Maze Solver Testing Document

The goal of this testing plan is to make sure that the maze solver reads, processes, and solves different maze layouts correctly. I also want to make sure it handles errors

Categories

File Reading and Maze Initialization

Purpose: to confirm that the maze is read correctly from a file and that the start ('S') and end ('E') positions are accurately identified. The maze structure should be correctly represented in the 2D array.

Test Cases:

Valid Maze File:

- **Input:** Use a valid maze file `maze.txt`
- **Expected Outcome:** The maze should be read correctly, with the start and end positions identified, and the maze printed.

Non-Rectangular Maze File:

- **Input:** Provide a maze file with rows of different lengths.
- **Expected Outcome:** The program should detect the shape not being an rectangle and print an error message, "error: Maze is not rectangular."

Missing File:

- **Input:** Try reading from a non-existent file.
- **Expected Outcome:** The program should handle the `FileNotFoundException` and print "ERROR: File not found."

Correctness of Maze Solving Algorithm

Purpose: Verify that the maze solver can find the path from start to end in solvable mazes and correctly handle unsolvable ones.

Test Cases:

Simple Solvable Maze:

- **Input:** Use a simple maze that's easy to solve.
- **Expected Outcome:** The solver should find and mark the path, then print "Maze solved."

Complex Solvable Maze with Loops:

- **Input:** Provide a maze with loops and multiple paths.
- **Expected Outcome:** The solver should handle loops correctly, find the exit, and print "Maze solved."

Unsolvable Maze:

- **Input:** Use a maze where the path from start to end is blocked.
- **Expected Outcome:** The solver should try all possible paths and, finding none, print "No solution found."

1. **Error Handling and Input Validation**

    Purpose: Ensure the program can handle invalid inputs and provide meaningful error messages.

    Test Cases:

Invalid Characters in Maze:

- **Input:** Use a maze file with invalid characters.
- **Expected Outcome:** The program should detect the invalid characters and print "ERROR: Invalid character 'X' in maze."

Missing Start/End Point:

- **Input:** Provide a maze file without 'S' or 'E'.
- **Expected Outcome:** The program should detect the missing points and print "ERROR: Maze must have exactly one start 'S' and one end 'E'."

2. **Performance Testing**

Purpose: Check how well the solver performs under different loads.

Test Cases:

Multiple Runs on Large Mazes:

- **Input:** Run the solver on multiple large mazes one after another.
- **Expected Outcome:** The solver should handle repeated runs without slowing down.

Time Complexity Analysis:

- **Input:** Measure the time taken to solve increasingly larger mazes.
- **Expected Outcome:** The solver's performance should scale reasonably with the size of the maze.

Test Execution

Setup preparing maze files , then running the program for ech test exacuation recording the outcomes and any error messages,to validate is by comparing the programs output with expected outcomes.