75 Days of Code
Day 47
Problem no : 1926
Problem Title : Nearest Exit from Entrance in Maze
Type : Graph / DFS

You are given an m x n matrix maze (0-indexed) with empty cells (represented as '.') and walls (represented as '+'). You are also given the entrance of the maze, where entrance = [entrancerow, entrancecol] denotes the row and column of the cell you are initially standing at.

In one step, you can move one cell up, down, left, or right. You cannot step into a cell with a wall, and you cannot step outside the maze. Your goal is to find the nearest exit from the entrance. An exit is defined as an empty cell that is at the border of the maze. The entrance does not count as an exit.

Return the number of steps in the shortest path from the entrance to the nearest exit, or -1 if no such path exists.

Example 1:

Input: maze = [["+","+",".","+"],[".",".",".","+"],["+","+","+","."]], entrance = [1,2]
Output: 1
Explanation: There are 3 exits in this maze at [1,0], [0,2], and [2,3].
Initially, you are at the entrance cell [1,2].
- You can reach [1,0] by moving 2 steps left.
- You can reach [0,2] by moving 1 step up.

It is impossible to reach [2,3] from the entrance.
Thus, the nearest exit is [0,2], which is 1 step away.
Example 2:

Input: maze = [["+","+","+"],[".",".","."],["+","+","+"]], entrance = [1,0]
Output: 2
Explanation: There is 1 exit in this maze at [1,2].
[1,0] does not count as an exit since it is the entrance cell.
Initially, you are at the entrance cell [1,0].
- You can reach [1,2] by moving 2 steps right.
Thus, the nearest exit is [1,2], which is 2 steps away.
Example 3:

Input: maze = [[".","+"]], entrance = [0,0]
Output: -1
Explanation: There are no exits in this maze.

```typescript
function nearestExit(maze: string[][], entrance: number[]): number {
    const rows: number = maze.length;
    const cols: number = maze[0].length;
    const directions: number[][] = [[0, 1], [1, 0], [0, -1], [-1, 0]]; // Right, Down, Left, Up

    // Initialize the queue for BFS
    const queue: number[][] = [[entrance[0], entrance[1], 0]]; // [row, col, steps]

    // Mark the entrance cell as visited
    maze[entrance[0]][entrance[1]] = '+';

    while (queue.length > 0) {
        const [row, col, steps]: number[] = queue.shift()!;

        // Check if we have reached an exit
        if ((row !== entrance[0] || col !== entrance[1]) && (row === 0 || row === rows - 1 || col === 0 || col === cols - 1)) {
            return steps;
        }

        // Explore neighboring cells
        for (const [dr, dc] of directions) {
            const r: number = row + dr;
            const c: number = col + dc;

            // Check if the neighboring cell is valid and not a wall
            if (r >= 0 && r < rows && c >= 0 && c < cols && maze[r][c] === '.') {
                maze[r][c] = '+';
                queue.push([r, c, steps + 1]);
            }
        }
    }

    // If no exit is found, return -1
    return -1;
};
```

Accepted

Editorial + Soluti

Runtime Details

98 ms

Beats 94.39% of users with TypeScript

Memory Detail

51.40 MB

Beats 83.18% of users with TypeScript

Shubham Agrahari

Shubham Agrahari