CS250 - ARTIFICIAL INTELLIGENCE LAB

ASSIGNMENT-3 : 8 Puzzle

Date: January 24, 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 : Solving 8-Puzzle Problem with Search Algorithms

Objective: Implement and analyze the space and time complexities of various search algorithms for solving the 8-puzzle problem.

Problem: The 8-puzzle problem involves arranging eight numbered tiles (1 to 8) in a 3x3 grid with one blank space, starting from a given initial configuration. The goal is to reach the desired final configuration (usually all tiles in ascending order with the blank space in the bottom right corner) by applying a sequence of valid moves. Valid moves involve sliding a tile adjacent to the blank space into the blank position.

Initial State:

0	1	3
4	2	6
7	5	8

Final State:

1	2	3
4	5	6
7	8	0

Algorithms: You are required to implement and analyze the following search algorithms for solving the 8-puzzle problem:

1. **Breadth-First Search (BFS):** Explores all nodes at a given level before moving to the next level.
2. **Depth-First Search (DFS):** Explores one branch completely before backtracking and trying another branch.
3. **Iterative Deepening Depth-First Search (IDFS):** Repeatedly applies DFS with increasing depth limits until the goal is found.
4. **Depth-Limited Depth-First Search (DDFS):** Similar to DFS, but sets a maximum depth limit to avoid infinite loops.

Tasks:

1. **Implementation:**
   - Implement each search algorithm in Python using appropriate data structures (e.g., queues, stacks) to represent the search space and maintain visited states.
   - Calculate the space and time complexity of each algorithm (measured in terms of memory usage and the number of nodes explored) for solving the 8-puzzle problem.
   - Provide clear and concise comments explaining the code and algorithms.
2. **Analysis:**
   - Compare the space and time complexities of different algorithms for the same 8-puzzle problem instances.
   - Discuss the trade-offs between space and time efficiency for each algorithm.
   - Analyze the impact of varying initial configurations and problem sizes on the performance of each algorithm.

Deliverables:

- Submit a Python code file containing your implementations of the search algorithms and analysis scripts.
- Write a report summarizing your findings, including algorithm comparisons, complexity analysis, and discussion of performance variations.

Marking Rubric:

- Implementation (70%): Correctness, efficiency, code clarity, documentation.
- Analysis (30%): Accuracy, insights, comparisons, understanding of complexities.

For any queries regarding this assignment, contact:

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