BACKTRACK 2

Good Evening

Starting at 9:06 pm

* Content

01. Print paths in Staircase
02. Print all paths from src to dest
03. shortest path in binary maze

* Try → N Queen problem
Sudoku
Flood fill
Knight Tour problem

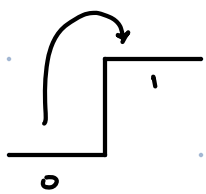
} DSA 4.2

* **Print all paths in Staircase**

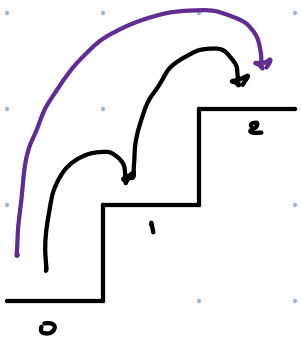
You are climbing a staircase & it takes A steps to reach top.

Each time we can go climb 1 step or 2 step.

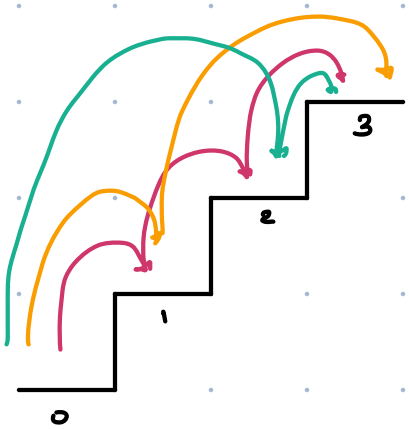
How many distinct ways to climb at the top



Ans = 1

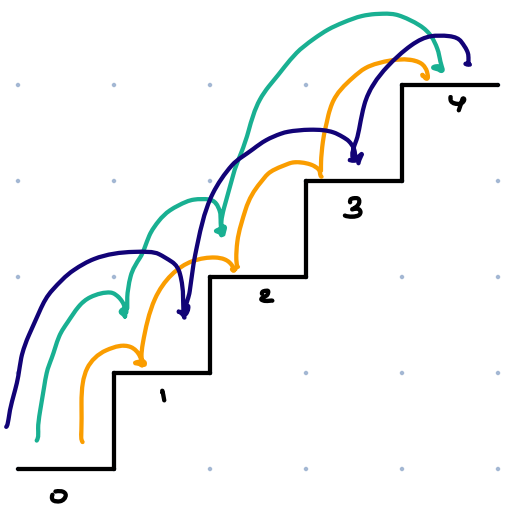


Ans = 1 1
2



Ans

{ 1 1 1
1 2
2 1 }



Ans =

{ 1 1 1 1
1 1 2
1 2 1
2 1 1
2 2 }

Recurrence relation

$\Downarrow$

generatepaths (A , currpath)

= generatepaths (A-1 , currpath + 1) &

generatepaths (A-2 , currpath + 2)

path(4) = path(3)

1 1 1 1
1 2 1
2 1 1

= path(2)

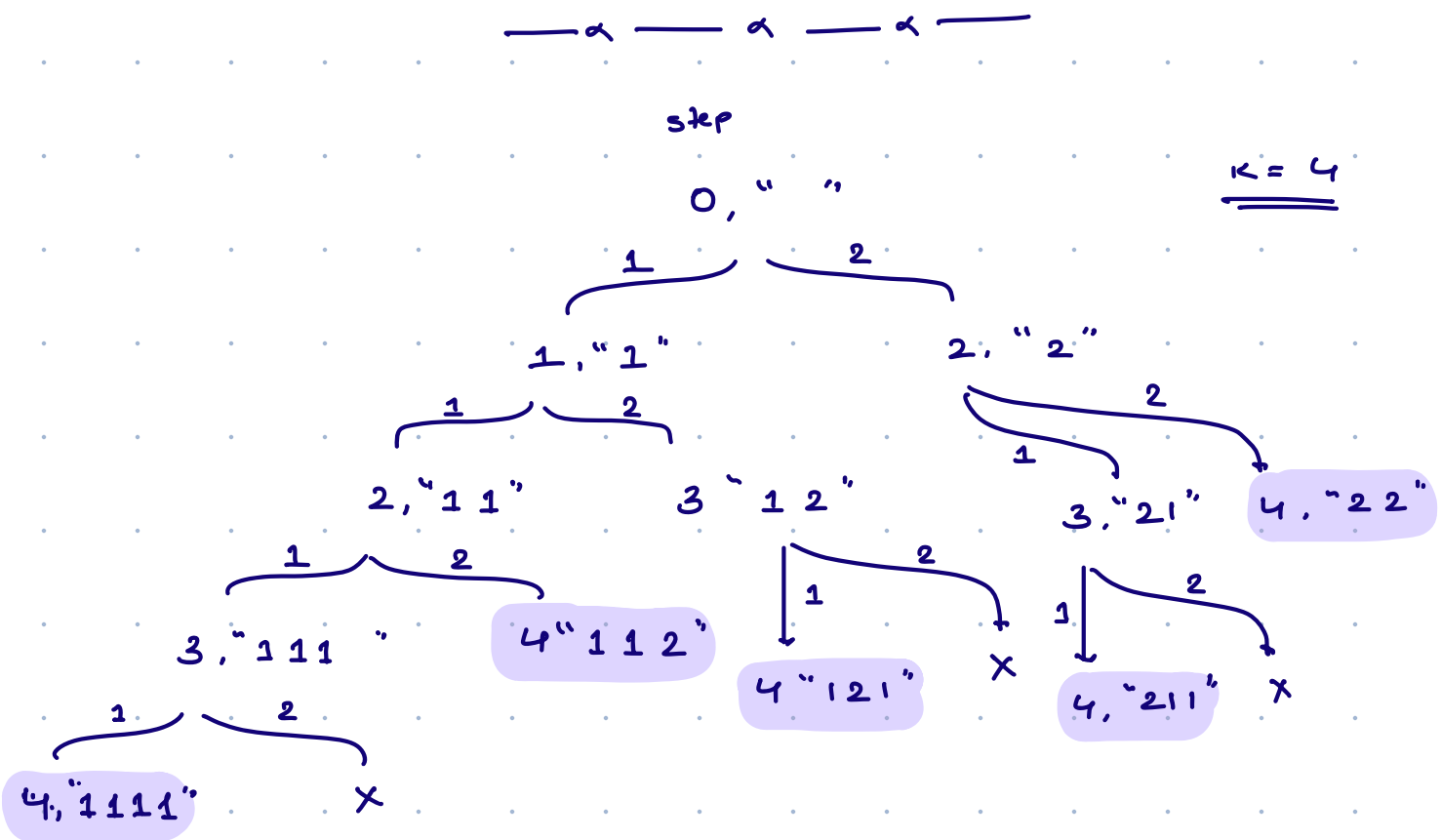
1 1 2
2 2

```
function generatepath ( A , AL<I> curr , AL<I> ans ){
    if ( A==0){
        ans.add( new AL<>(curr));
        return ;
    }
    if (A>=1){
        curr.add( 1 )
        generatepath (A-1 , curr, ans);
        curr.remove ( curr.size()-1);
    }
```

TC : O(2^n)

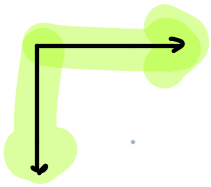
SC : O(n)

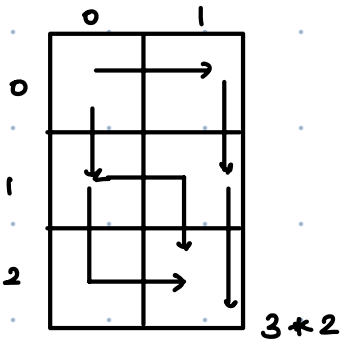
```
        if (n >= 2)
            curr.add(2)
            generatepath (n-2, curr, ans);
            curr.remove(curr.size()-1);
    }
}
```



* Print Maze paths

We will be given src & destination. Print all the paths from src to destination.





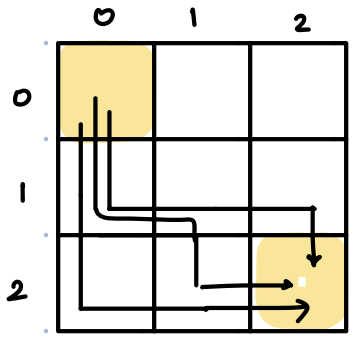
src = (0,0)

dest (2,1)

D D R

D R D

R D D



D D R R ✓

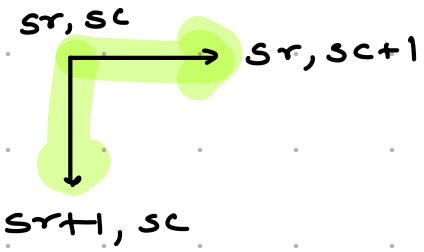
D R R D ✓

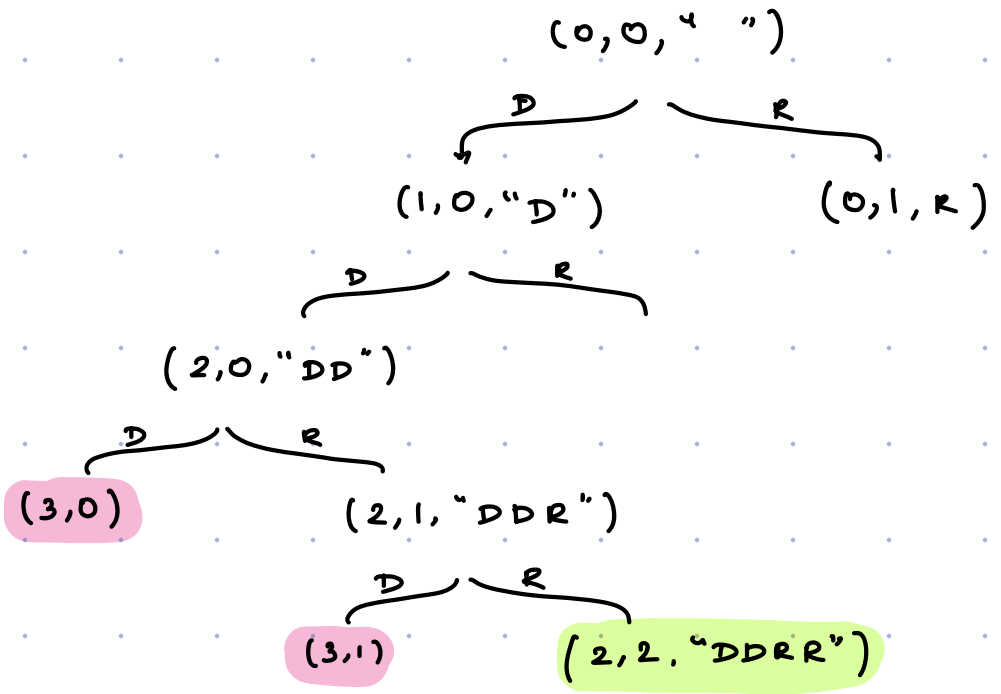
D R D R ✓

R R D D

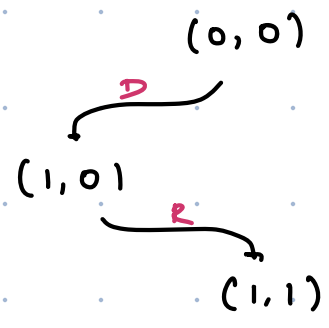
R D R D

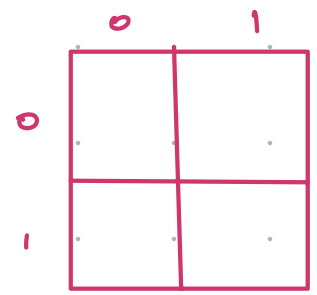
R D D R






```
void printmaze ( sr, sc, dr, dc, String ans ){

    if ( sr == dr && sc == dc ){
        print (ans);
        return;
    }

    if ( sr+1 <= dr ){
        printmaze ( sr+1, sc, dr, dc , ans + "D" );
    }

    if ( sc < dc ){
        printmaze ( sr, sc+1, dr, dc, ans + "R" );
    }

}
```





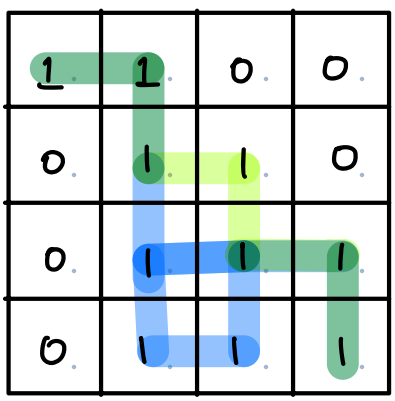
* Shortest path in Binary maze with Hurdles

Problem Statement

Given an **MxN** matrix where each element can either be **0** or **1**. We need to find the length of **shortest path** between a given **source** cell to a **destination** cell.

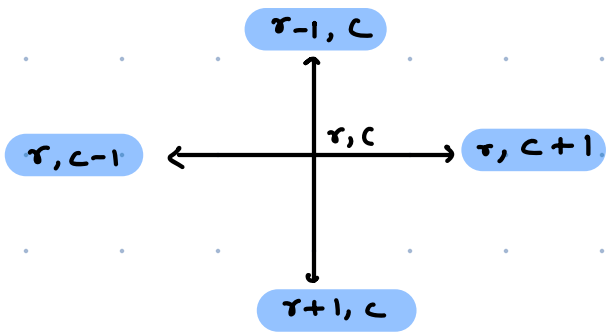
A cell with value **0** denotes that it's a hurdle. The path can only be created out of the cells with values **1**.

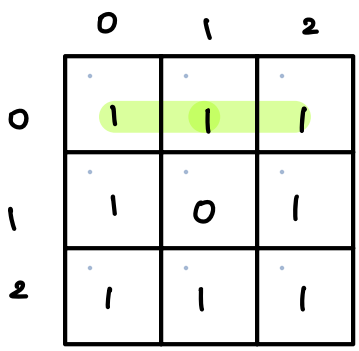
If **NO** path exists then print -1.



Ans = 6

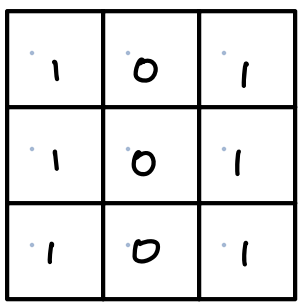
(0,0) to (3,3)





Ans = 2

(0,0) to (0,2)



Ans = -1

Not reachable

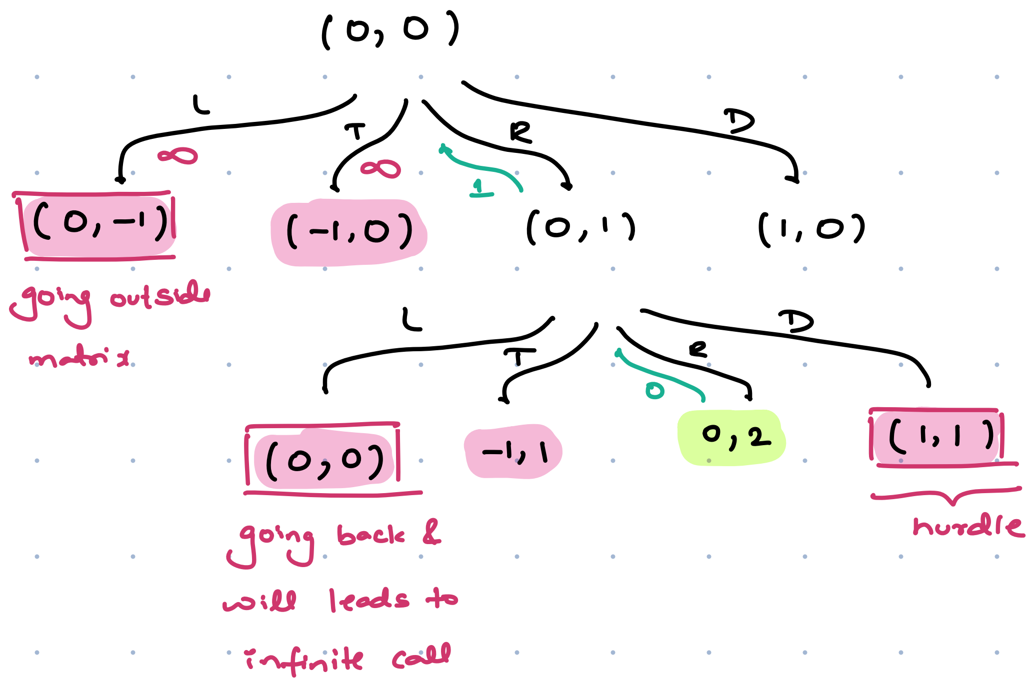
(0,0) to (0,2)

	0	1	2
0	1	1	1
1	1	0	1
2	1	1	1

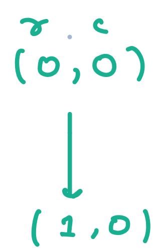
(0,0) to (0,2)

	0	1	2
0	T	F	F
1	F	F	F
2	F	F	F

(0,0) to (0,2)



	r	c	
dir[ ][ ] = 0	{ 0,	−1 }	L
1	{ −1,	0 }	T
2	{ 0,	1 }	R
③	{ 1,	0 }	D



d=3

```
for ( d=0 ; d<4 ; d++) {
    int nr = r + dir[d][0]    = 0 + 1 = 1
    int nc = c + dir[d][1]    = 0 + 0 = 0
}
```

```
int findshortest ( )[ ]A, sr, sc, dr, dc, vis[ ][ ]) {
    if ( sr==dr && sc==dc) return 0;
    vis[sr][sc] = true;
    min = ∞
    for ( d=0 ; d<4 ; d++) {
        int nr = sr + dir[d][0]
        int nc = sc + dir[d][1]
        if (nr≥0 && nr<m && nc≥0 && nc<m &&
            A[nr][nc] == 1 && vis[nr][nc] == false) {
            min = Math.min (min, findshortest (A, nr, nc, dr, dc, vis);
        }
    }
    vis[sr][sc] = false;
    return min == ∞ ? min : min+1
}
```

TC = 4^{m*n}

SC : O(m*n)

```
main() {
    int ans = findshortest(A, sr, sc, dr, dc, vis[][]);
    if (ans == ∞) return -1;
    else return ans;
}
```