DP Concepts & Questions

Video 31

भाषण

(Motivation)

" You have two options when 2025 starts :-

① Work Hard → Chase your dream → Enjoy

② Relax and be lazy .

The decision you take can change your life. Think wisely . "

MIK...

cswithMIK → Twitter

Facebook
Instagram → codestorywithMIK

whatsapp → codestorywithMIK

Done

- 1-D based DP ✓

✓

- Grid based DP

Done

- String based DP ✓

- Digit DP

- Game Strategy

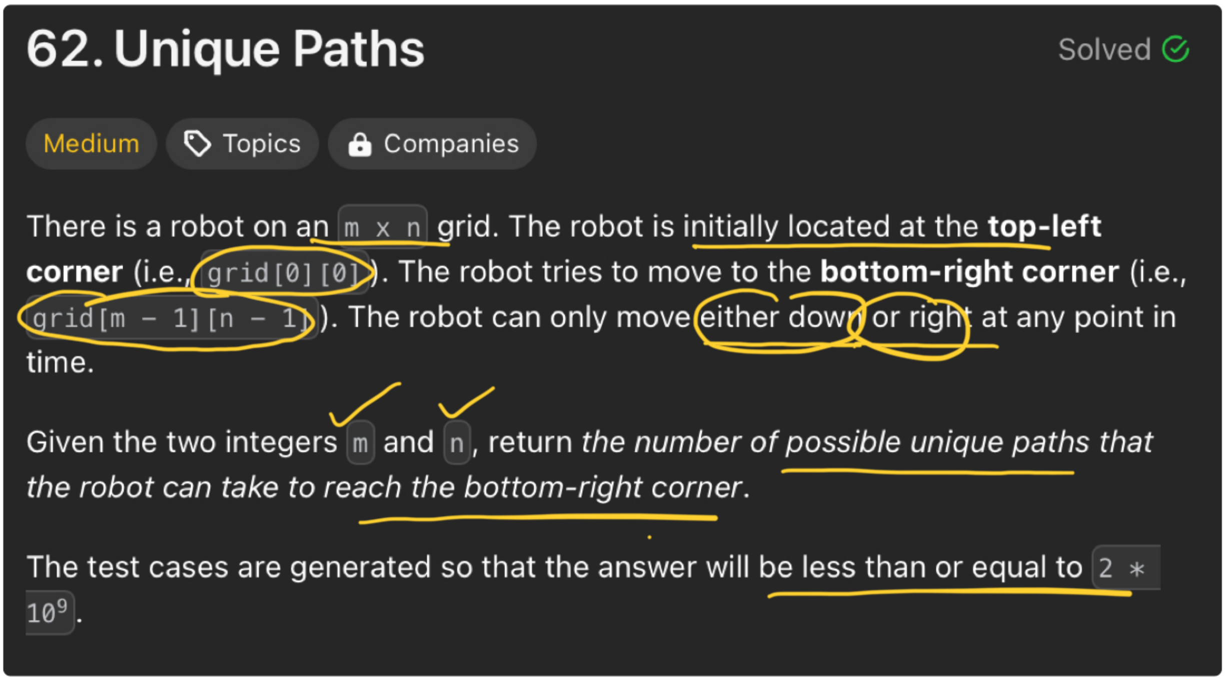
We'll do :-

(•) RECURSION
+
MEMOIZATION
(Top Down)

(•) Bottom UP

(•) Time & Space

DP on Grids



62. Unique Paths

Solved ✓

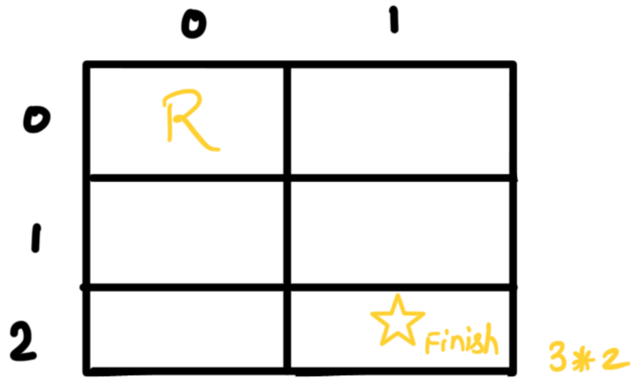
Medium | Topics | Companies

There is a robot on an `m x n` grid. The robot is initially located at the **top-left corner** (i.e., `grid[0][0]`). The robot tries to move to the **bottom-right corner** (i.e., `grid[m - 1][n - 1]`). The robot can only move either down or right at any point in time.

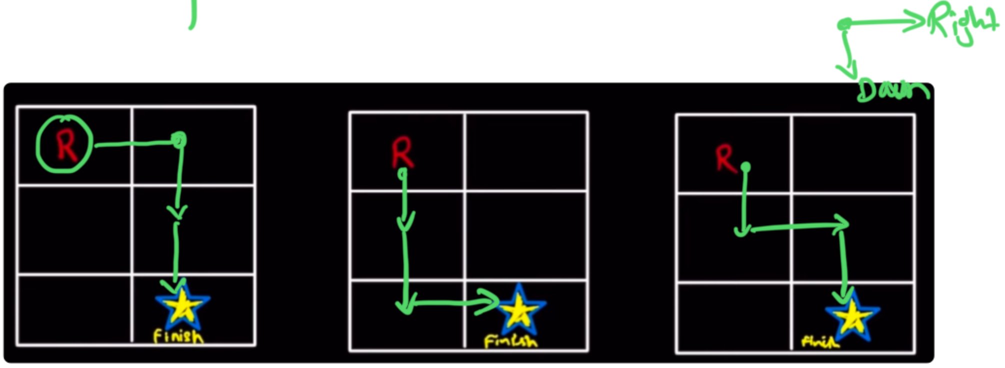
Given the two integers `m` ✓ and `n` ✓, return *the number of possible unique paths that the robot can take to reach the bottom-right corner*.

The test cases are generated so that the answer will be less than or equal to `2 * 10^9`.

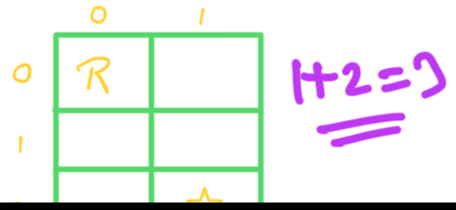
Example:- m = 3, n = 2



Output:- 3



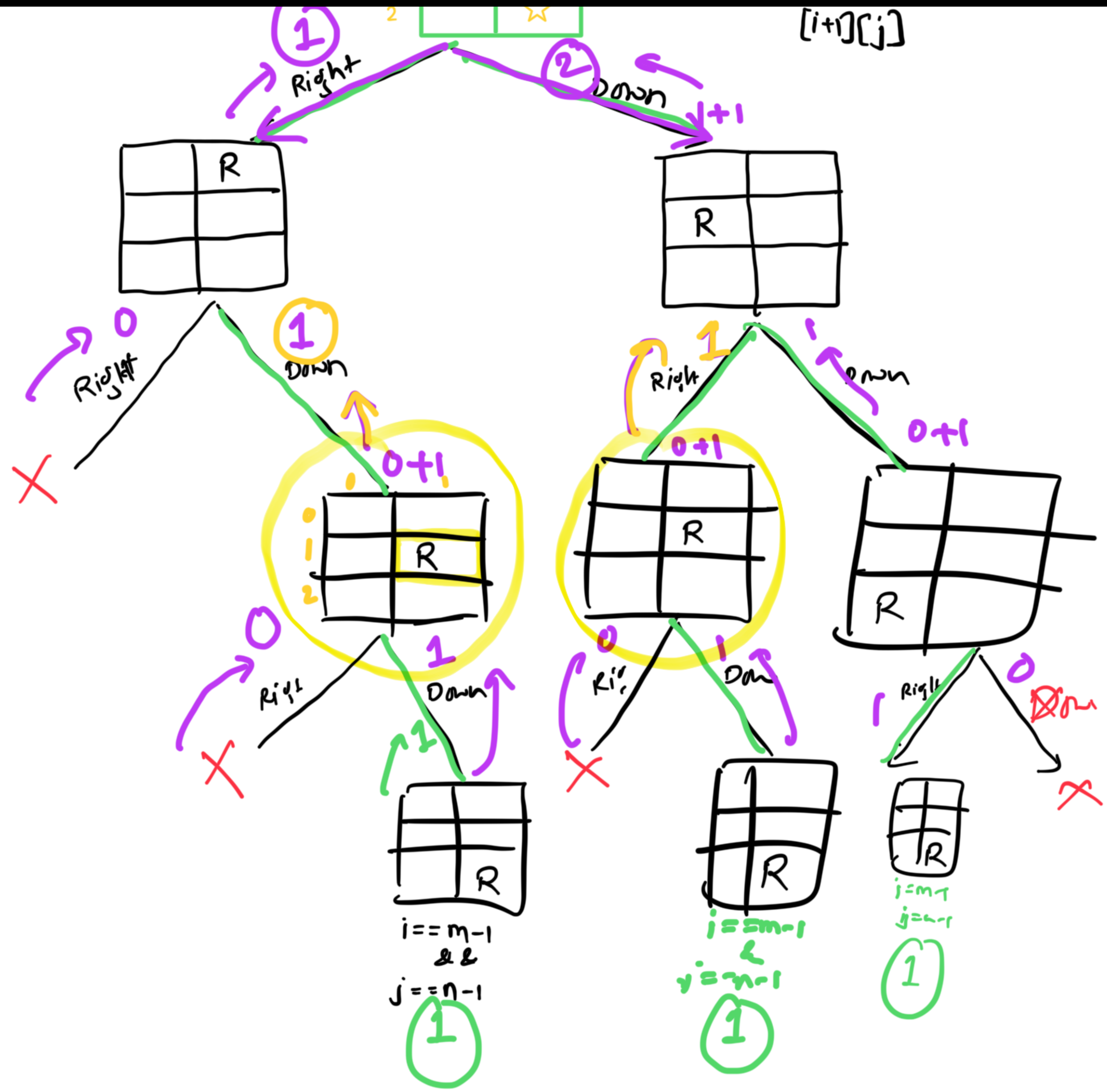
Thought Process...



Robot

[i][j] ⟶ (Right) [i][j+1]

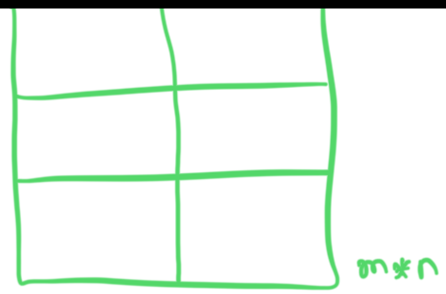
Down ↓

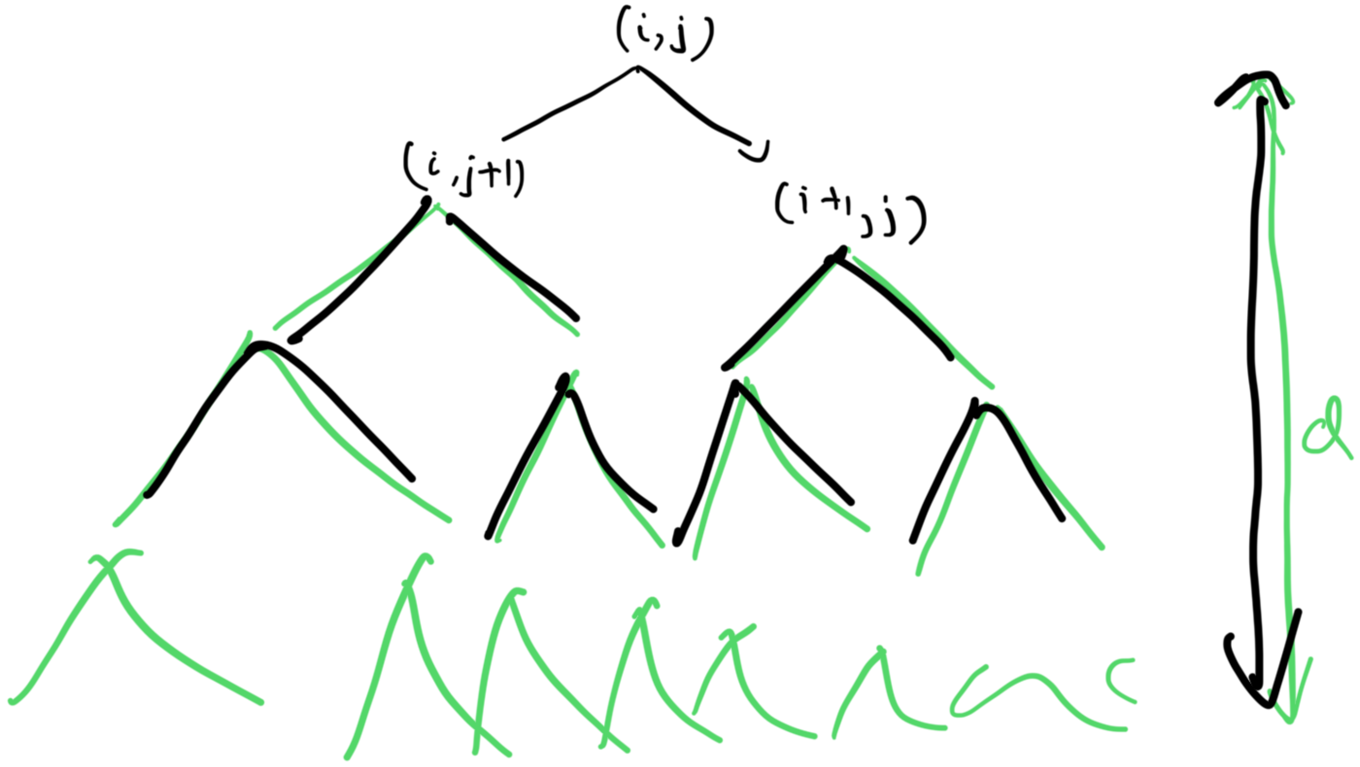


```
Solve ( 0, 0) ;
t [m+1] [n+1] ; = {-1}
int Solve ( int i, int j) {

        if ( i == m-1 && j == n-1) {
            return 1;
        }
        if(i<0 || i>=m || j<0 || j>=n){
            return 0;
        }
        if(t[i][j] != -1) return t[i][j];
        int right = solve(i, j+1);
        int down = solve(i+1, j);

    return t[i][j] = right + down ;  //Memoization.

    }
}
```

Time Complexity :-

without Memoization :-



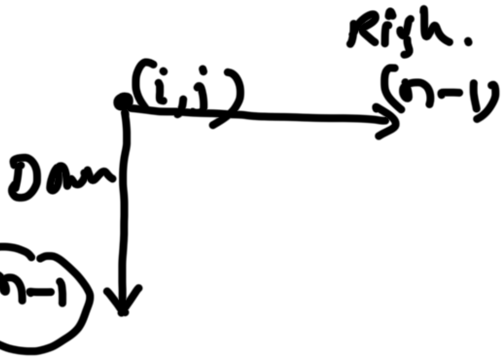


2^d

$\simeq 2^d$

$O\sim(2^{m+n})$

Exponential.



System Stack space:-

$(m-1)+(n-1)$

space:: $O(m+n-2)$ $(m+n\cancel{-2})$

Memoization:.

$t[m+1]\ [n+1]$

$T \cdot C = O\big((m+1) * (n+1)\big) \simeq O(m*n)$

$S \cdot C = O(m*n)$

Bottom Up :-

$t[m+1][n+1]$

// $t[i][j] = x$ ⟹ State definition ⇐

# of ways to reach [i][j] from [0][0] ⇐

return f[m-1][n-1] ;

m=3

n=2

	0	1
0	0	1
1	1	?
2	1	?

f

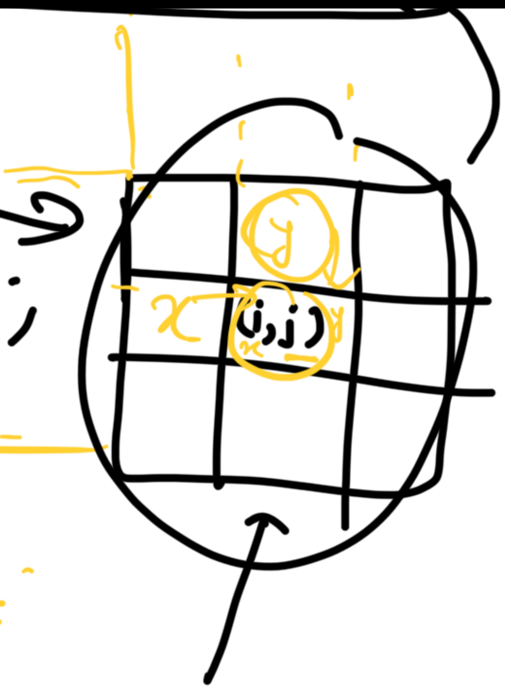
f[0][0] = # ways to reach [0][0] from [0][0]

f[0][1] = # ways to reach [0][1] from [0][0]
= 1

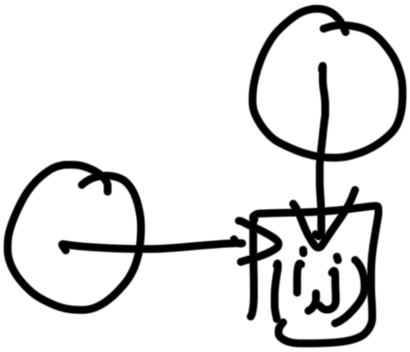
f[1][0] = # ways to reach [1][0] from [0][0]
= 1

```
f[0][0] = 0 ;
// Fill 0th Row with 1

// Fill 0th Col with 1

for (i = 1; i<m; i++) {
        for(j=1; j<n; j++) {
                f[i][j] = f[i-1][j] +
                          f[i][j-1];
}
```

return t[m-1][n-1];



$x+y$



Movements.

$$t[i][j] = x + y$$

$$T.C = O(m*n)$$

$$S.C = O(m*n).$$