -1)

0, 1

0, 2 (-1, 1)

(-1, 0)

0, 1

0, 2

(-1, 1)

0, 1

0, 2

$n=4$	1	2	3
D	Q	-	-
1	-	Q	-
2	-	-	Q
3	-	Q	-

$n \times n$

(0, 1)
1, 0
2, 0
(1, -1)
2, 1
(1, 0)

1, 2 (0, 1)
2, 2 (1, 1)

0 1
0 1
0 1 1 1

3	Q	Q	
2			

$$dir = [(-1, -1), (-1, 0), (-1, 1)]$$

0	1	2	3
0	Q		
1			Q
2			
3			

$$\begin{aligned} nr &= 2 \times 0 \\ nc &= 2 \times 2 \\ dirR &= -1 \\ dirL &= 0 \\ i &= 0 \end{aligned}$$

```
public static boolean canBePlaced(char[][] chess, int r, int c) {
    int[][] dir = {{-1, -1}, {-1, 0}, {-1, 1}};
    for (int i = 0; i < dir.length; i++) {
        int nr = r;
        int nc = c;
        int dirR = dir[i][0];
        int dirC = dir[i][1];
        while (nr >= 0 && nc >= 0 && nc < chess.length) {
            if (chess[nr][nc] == 'Q') return false;
            nr += dirR;
            nc += dirC;
        }
    }
    return true;
}
```

n Queen (chess, n, c,
if (r == queens) solFour
y (c == chess.length) re

y (isNotPlaced (chess
array[r][c] != 'Q'))

n Queen (chess,
chess[r][c] = '.')

}

n Queen (chess, R,
?)

Rat in A Maze :- <https://practice.geeksforgeeks.org/problems/rat-in-a-maze-problem/1>

0	1	2	3
1	0, 0, 0, 0},		
2	{0, 1, 0, 1},		
3	{1, 1, 0, 0},		
	{0, 1, 1, 1}}		

| = trace through it

0 = It is blocked

[D D R D R R
D R D D R R

→ 2 | , n

```
2 2  
ess, int r, int c) {  
};
```

```
ss.length) {  
false;
```

- queens)
d();

return :

, r, c)) {

; r+1, 0, queen+1),

;

(+ 1, queens);

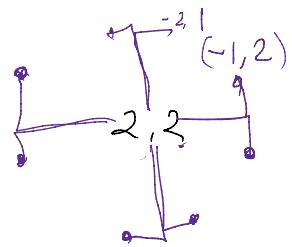
3

{0, 1, 1, 1}

 $\begin{array}{c} \text{L} \\ \text{D} \\ \text{R} \end{array} \begin{array}{c} \text{U} \\ \text{V} \\ \text{W} \end{array} \begin{array}{c} \text{R} \\ \text{D} \\ \text{O} \end{array} \begin{array}{c} \text{U} \\ \text{D} \\ \text{O} \end{array} \begin{array}{c} \text{L} \\ \text{R} \\ \text{R} \end{array}$
Knights - Tour :-

$n \times n$

0	1	2	3	4
0	1		3	
1	4			
2		2		
3			5	
4				



=> Place the Knight

=> Iterate over all directions
 ↳ If you can place the knight
 ↳ recursion

=> Reset the Knight

