Find paths

10 June 2022 00:52

Rat in a Maze Problem - I

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

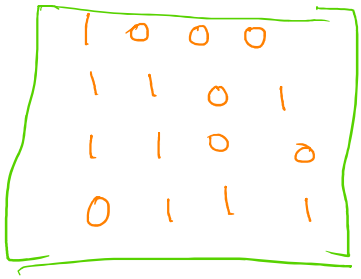
This problem is part of GFG SDE Sheet. Click here to view more.

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.

Example 1:

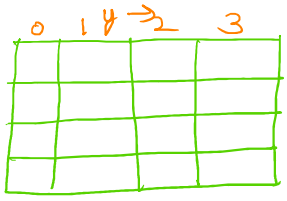
```
Input:
N = 4
m[][] = {{1, 0, 0, 0},
         {1, 1, 0, 1},
         {1, 1, 0, 0},
         {0, 1, 1, 1}}
Output:
DDRDRR DRDDRR
Explanation:
The rat can reach the destination at
(3, 3) from (0, 0) by two paths - DRDDRR
and DDRDRR, when printed in sorted order
we get DDRDRR DRDDRR.
```

Approach :-

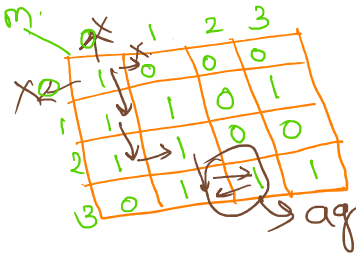


1 → open path. Allocated.
0 → closed path.

src → (0,0)
dest → (n-1, n-1)



up → (x-1, y)
down → (x+1, y)
left (x, y-1)
Right (x, y+1)

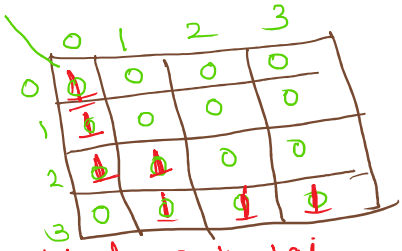


src x = 0
src y = 0
(0,0) → D/L/R/U.

agar Again left me bhejenge then infinite loop.
to handle this condition.

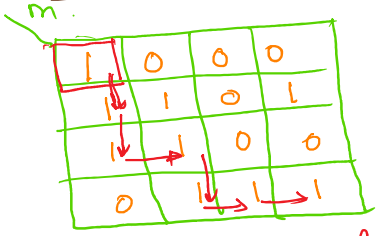
We create a visited array.

Visited



Yaha check karenge kya ye visited mark hai

Agar nhi then visited mark kar diya.



DDDRDRR possible solution

three Base case

① matrix ke Andar honi chahiye
② m[k][l] = 1
③ visited[k][l] = 0

safe to move

kar diya

three Base case

① matrix ke Andar honi chahiye

② m[k][l] = 1 safe to move

③ visited[k][l] = 0

```
    public:
    vector<string> findPath(vector<vector<int>> &m, int n) {
        vector<string> ans;

        if(m[0][0] == 0)
        {
            return ans;
        }

        int x = 0;
        int y = 0;
        vector<vector<int>> visited = m;

        for(int i = 0; i < n; i++)
        {
            for(int j = 0; j < n; j++)
            {
                visited[i][j] = 0;
            }
        }

        string path = "";

        solve(m,n,ans,path,visited,x,y);
        sort(ans.begin(), ans.end());
        return ans;
    }
};
```

```
private:
bool issafe(int x, int y, int n, vector<vector<int>>m, vector<vector<int>> visited)
{
    if((x >= 0 && x < n) && (y >= 0 && y < n) && m[x][y] == 1 && visited[x][y] == 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}


void solve(vector<vector<int>>&m, int n, vector<string>&ans, string path,
    vector<vector<int>> &visited,int x, int y)
    {
        //Base case
        if(x == n-1 && y == n-1)
        {
            ans.push_back(path);
            return;
        }

        visited[x][y] = 1;

        //Down
        int newx = x + 1;
        int newy = y;
        if(issafe(newx, newy, n, m, visited))
        {
            path.push_back('D');
            //Recursive call
            solve(m, n, ans, path, visited, newx, newy);
```