Q Imp Ques

Rat in a maze:

src	0	0	0
1	1	0	0
1	1	0	0
0	1	1	dest

0 → You can't go from here
1 → You can go from here.

rat is at source (src) it can make 4 movement either up, down, left right

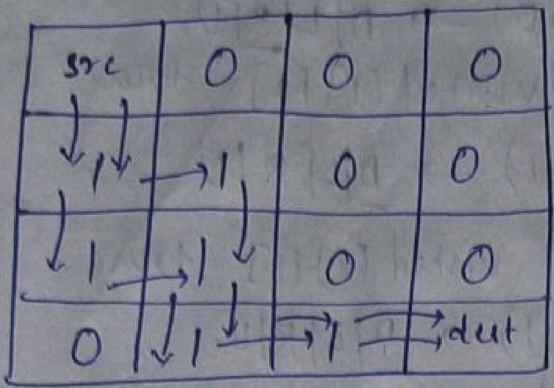
up
left ← rat → Right
↓
Down

rat can take 1 movement at a time.

find all possible solutions to reach from src to destination.

~~You need~~

→ Find the total possible ways to reach from source to destination.



src	0	0	0
1	1	0	0
1	1	0	0
0	1	1	dest

D → Down U → Up
R - Right L - Left.

① DDRDRR
② DRDDRR

→ These are the possible ways to reach from src to des.

→

	0	1	2	3
0	src 1	0	0	0
1	1	1	0	0
2	0	1	1	0
3	0	1	1	dest 1

(0,0) → D|~~L~~|R|U
↓
(1,0) → ~~D~~|~~L~~|R|U.
↓
(1,1) → D|L|R|U.
↓
(2,1) → D|L|R|U.
↓
(3,2) ← (3,1) → ~~D~~|~~L~~|R|U

↑ Infinite Loop
(3,1)
↑— (3,2) ← (3,1) ← ~~D~~|L|R|U

we will stuck in infinite loop as we reach (3,2) will check for D|L|R|U we have 1 in left of (3,2) so we will go their ~~same~~ same case of (3,1) we will check for D|L|R|U we have 1 in right of (3,1) so we will go their but it is already visited the indexes we have visited will be mark as visited. and we can't go their again.

→ let understand it with an example.

	0	1	2	3
0	src 1	0	0	0
1	↓1 →	1	1	0
2	0	↓1	1	0
3	0	1	1	dest 1

1	0	0	0
1~~0~~	1~~0~~	0	0
0	1~~0~~	0	0
0	1~~0~~	1~~0~~	~~0~~1

visited 2-D Array

safe()

1. index inside an array
2. 1 is present or not.
3. not visited.

(0,0) → <u>D</u>|L|R|U.

↓ visited[1][0] = true

(1,0) → D|L|<u>R</u>|U.

↓ visited[1][1] = true

(1,1) → <u>D</u>|L|R|U

↓ visited[2][1] = true.

(2,1) → <u>D</u>|L|R|U.

↓ visited[3][1] = true

(3,1) → D|L|<u>R</u>|U.

↓ visited[3][2] = true

(3,2) → D|L|<u>R</u>|U.

↓ visited[3][3] = true.

(3,3) → D

↓

B.C (dest reached)

DRDDRR → ① solution

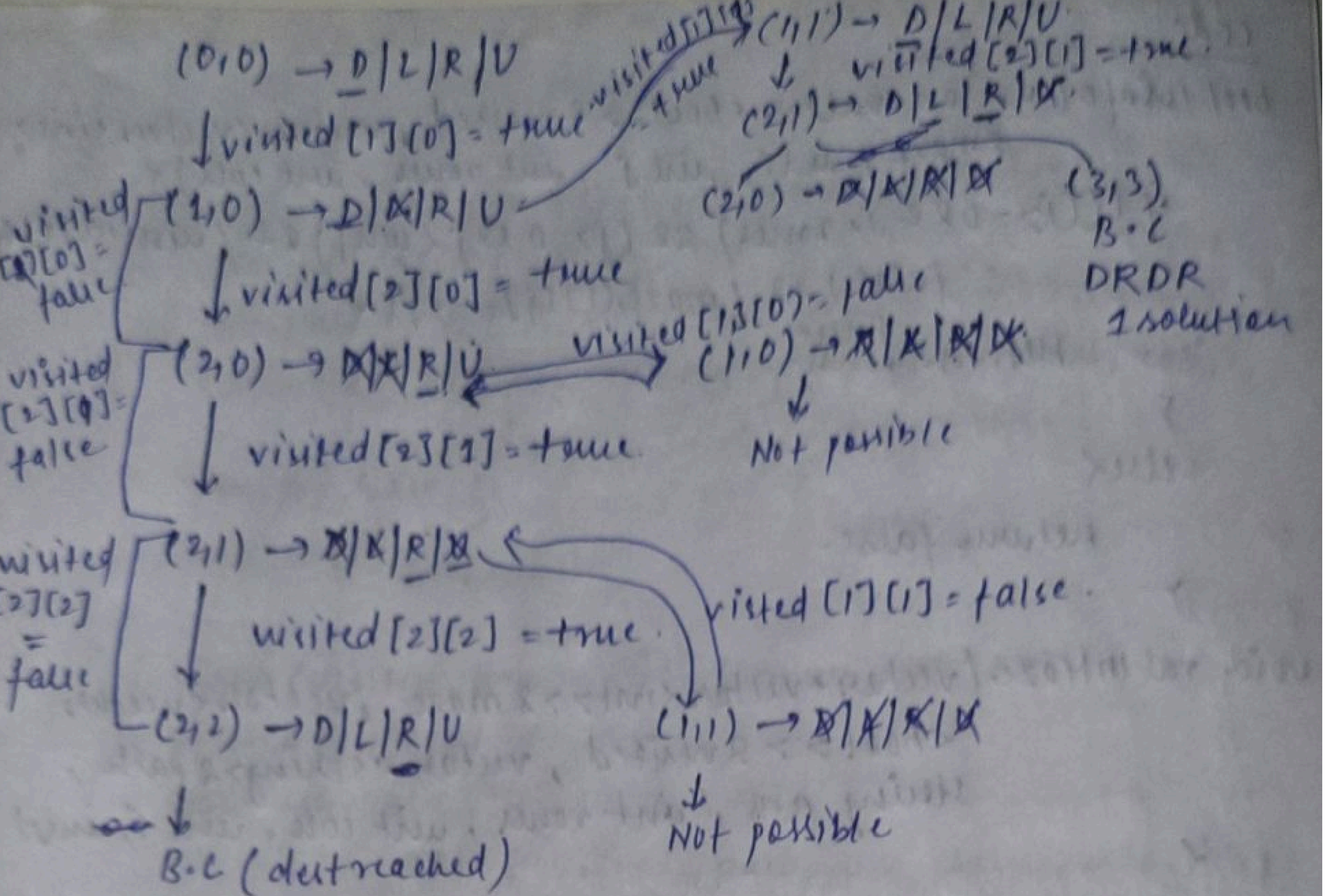
→ **Another Example**

once we have marked visited[i][j] = true while returning back we need to mark it visited[i][j] = false. because we need to find all possible solution if visited remain true can't go from there

	0	1	2
0	src	0	0
1	↓1	1	0
2	↓1 →	1 →	1 dest

1	0	0
~~1 0~~ 0 (1,0)	1 ~~0~~ ~~1~~	0
~~1 0~~ 0	~~1 0~~ 0	~~1 0~~ 0

visited.



DDRR

1 solution

→

	0	1	2	3
0	(0,0)	(0,1)	(0,2)	(0,3)
1	(1,0)	(1,1)	(1,2)	(1,3)
2	(2,0)	(2,1)	(i,j)	
3	(3,0)	(3,1)		(3,3)

If we are at (i,j)

Down (D) → (i+1, j)

Right (R) → (i, j+1)

Left (L) → (i, j-1)

Up (U) → (i-1, j)

	j-1	j	j+1	
i-1		(i-1, j)		
i	(i, j-1)	(i, j)	(i, j+1)	
i+1		(i+1, j)		

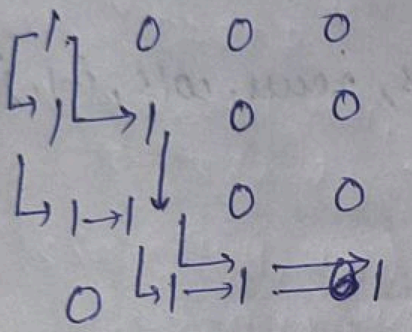
code

```
bool isSafe( vector <vector <bool>> &visited , vector <vector <int>
            &maze , int i, int j , int rows , int cols ) {
    if ((( i>=0 && i< rows) && (j>=0 && j< cols)) && (visited[i][j]
            == false) && (maze[i][j]==1)) {
        return ~~false~~ true;
    }
    else{
        return false;
    }
}
void ratInMaze (vector<vector<int>> &maze , vector<vector
                <bool>> &visited , vector<string>& path ,
                string ans , int rows , int cols , int i, int j)
{
    //base case
    if (i== rows-1 && j== cols-1){
        path.push_back(ans);
        return;
    }
    if(isSafe(visited, maze, i+1, j, rows, cols)){
        visited[i+1][j] = true;
        ans.push_back ('D').
        ratInMaze (maze, visited, path, ans, rows, cols,
                   i+1, j);
        ans.pop_back().
        visited[i+1][j] = false;
    }
    if (isSafe (visited, maze, i, j-1, rows, cols)){
        visited[i][j-1] = true.
        ans.push_back ('L').
        ratInMaze (maze, visited, path, ans, rows, cols, i,
                   j-1);
        ans. pop-back();
        visited [i][j-1] = false;
    }
    if(isSafe (visited, maze, i, j+1, rows, cols)){
        visited [i][j+1] = true;
        ans.push.back ('R');
        ratInMaze (maze, visited, path, ans, rows, cols, i,
                   j+1);
        ans.pop-back();
        visited [i][j+1] = false;
    }
    if (isSafe (visited, maze, i-1, j, rows, cols)){
        visited [i-1][j]= true;
        ans.push-back ('U');
        ratInMaze (maze, visited, path, ans, rows, cols, i-1, j);
        ans.pop_back();
        visited [i-1][j]= false;
    }
}
int main(){
    int rows, cols;
    cin >> rows >> cols;
    vector <vector <int> > maze (rows, vector <int> (cols, 0));
    for (int i=0 ; i< maze.size(); i++){
        for( int j=0 ; j< maze[i]. size(); j++){
            cin >> maze[i][j];
        }
    }
    if (maze [0][0] == 0) {
        cout << "No path exists";
        return 0;
    }
```

```
vector <vector<bool> > visited (rows , vector <bool> (cols, false));
vector <string> path.
string ans = "";
int i=0 , j=0,
ratInMaze (maze,visited , path, ans, rows, cols, i, j);
cout << "Printing of final ans "<<endl.
 for (int i=0 ;  i< path.size() .; i++) {
        cout<< path[i]<<" ";
 }
}.
```

Output

4
4



DDRDRR] → Possible
DRDDRR

Dry Run:

4
4

maze array

	0	1	2	3
0	srf	0	0	0
1	1 (1,0)	1	0	0
2	1 (2,0)	1 (2,1)	0	0
3	0 (3,0)	1 (3,1)	1 (3,2)	1 dest (3,3)

visited array

	0	1	2	3
0	1	0	0	0
1	0 ~~1~~ (1,0)	0	0	0
2	0 ~~1~~ (2,0)	0 (2,1)	0	0
3	0 ~~1~~ (3,0)	1 ~~0~~ 1 0 (3,1)	~~0~~ 1 0 (3,2)	~~0~~ 1 0 (3,3)

visited[0][0]= true.
ans= "".
ratInMaze(maze, visited, path, ans, 4,4,0,0)
    0=3 && 0=3 X
isSafe(visited, maze, 1,0,4,4)

(1>0 && 1<4) && (0>=0 && 0<4) &&] true.
visited[0][0] = 0 .
maze[1][0] = 1

true.

ans = "D'
ratInMaze (maze, visited, paths, 'D', 4, 4, 1, 0)
1=3 && 0=3

is Safe(visited., maze, 2, 0, 4, 4)
(2>=0 && 2<4) && (0>=0 && 0<4) &&] true
visited.[2][0] = 0;
maze[2][0] = 1;

ans = "DD"
ratInMaze(maze, visited , path, "DD", 4, 4, 3, 0)
3=3 && 0=3
isSafe (visited, maze, 3,0 }, 4, 4)
(3>=0 && 3<4) && (0>=0 && 0<4) &&] ~~true~~ false.
visited [3][0] = 0.
maze [3][0] = 1. X

~~ans~~ isSafe(4,4, 1, -1)] false
-1>=0 X

isSafe(4,4, 2,1)
(2>=0 && 2<4) && (1>0 && 1<4) &&] true.
visited [2][1] = 0 &&
~~visited~~ maze [2][1] = 1

ans = "DDR"
ratInMaze("DDL", 4,4, 2,1),
2==3 && 1==3 X

Is Safe (4,4, 3,1);
(3>=0 && 3<4) && (1>0 && 1<4) &&.] true.
visited [3][1] = 0 &&
maze [3][1] = 1

~~ans = "DDRD~~.

ans = "DDRD"

is safe(4, 4, 4, 1)] false
4<4

if safe (4, 4, 3, 0)
(3>=0 && 3<4) && (0>=0 && 0<4) &&] false
visited [3] [0] = 0 X

if safe (4, 4, 3, 2)
(3>=0 && 3<4) && (2>=0 && 2<4) &&] true
visited [3] [2] = 0 &&
maze [3] [2] = 1

ans = "DDRDR"

ratInMaze (4, 4, 3, 2)
3==3 && 2==3 X

is safe (4, 4, 4, 2)
(4>=0 && 4<4) X] false Down

is safe (4, 4, 3, 1)
(3>=0 && 3<4) && (1>=0 && 1<=4) &&] false Left
visited [3] [1] = 0 X

is safe (4, 4, 3, 3)
(3>=0 && 3<4) && (3>=0 && 3<4) &&] true
visited [3] [3] = 0 &&
maze [3] [3] = 1 &&

ans = "DDRDRR"

ans = "DDRDR"
"DRDDRR

ratInMaze (4, 4, 3, 3)
3==3 && 3==3 True

path = | DDRDRR | DRDDRR |