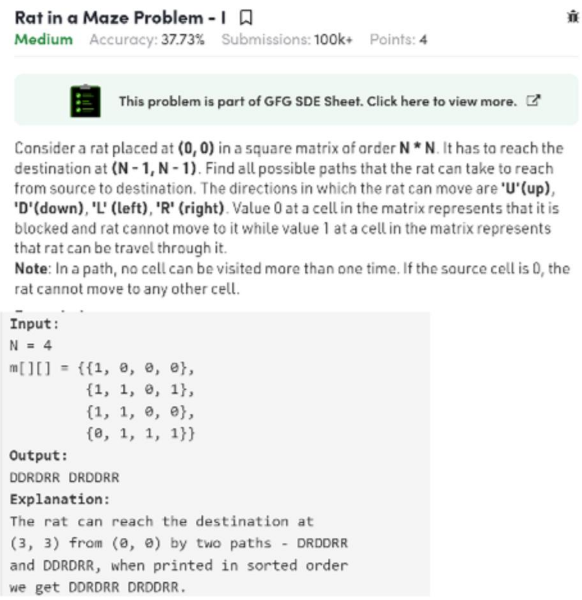
18. Rat in A Maze



Rat in a Maze Problem - I

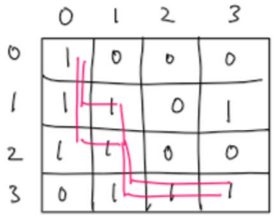
Medium Accuracy: 37.73% Submissions: 100k+ Points: 4

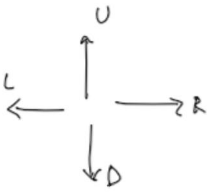
This problem is part of GFG SDE Sheet. Click here to view more.

Consider a rat placed at **(0, 0)** in a square matrix 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.

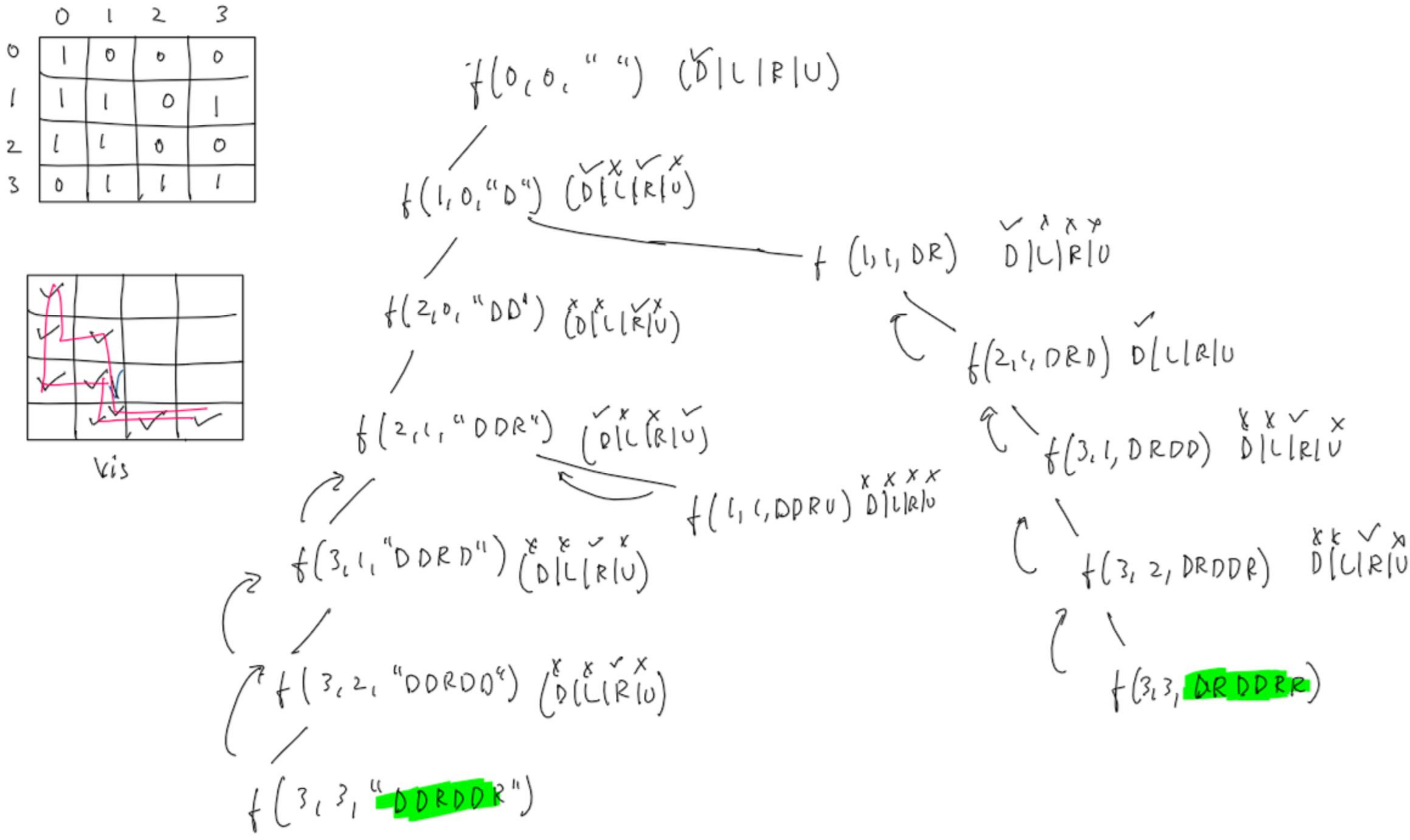
```
Input:
N = 4
m[][] = {{1, 0, 0, 0},
         {1, 1, 0, 1},
         {1, 1, 0, 0},
         {0, 1, 1, 1}}
Output:
DDRDRR DRDDRR
Explanation:
The rat can reach the destination at
(3, 3) from (0, 0) by two paths - DRDDRR
and DDRDRR, when printed in sorted order
we get DDRDRR DRDDRR.
```





DDRDRR < DRDDRR

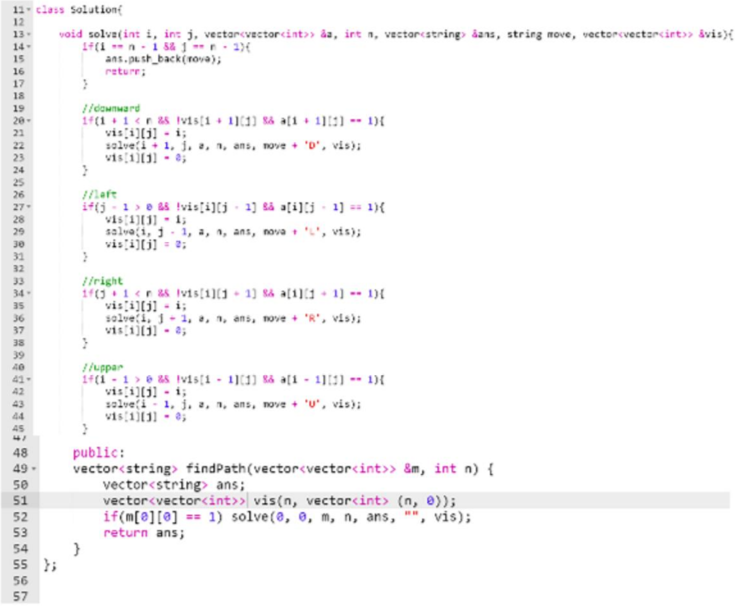
lexiographically order => DLRU


0 1 2 3
0 1 0 0 0
1 1 1 0 1
2 1 1 0 0
3 0 1 1 1
vis
f(0, 0, " ") (D|L|R|U)
f(1, 0, "D") (D|L|R|U) ✓ X ✓ X
f(1, 1, DR) D|L|R|U ✓ X X X
f(2, 0, "DD") (D|L|R|U) X X ✓ X
f(2, 1, DRD) D|L|R|U ✓
f(2, 1, "DDR") (D|L|R|U) ✓ X X ✓
f(1, 1, DDRU) D|L|R|U X X X X
f(3, 1, DRDD) D|L|R|U X X ✓ X
f(3, 1, "DDRD") (D|L|R|U) X X ✓ X
f(3, 2, DRDDR) D|L|R|U X X ✓ X
f(3, 2, "DDRDD") (D|L|R|U) X X ✓ X
f(3, 3, DRDDRR)
f(3, 3, "DDRDDR")

Pseudo Code :

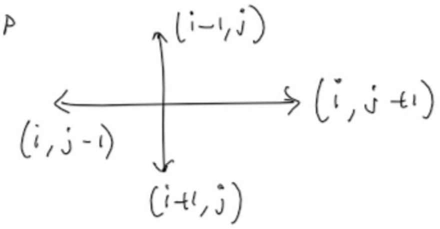
```
f()
{
    Down
    f()
    Left
    f()
    Right
    f()
    Upwards
    f()
}
```

$T.C = O(4^{n*m})$ $S.C = O(nm)$



```cpp
class Solution{

    void solve(int i, int j, vector<vector<int>> &a, int n, vector<string> &ans, string move, vector<vector<int>> &vis){
        if(i == n - 1 && j == n - 1){
            ans.push_back(move);
            return;
        }

        //downward
        if(i + 1 < n && !vis[i + 1][j] && a[i + 1][j] == 1){
            vis[i][j] = 1;
            solve(i + 1, j, a, n, ans, move + 'D', vis);
            vis[i][j] = 0;
        }

        //left
        if(j - 1 >= 0 && !vis[i][j - 1] && a[i][j - 1] == 1){
            vis[i][j] = 1;
            solve(i, j - 1, a, n, ans, move + 'L', vis);
            vis[i][j] = 0;
        }

        //right
        if(j + 1 < n && !vis[i][j + 1] && a[i][j + 1] == 1){
            vis[i][j] = 1;
            solve(i, j + 1, a, n, ans, move + 'R', vis);
            vis[i][j] = 0;
        }

        //upper
        if(i - 1 >= 0 && !vis[i - 1][j] && a[i - 1][j] == 1){
            vis[i][j] = 1;
            solve(i - 1, j, a, n, ans, move + 'U', vis);
            vis[i][j] = 0;
        }

    public:
    vector<string> findPath(vector<vector<int>> &m, int n) {
        vector<string> ans;
        vector<vector<int>> vis(n, vector<int> (n, 0));
        if(m[0][0] == 1) solve(0, 0, m, n, ans, "", vis);
        return ans;
    }
};
```

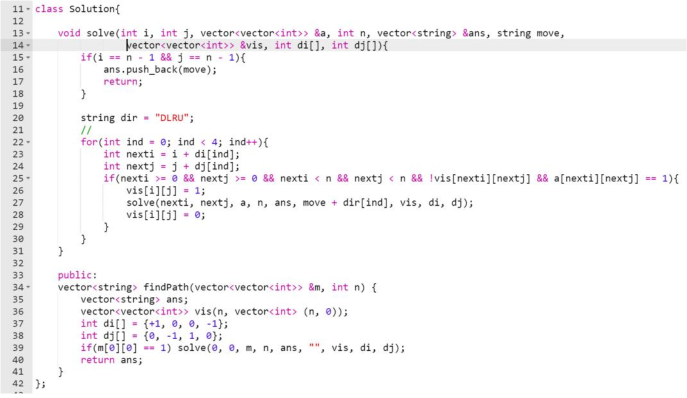
But writing individual code for every direction is a lengthy process therefore we truncate the '4' if statements into a single loop



	D	L	R	U
di[]	+1	+0	+0	-1
dj[]	+0	-1	+1	+0

⇒ i + di[ind]

⇒ j + dj[ind]



```cpp
class Solution{

    void solve(int i, int j, vector<vector<int>> &a, int n, vector<string> &ans, string move,
                    vector<vector<int>> &vis, int di[], int dj[]){
        if(i == n - 1 && j == n - 1){
            ans.push_back(move);
            return;
        }

        string dir = "DLRU";
        //
        for(int ind = 0; ind < 4; ind++){
            int nexti = i + di[ind];
            int nextj = j + dj[ind];
            if(nexti >= 0 && nextj >= 0 && nexti < n && nextj < n && !vis[nexti][nextj] && a[nexti][nextj] == 1){
                vis[i][j] = 1;
                solve(nexti, nextj, a, n, ans, move + dir[ind], vis, di, dj);
                vis[i][j] = 0;
            }
        }
    }

    public:
    vector<string> findPath(vector<vector<int>> &m, int n) {
        vector<string> ans;
        vector<vector<int>> vis(n, vector<int> (n, 0));
        int di[] = {+1, 0, 0, -1};
        int dj[] = {0, -1, 1, 0};
        if(m[0][0] == 1) solve(0, 0, m, n, ans, "", vis, di, dj);
        return ans;
    }
};
```