Lec-23

13/06/25 8:25 PM

You are given an N*M grid. Each cell (i,j) in the grid is either blocked, or empty. The rat can move from a position towards left, right, up or down on the grid.

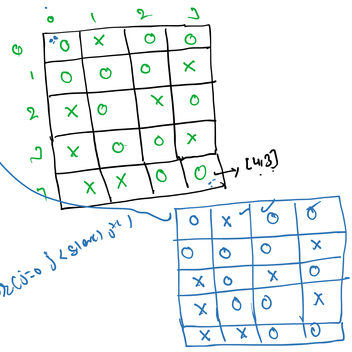
Initially rat is on the position (1,1). It wants to reach position (N,M) where it's cheese is waiting for. If a path exists-it is always unique. Find that path and help the rat reach its cheese.

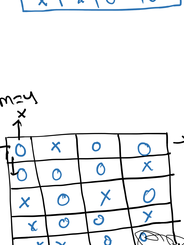
**Input**

```
5 4
0X00
000X
X0X0
X00X
XX00
```

**Output**

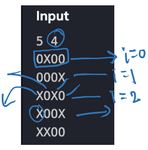
```
1 0 0 0
1 1 0 0
0 1 0 0
0 1 1 0
0 0 1 1
```





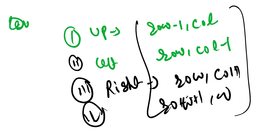
```
import java.util.Scanner;

public class Rat_Chases_its_cheese {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int m = sc.nextInt();
        char[][] maze = new char[n][m];
        for (int i = 0; i < maze.length; i++) {
            String s = sc.next();
            for (int j = 0; j < maze[0].length; j++) {
                maze[i][j] = s.charAt(j);
            }
        }

    }
}
```



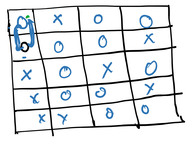


You are given an N*M grid. Each cell (i,j) in the grid is either blocked, or empty. The rat can move from a position towards left, right, up or down on the grid.

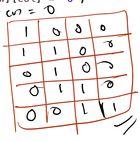
Initially rat is on the position (1,1). It wants to reach position (N,M) where it's cheese is waiting for. If a path exists-it is always unique. Find that path and help the rat reach its cheese.

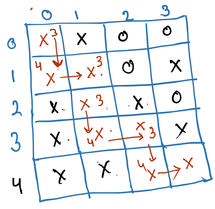


```
public static void printpath(char[][] maze, int row, int col) {
        if (row < 0 || col < 0 || row >= maze.length || col >= maze[0].length) {
            return;
        }
        printpath(maze, row - 1, col);// up
        printpath(maze, row, col - 1);// left
        printpath(maze, row + 1, col);// down
        printpath(maze, row, col + 1);// right

    }
```

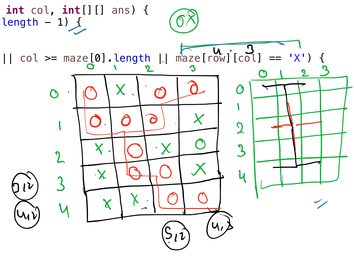


```
public static void printpath(char[][] maze, int row, int col) {
        if (row < 0 || col < 0 || row >= maze.length || col >= maze[0].length || maze[row][col] == 'X') {
            return;
        }
        maze[row][col] = 'X';
        printpath(maze, row - 1, col);// up
        printpath(maze, row, col - 1);// left
        printpath(maze, row + 1, col);// down
        printpath(maze, row, col + 1);// right
        maze[row][col] = 'O';
    }
```





```
public static void printpath(char[][] maze, int row, int col, int[][] ans) {
        if (row == maze.length - 1 && col == maze[0].length - 1) {
            Display(ans);
        }
        if (row < 0 || col < 0 || row >= maze.length || col >= maze[0].length || maze[row][col] == 'X') {
            return;
        }
        maze[row][col] = 'X';
        ans[row][col] = 1;
        printpath(maze, row - 1, col, ans);// up
        printpath(maze, row, col - 1, ans);// left
        printpath(maze, row + 1, col, ans);// down
        printpath(maze, row, col + 1, ans);// right
        maze[row][col] = 'O';
        ans[row][col] = 0;
    }
```



```
public static void printpath(char[][] maze, int row, int col, int[][] ans) {
        if (row == maze.length - 1 && col == maze[0].length - 1 && maze[row][col] != 'X') {
            ans[row][col] = 1;
            Display(ans);
            val = 1;
            return;
        }
        if (row < 0 || col < 0 || row >= maze.length || col >= maze[0].length || maze[row][col] == 'X') {
            return;
        }
        maze[row][col] = 'X';
        ans[row][col] = 1;
        printpath(maze, row - 1, col, ans);// up
        printpath(maze, row, col - 1, ans);// left
        printpath(maze, row + 1, col, ans);// down
        printpath(maze, row, col + 1, ans);// right
        maze[row][col] = 'O';
        ans[row][col] = 0;
    }
```