SIT215 Assignment 1: Search for Agent Navigation

Before you start

Due: April 6 2023, 8:00 pm (AEST)

This assignment contributes 20% to your final mark. Read the assignment instructions carefully.

Work in teams

In this assignment, you are encouraged to work in teams and to learn from your peers. You may form a "group" with up to 5 students. However, each of you needs to make a **separate submission** documenting your individual effort and learning, which will be **individually marked**. Highly similar reports will be investigated to ensure no plagiarism occurs.

What to submit

By the due date, you are required to submit the following files to the corresponding Assignment (Dropbox) in CloudDeakin:

- A brief report explaining the approach taken, any challenges faced during implementation, and any improvements or extensions made to the project. You may structure your report in the following sections:
  - Introduction and problem description
  - Approach and implementation
  - Results and analysis
  - Conclusions and lessons learned
  - Acknowledgement of any external assistance, if applicable
  - References
- A code file of your implementation `[YourID]_assignment1_solution.ipynp` or `[YourID]_assignment1_solution.py`. If you submit an IPython notebook, you also need to submit the HTML export, named `[YourID]_assingment1_output.html`.
- Extra files to complete your assignment, if any (e.g., input files used to reproduce your answers).

Please keep your report short and to the point. Clean up your code outputs to reduce unnecessary information (e.g., excessively long logs).

Marking criteria

Indicative weights of various tasks are provided below, but your submission will be marked by the following criteria, adjusting for the overall quality.

P-level expectation

- Showing good effort through completed tasks.
- Applying unit learning to design suitable solutions for the tasks.

C-level expectation

- Showing attention to details through a good quality assignment report.

D-level expectation

- Demonstrating creativity and resourcefulness in providing unique solutions or unique contributions to a joint solution.
- Critically evaluating and reflecting on the pros and cons of various design decisions.

HD-level expectation

- Extending classroom learning to research and tackle previously unexplored theoretical questions or novel applications

(Warning: Highly similar solutions will be investigated for collusion.)

Your Task

In this project, you will implement the Depth-First Search (DFS) algorithm to help a non-playable character (NPC) navigate a maze. The maze is represented as a 2D grid, with walls represented as blocks and open paths represented as empty spaces. The NPC is represented by an ASCII character in the game, and can move in four directions (up, down, left, and right).

Your task is to implement the DFS algorithm to find a path for the NPC to reach the endpoint in the maze. The NPC must navigate around walls to reach the endpoint. You will also need to implement a visualization of the NPC's movement in the maze.

Preparing and Loading Input File (P Task)

Create an input file containing the maze and the NPC and end positions, with walls represented by blocks and open paths represented by empty spaces. An example maze is shown below, with S (starting point) and E (endpoint).

```
##########
#        #
# ###### #
# #   E# #
# # ###  #
# #   #  #
# ### # ##
#   # #  #
# ### #S #
#        #
##########
```

Reads the input file. Identifies the start and end positions of the NPC and the end point of the maze.

Implementing DFS Algorithm (P Task)

Implement the DFS algorithm to traverse the maze. Return the path from the NPC's starting position to the endpoint of the maze.

Visualization (C/D Task)

Provide a graphical interface to display the maze and the NPC's movement. The NPC's position is represented by a moving character or sprite. Walls and the endpoint are also displayed. The path can be highlighted in a different color than the walls and open paths.

Performance Evaluation (D/HD Task)

Test the program with different mazes of varying sizes and complexity. Evaluate the performance of the DFS algorithm on these mazes. Measures the time taken to find the path and the number of nodes explored.