CS250 - ARTIFICIAL INTELLIGENCE LAB

ASSIGNMENT-2: BFS and DFS

Date: January 17, 2024

Total Credit: 10

- Markings will be based on the correctness and soundness of the outputs.
- Marks will be deducted in case of plagiarism.
- Proper indentation and appropriate comments are mandatory.
- *All code needs to be submitted in '.py' format.* Even if you code it in '.IPYNB'format, download it in '.py' format and then submit
- You should zip all the required files and name the zip file as:
    - <roll_no>_assignment_<#>.zip, eg. 1501cs11_assignment_01.zip.

Problem Statement:

Navigating the Maze with AI

Scenario: You are building an AI agent to navigate a simple maze-like environment represented as a 2D grid. Each cell in the grid can be either empty (walkable), wall (obstacle), or the exit. The agent starts at a designated starting cell and needs to reach the exit in the shortest path possible.

Task: 1. Implement two algorithms for the agent's navigation:

- Breadth-First Search (BFS): Explore all neighbouring cells one level at a time before moving to the next level.

- Depth-First Search (DFS): Explore one branch of the search tree completely before backtracking and trying another branch.

2. For each algorithm, write Python code to find the shortest path from the starting cell to the exit in the given maze.

3. Compare the performance of BFS and DFS in terms of:

    - Number of steps taken: How many cells did the agent visit before reaching the exit?
    - Time taken: How much time did each algorithm require during execution?
    - Memory usage : How much memory did each algorithm require during execution?

- Suitability for different maze scenarios: Discuss situations where one algorithm might be preferable over the other.

4. Modify your code to display the visualization of the path taken by each algorithm on the maze grid.

5. Test your code for any 8*8 maze and 12*12 maze (code should handle any size of maze).

Maze example:

```
maze_16x16 = [
['S', 'W', 'W', 'W', ' ', ' ', ' ', ' ', 'W', ' ', ' ', 'W', ' ', ' ', ' ', 'E'],
[' ', ' ', ' ', 'W', 'W', 'W', 'W', ' ', 'W', 'W', 'W', ' ', 'W', ' ', 'W', ' '],
[' ', ' ', 'W', ' ', ' ', ' ', 'W', ' ', 'W', ' ', ' ', ' ', 'W', ' ', ' ', ' '],
[' ', 'W', ' ', ' ', 'W', ' ', 'W', ' ', ' ', ' ', 'W', ' ', 'W', 'W', 'W', ' '],
[' ', 'W', 'W', ' ', 'W', ' ', 'W', 'W', 'W', ' ', 'W', ' ', 'W', ' ', ' ', ' '],
[' ', ' ', 'W', ' ', 'W', ' ', ' ', ' ', ' ', ' ', 'W', ' ', ' ', ' ', 'W', ' '],
[' ', 'W', 'W', ' ', 'W', ' ', 'W', ' ', 'W', ' ', 'W', ' ', 'W', ' ', ' ', ' '],
[' ', ' ', ' ', ' ', 'W', ' ', 'W', ' ', ' ', ' ', ' ', ' ', 'W', ' ', 'W', ' '],
['W', 'W', 'W', 'W', 'W', ' ', 'W', ' ', 'W', 'W', 'W', 'W', 'W', ' ', 'W', ' '],
[' ', ' ', ' ', ' ', ' ', ' ', 'W', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'W', ' '],
['W', 'W', 'W', 'W', 'W', ' ', 'W', ' ', 'W', ' ', 'W', 'W', 'W', ' ', 'W', ' '],
[' ', ' ', ' ', ' ', 'W', ' ', ' ', ' ', 'W', ' ', ' ', ' ', ' ', ' ', 'W', ' '],
[' ', 'W', 'W', ' ', 'W', ' ', 'W', ' ', 'W', ' ', 'W', ' ', 'W', 'W', 'W', ' '],
[' ', 'W', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'W', ' ', ' ', ' ', ' ', ' '],
[' ', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', ' ', 'W', 'W', 'W', ' ', 'W', ' '],
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
['W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W'],
            ]
```

Note:

- S: Starting cell
- E: Exit cell
- W: Wall
- []: Empty cell
- You can choose any data structure to represent the maze and the agent's path.

For any queries regarding this assignment, contact:

Utsav Kumar Nareti (utsavkumarnareti@gmail.com)
Kumari Priya(kumaripriya.manit@gmail.com)
Akash Zingade(akashzingade@gmail.com)