Assignment 4: Tree and Traversal (30 Points)

Maze Escape

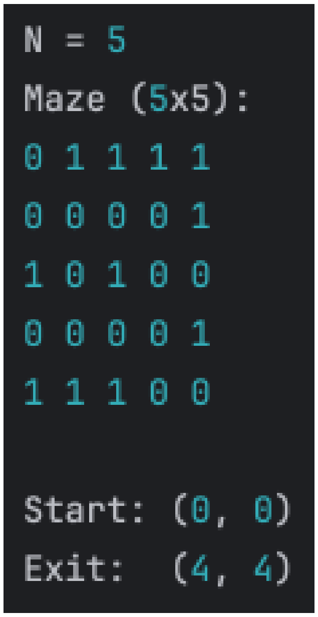
Concept: Create a maze game on an N×N grid with a **start** position and an **exit** position. The player must write code to move a character from the start to the exit, moving **up/down/left/right** by one cell at a time. Certain cells are **walls** (impassable) while others are **paths** (walkable). The challenge is to find a way through the maze—preferably the **shortest path**—to reach the exit.

Challenge Details

1. **Maze Setup**
    - The maze is represented by an N×N grid.
    - Each cell in the grid can be:
        - **0** (walkable path),
        - **1** (wall/obstacle).
    - Input also includes the start coordinates and the exit coordinates.
2. **Movement Rules**
    - From any position, you can move one step at a time in one of four directions: **up, down, left, or right**.
    - You must ensure you don't collide with walls (cells marked **1**) or move outside the grid boundaries.
3. **Goal**
    - Write code to move the character from the **start** cell to the **exit** cell successfully.
    - (Optional) Display the number of steps taken or show the actual path taken from start to exit.
4. **Possible Approaches**
    - **Breadth-First Search (BFS)**: Highly suitable to find the **shortest path** in an unweighted grid.
    - **Depth-First Search (DFS)** with a **stack** could also work to find *a* path (not necessarily the shortest).
5. **Output**
    - If a path exists, you can output the sequence of cells visited, the directions moved, or at least the path length.
    - If no path exists (the exit is blocked off by walls, for example), display a message such as "No path found."

Example Input/Output

Sample Input:



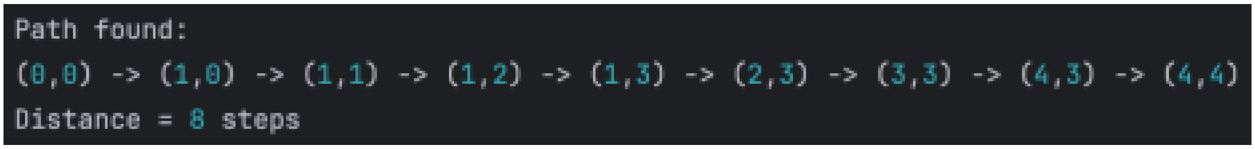
```
N = 5
Maze (5x5):
0 1 1 1 1
0 0 0 0 1
1 0 1 0 0
0 0 0 0 1
1 1 1 0 0

Start: (0, 0)
Exit:  (4, 4)
```

Here,

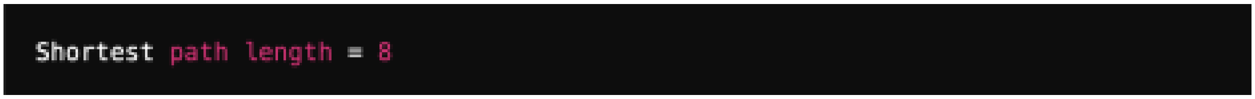
- `0` = walkable path
- `1` = wall/obstacle
- The start is at (0,0) (top-left corner).
- The exit is at (4,4) (bottom-right corner).

Sample Output:



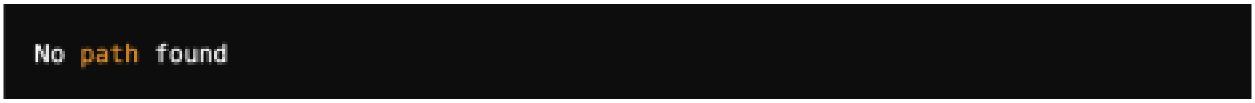
```
Path found:
(0,0) -> (1,0) -> (1,1) -> (1,2) -> (1,3) -> (2,3) -> (3,3) -> (4,3) -> (4,4)
Distance = 8 steps
```

Alternatively:



```
Shortest path length = 8
```

If there were no valid path, display something like:



```
No path found
```