problem nr. 1

Session 1-Report

Session 1-Homework

Maze Data Structure (Tree-Based)

1. **Cell Structure**:
    - Each cell in the maze represents a **node** in the tree.
    - Fields for each cell:
        - **Walls**: Four boolean fields `N`, `S`, `E`, `W` to represent walls on the North, South, East, and West sides.
        - **Visited**: A boolean field to track if the cell has been visited during maze generation.
        - **Parent**: A pointer/reference to another cell (the parent node), representing the tree structure. This helps keep track of the path back to the start node.
2. **Maze Grid**:
    - The maze is represented as a 2D array of cells, where each cell initially has all walls intact and no connections to other cells.
    - The grid's dimensions are defined by the `width` and `height` values provided.

Maze Generation Algorithm (Recursive Division with Tree Structure)

This algorithm generates a tree-like structure where each cell/node is connected by a unique path, ensuring only one possible route to the exit.

1. **Initialize the Maze**:
    - Create a grid of cells, setting each cell's walls (`N`, `S`, `E`, `W`) to `true`.
    - Set all cells as unvisited (`visited = false`), with no parent assigned initially.
2. **Start Recursive Generation**:
    - Select a **starting cell** in the grid, mark it as visited, and set it as the current cell.
    - Perform recursive division using a **Depth-First Search (DFS)**-like algorithm.
3. **Cell Traversal (Recursive DFS)**:
    - From the current cell, randomly order the four possible directions: North, South, East, and West.
    - For each direction:
        - Determine the **neighboring cell** in that direction.

        - Check if the neighboring cell is within bounds and if it has not been visited.
        - If it is valid and unvisited:
            - **Remove the wall** between the current cell and the neighboring cell in that direction.
            - Set the neighboring cell’s **parent** to the current cell, connecting them in the tree structure.
            - Mark the neighboring cell as **visited**.
            - Recursively apply the same process on the neighboring cell.
    - Repeat this process until there are no unvisited cells connected to the current path. The recursion will backtrack to the previous cell (using the parent node), and the algorithm will continue until all cells in the grid have been visited.

4. **Exit Placement**:
    - The **exit** is defined as a unique cell, typically placed at the **bottom-right corner** of the maze grid (or other specified location).
    - Since all cells are connected by a tree structure, each cell in the maze has a unique path leading to the exit.

Key Concepts

- **Tree Structure**: Each cell (node) has only one parent (connection), ensuring no cycles are created, which guarantees a unique path to the exit.
- **Backtracking**: The algorithm backtracks when it reaches a dead end (a cell with no unvisited neighbors) by returning to the parent cell.
- **Maze Characteristics**: The resulting maze will have no loops, and any point can reach the exit by following the unique path through the tree.