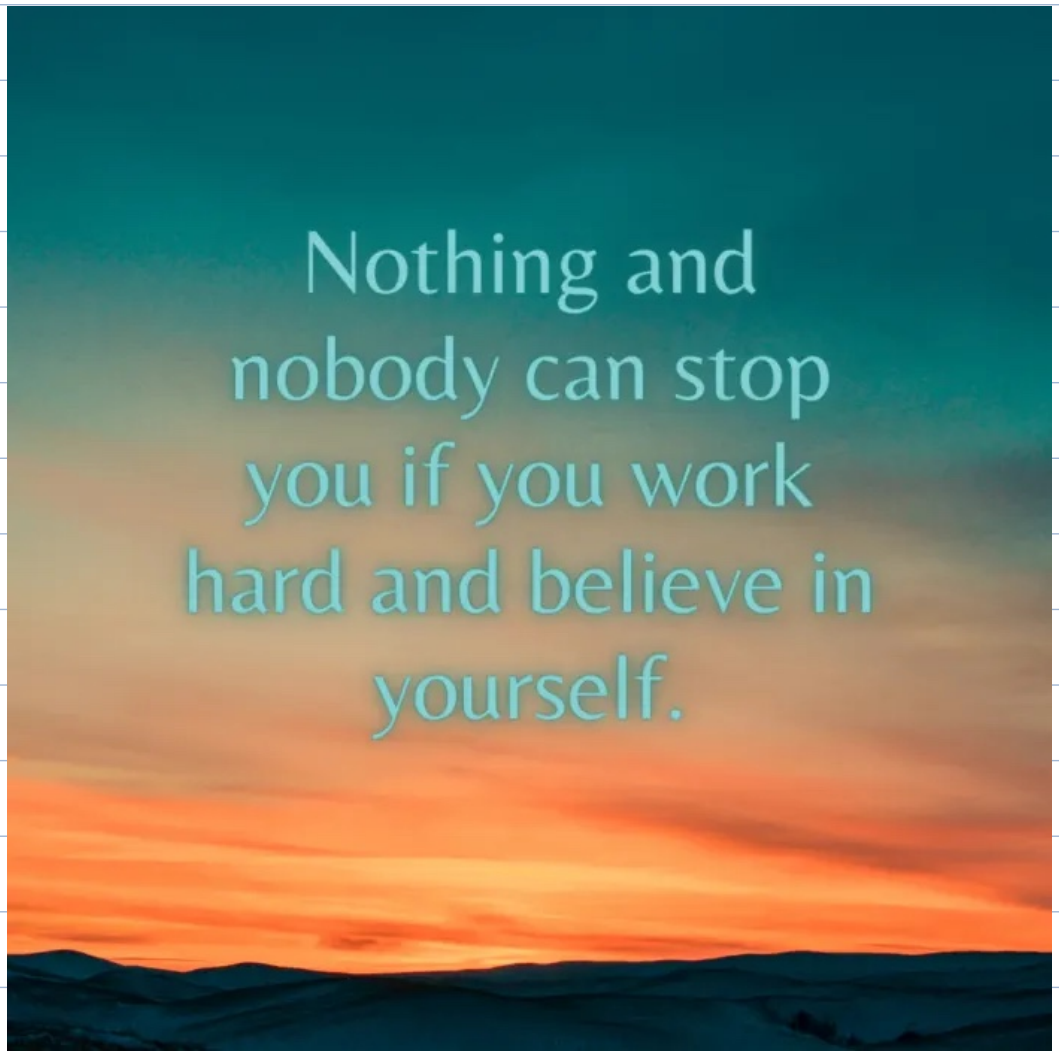
Backtracking 2

AGENDA:

- Print paths in staircase
- Print all paths from source to destination
- Shortest path in a matrix with hurdles



Abishek Ilango
Akash Deep Verma
Amreshwar
Anil
ANUBHAV RAMNANI
Arunava Basak
Debasish Brahma
Gautam Vysyaraju
Harshad Sanjay Marathe
Harshdeep Srivastava
krishnamurthy Donta
M S Haseeb Khan
Mahesh Baswaraj
Mohan Kumar M
Nishant Raj
Pranjul Kesharwani
Prasanna
Priyank Varshney
Rahul
Sahil Urade
Saurabh Dayal
SHIVAM SHIV
Subhash
Surya Shanmughasundaram
Vikas Yadav
Yoshita Rathore

Rules

Q ⟶ QT

A ⟶ PC

100% active

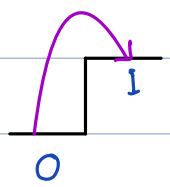
— Print paths in staircase

You are climbing a staircase and it takes A steps to reach top. Each time you either climb 1 or 2 steps

In how many ways can you climb to the top?

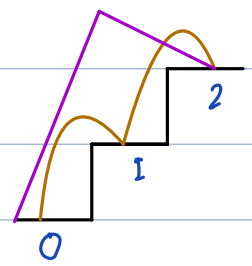
NOTE: you need to return all distinct ways to climb in lexicographical order

A = 1

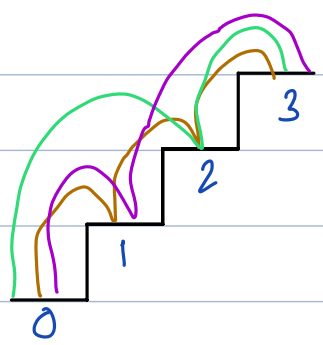


O/p = [[1]]

A = 2

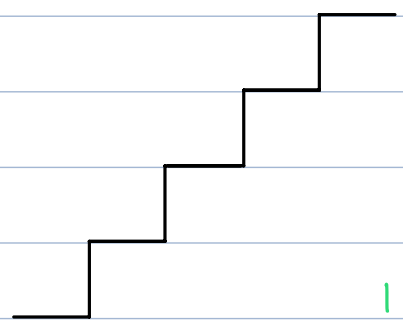


O/p = [[1 1], [2]]



A = 3

O/p = [[1 1 1], [1 2] [2 1]]



A = 4

1 1 1 1

1 1 2

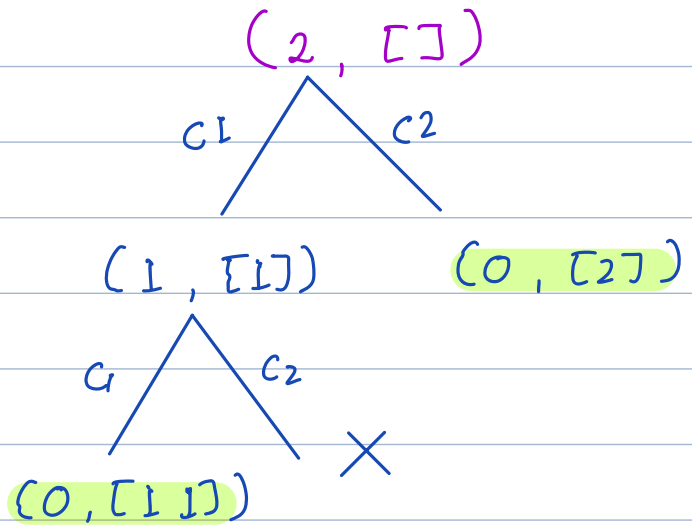
1 2 1

2 1 1

2 2

Idea At every step I can climb either 1 step or 2 steps

A = 2



To print all paths in a Staircase using recursion, what are the recursive states to be used from the state `generatePaths(A, currentPath)`?

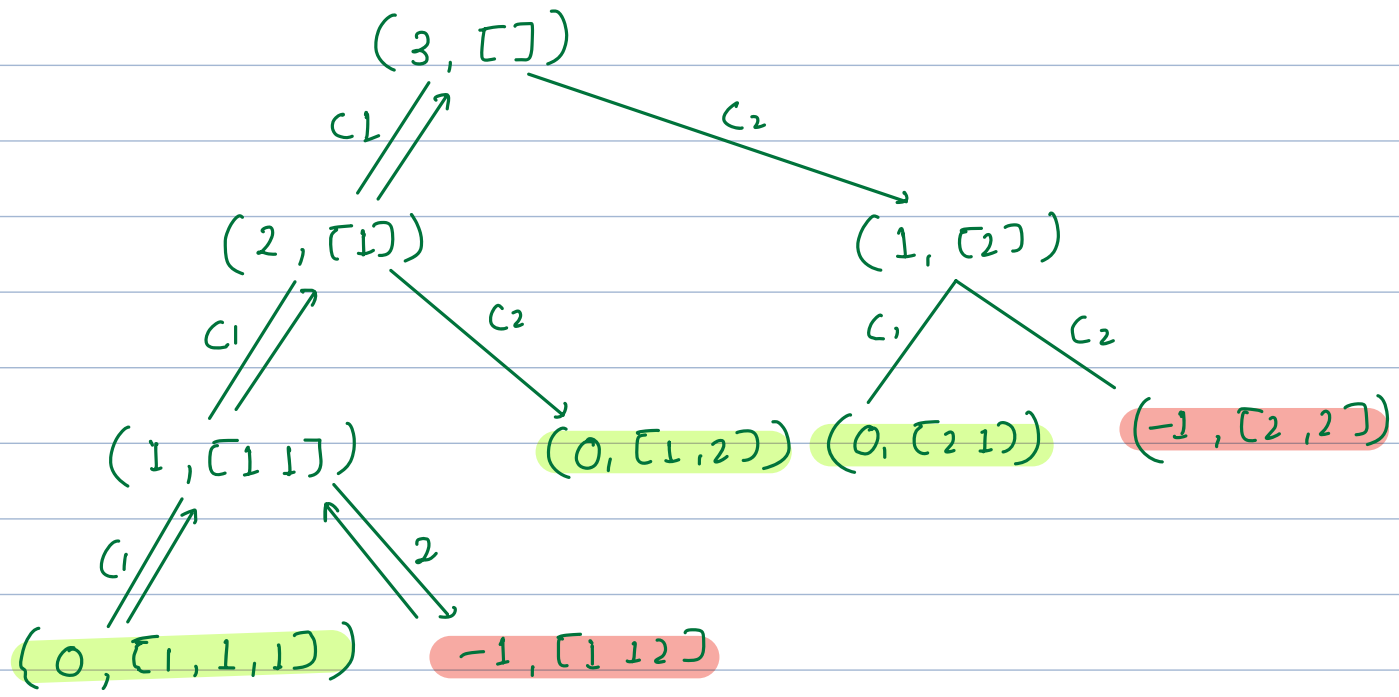
0 users have participated

A	generatePaths(A - 2, currentPath + [1]) and generatePaths(A - 1, currentPath + [2])	0%
B	generatePaths(A - 2, currentPath + [2])	0%
✓ C	generatePaths(A - 2, currentPath + [2]) and generatePaths(A - 1, currentPath + [1])	0%
D	generatePaths(A - 1, currentPath + [1])	0%

Pseudocode

```
ans    // List of List
main (int A) {
    ans = new ArrayList<>()
    AL path = new ArrayList<>()

    generatePath ( A , path )

    return ans
}

void generatePaths ( int A , List<Integer> path ) {
    // Base
    if ( A==0 ) {
        ans.add (path)        always add deep copy
                              ans.add ( new AL<path>)
        return
    }
    if (A<0) return ;
    // climb one step
    path.add (1)                          // Do
    generatePaths ( A-1 , path )          // recurse
    path.remove ( path.size() - 1 )       // undo

    // climb step
    path.add (2)
    generatePaths ( A-2 , path )
    path.remove ( path.size() - 1 )
}
```

$A = 3$



assume $N = A$

TC: $O(n * 2^n)$

SC : $O(N)$

— Print all paths from source to destination

you are given the dimenion of rectangular board A * B
you need to print all the possible paths from TL to BR
top left Bottom right

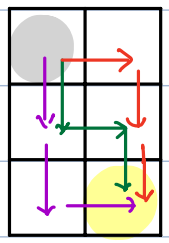
you can only move down or right

R →
D ↓

you need to print the paths in lexicographical order
dictionary order

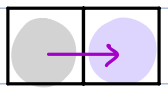
I/P O/P

A = 3 B = 2



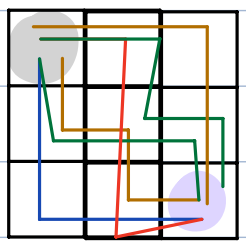
["DDR"
"DRD"
"RDD"]

A = 1 B = 2

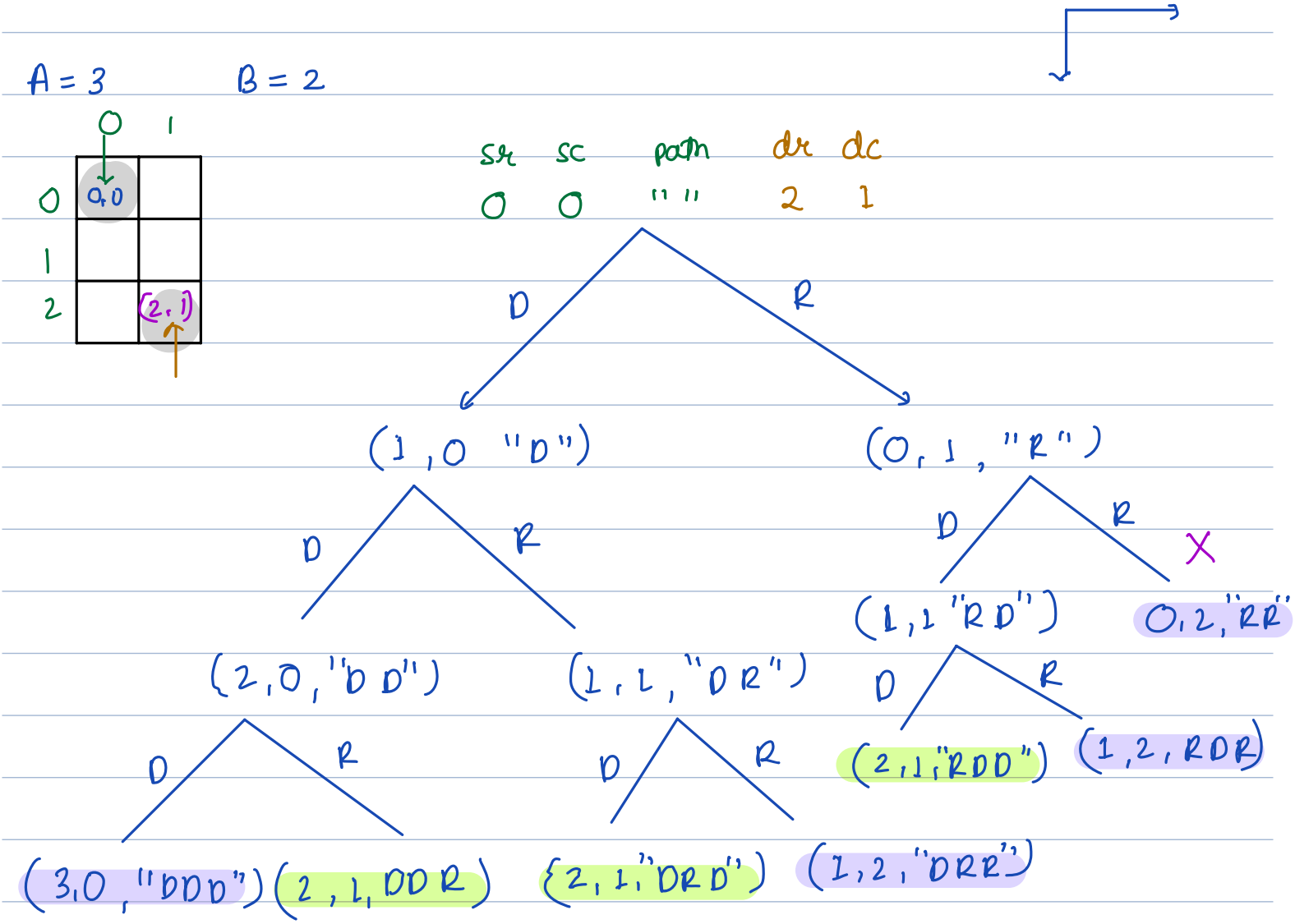


["R"]

A = 3 B = 3



[DDRR
DRDR
DRRD
RDDR
RDRD
RRDD]



To print all paths in lexicographical order the easiest way is to ?

45 users have participated

A	Generate those paths first, which exhaust **right** moves before **down** moves	7%
✓ B	Generate those paths first, which exhaust **down** moves before **right** moves	73%
C	Generate all paths first in any order then, then **sort** them treating as strings	9%
D	Generate paths going down / right in random order	11%

```
ans = list of string   // global
main ( A , B ) {
    ans // re-init

                sr  sc  path   dr   dc
    allPaths ( 0, 0, "", A-1, B-1 )

    ret ans
}

void allPaths ( sr , sc , path , dr , dc ) {
    // Base
    if ( sr == dr && sc == dc ) {
        ans.add (path)
        return
    }

    // Edge cases out of bounds
    if ( sr > dr || sc > dc )  return

    // go down
    allpaths ( sr+1 , sc , path + "D" , dr , dc )

    // go right
    allpaths ( sr , sc+1 , path + "R" , dr , dc )
}
```

TC $O(N * 2^N)$

SC $O(N)$

$N = A + B - 2$

Break : 22 : 35

— Shortest path in a binary maze with hurdles

airbnb
msft
google

Given $R \times C$ matrix where each element is 0 or 1

Find the length of shortest path between a given source cell to a destination cell.

U
L ← → R
D

A cell with value 0 ⟶ hurdle

The path can only be created out of cells with value 1

I/P

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

O/P

6

$sr = 0 \quad sc = 0$

$dr = 3 \quad dc = 3$

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

$sr = 0 \quad sc = 0$

$dr = 0 \quad dc = 2$

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

6

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

2

min

2

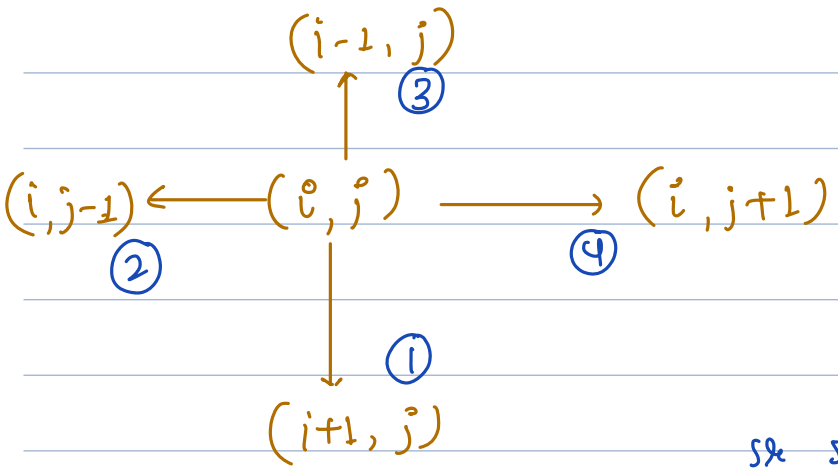
	0	1	2
0	1	0	1
1	1	0	1
2	1	0	1

-1

$\because$ its impossible

$sr = 0 \quad sc = 0$

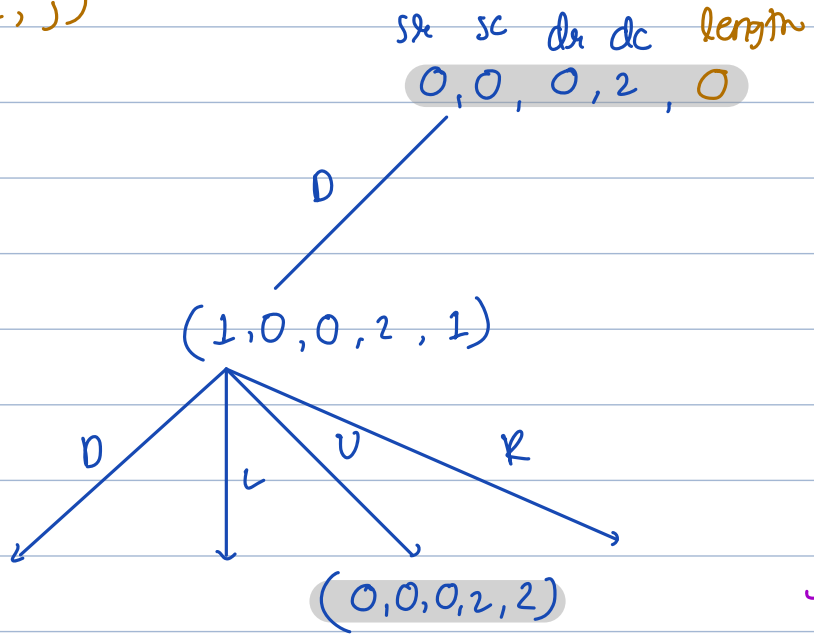
$dr = 0 \quad dc = 2$



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

$sr = 0 \quad sc = 0$

$dr = 0 \quad dc = 2$



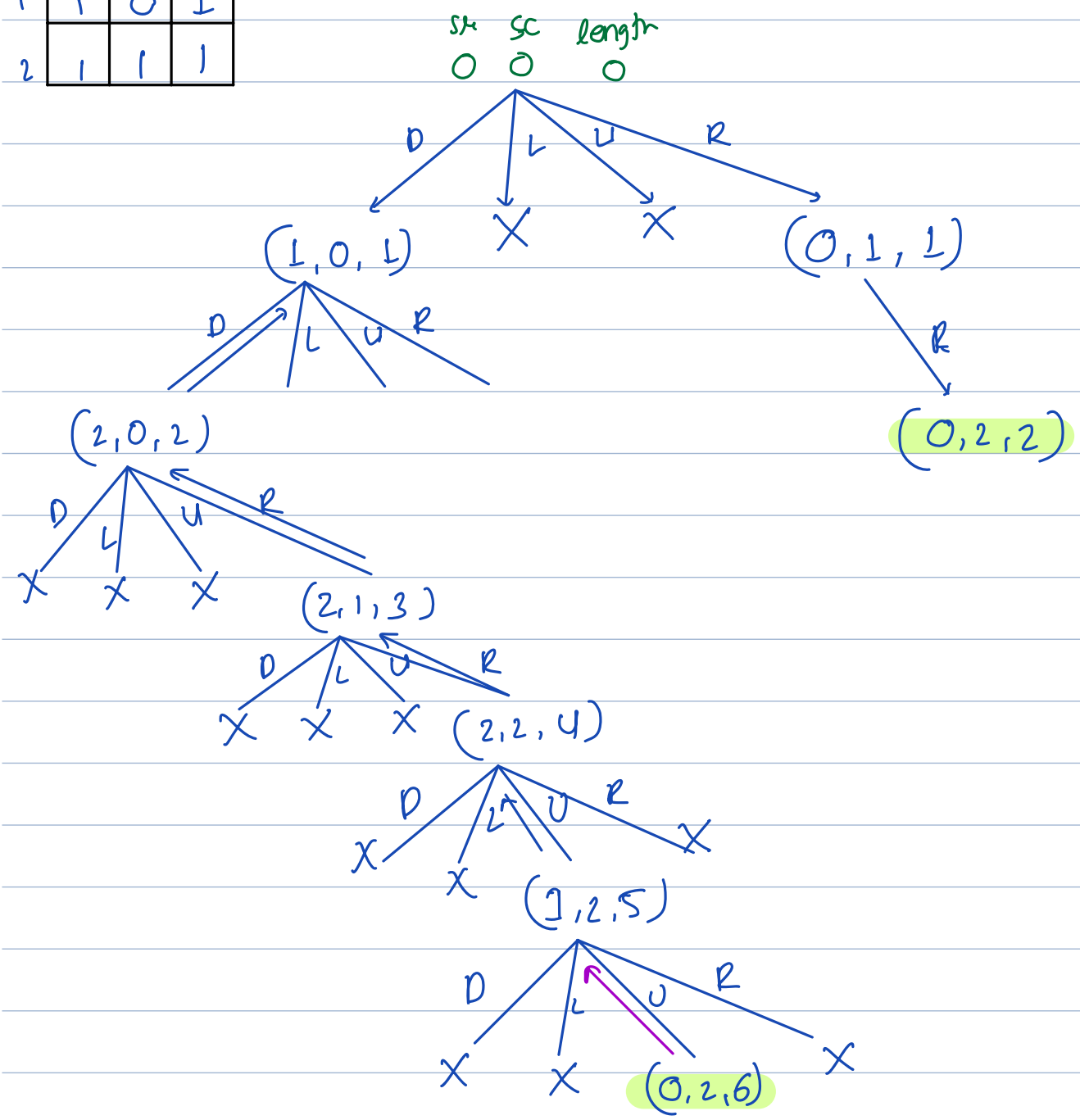
To handle infinite rec

Lets use visited matrix

$sr = 0 \quad sc = 0$

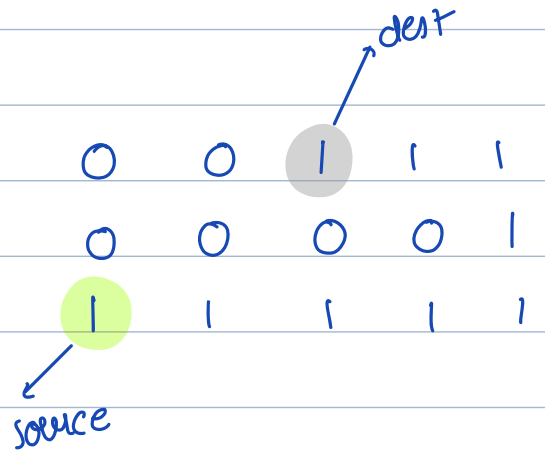
$dr = 0 \quad dc = 2$

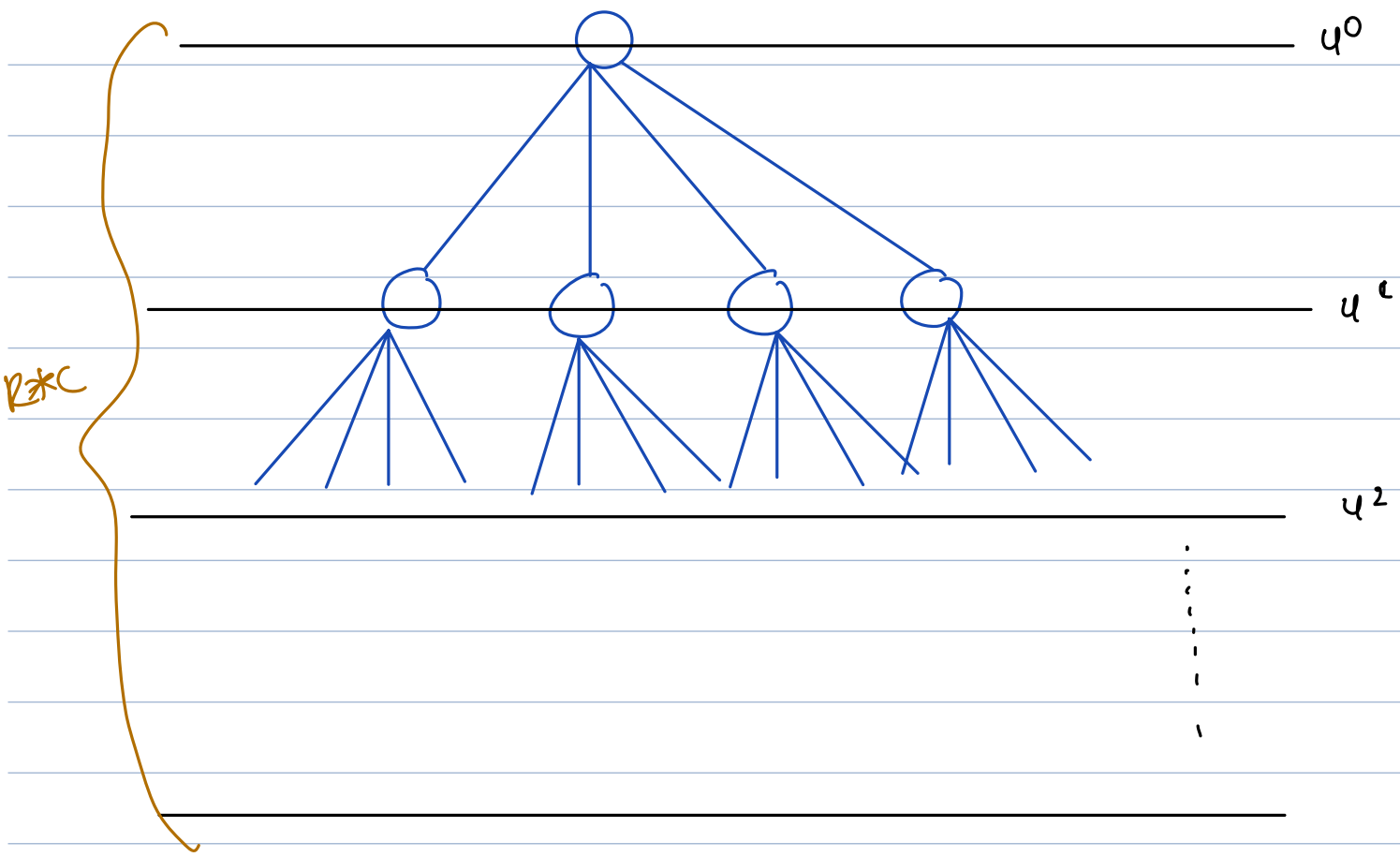
	0	1	2
0	1	1	1
1	1	0	1
2	1	1	1



```
minlength = ∞
A[][]   // global
main ( A[][], sr, sc, dr, dc ) {
    minlength = ∞
    visited[][]   // boolean matrix of same dimension as A[][]
    findShortestPathLength ( sr, sc, dr, dc, length, visited )
                                                  0

    if (minLength == ∞)  ret -1
    ret minLength
}

void findShortestPathLength ( sr, sc, dr, dc, length, visited)
    if ( sr < 0 || sr ≥ A.length || sc < 0 || sc ≥ A[0].length
        || A[sr][sc] == 0 || visited[sr][sc] ) {
        return
    }

    // when we reach dest
    if ( sr == dr && sc == dc ) {
        ans = min (ans, length)
        return
    }

    visited[sr][sc] = true        // DO
    // Down
    findShortestPathLength ( sr+1, sc, dr, dc, length+1, visited)
    // Left
    findShortestPathLength ( sr, sc-1, dr, dc, length+1, visited)
    // Up
    findShortestPathLength ( sr-1, sc, dr, dc, length+1, visited)
```

```
// Right
findShortestPathLength(sr, sc+1, dr, dc, length+1, visited)
visited[sr][sc] = false   // undo
}
```





TC : $O(4^{(R*C)}) \approx 3^{(R*C)}$

SC : $O(R*C)$ height of my tree

worst case path