Rat in a Maze Problem – I

Consider a rat placed at **(0, 0)** in a square matrix **mat** of order **n* n**. It has to reach the destination at **(n - 1, n - 1)**. Find all possible paths that the rat can take to reach from source to destination. The directions in which the rat can move are **'U'(up)**, **'D'(down)**, **'L' (left)**, **'R' (right)**. Value 0 at a cell in the matrix represents that it is blocked and rat cannot move to it while value 1 at a cell in the matrix represents that rat can be travel through it.
Note: In a path, no cell can be visited more than one time. If the source cell is 0, the rat cannot move to any other cell. In case of no path, return an empty list. The driver will output **"-1"** automatically.

Examples:

Input: mat[][] = [[1, 0, 0, 0],

[1, 1, 0, 1],

[1, 1, 0, 0],

[0, 1, 1, 1]]

Output: DDRDRR DRDDRR

Explanation: The rat can reach the destination at (3, 3) from (0, 0) by two paths - DRDDRR and DDRDRR, when printed in sorted order we get DDRDRR DRDDRR.

Input: mat[][] = [[1, 0],

[1, 0]]

Output: -1

Explanation: No path exists and destination cell is blocked.

Expected Time Complexity: $O(3^{n\hat{}2})$
Expected Auxiliary Space: O(l * x)
Here l = length of the path, x = number of paths.

Constraints:
$2 \leq n \leq 5$
$0 \leq mat[i][j] \leq 1$

```
from typing import List

class Solution:
    def findPath(self, m: List[List[int]]) -> List[str]:
        n = len(m)

        def isSafe(x, y):
            return 0 <= x < n and 0 <= y < n and m[x][y] == 1
and not visited[x][y]

        def solve(x, y, path):
            if x == n - 1 and y == n - 1:
                result.append(path)
                return
            visited[x][y] = True
            if isSafe(x + 1, y):  # Move Down
                solve(x + 1, y, path + 'D')
            if isSafe(x, y - 1):  # Move Left
                solve(x, y - 1, path + 'L')
            if isSafe(x, y + 1):  # Move Right
                solve(x, y + 1, path + 'R')
            if isSafe(x - 1, y):  # Move Up
                solve(x - 1, y, path + 'U')
            visited[x][y] = False

        result = []
        visited = [[False] * n for _ in range(n)]
        if m[0][0] == 1:
            solve(0, 0, "")
        return result
```