8-Puzzle AI Project Report

Project Name: 8-Puzzle Solver using AI Algorithms

Introduction

The 8-Puzzle is a classic sliding puzzle consisting of a 3×3 board with eight numbered tiles and one empty space. The objective is to reach the goal configuration from an initial state using valid moves. This problem is widely used to demonstrate and compare AI search techniques.

Objectives

- Solve the 8-Puzzle problem using different AI search algorithms.
- Compare the algorithms based on solution optimality, memory usage, and execution time.

Algorithms Used

- **Breadth-First Search (BFS):** Guarantees the shortest path but consumes high memory.
- **Depth-First Search (DFS):** Uses less memory but does not guarantee an optimal solution.
- **Uniform Cost Search (UCS):** Expands the lowest-cost node and guarantees optimality.
- **A* Search:** Uses heuristics (Manhattan Distance and Misplaced Tiles) to efficiently find optimal solutions.
- **Greedy Best-First Search:** Uses heuristics only to guide the search; fast but not optimal.

Implementation Overview

The project was implemented using Python. Each algorithm was implemented in a separate file, while shared helper functions such as state validation, move generation, and goal checking were placed in a utils module.

Project Structure

bfs.py, dfs.py, ucs.py, a_star.py, greedy.py, puzzle.py, utils.py, README.md, report.pdf

Performance Comparison

- **BFS:** Finds the shortest path but has very high memory consumption.
- **DFS:** Uses less memory but may take longer and produce non-optimal solutions.
- **UCS:** Produces optimal solutions but is slower compared to A*.
- **A*:** Provides the best balance between speed and optimality with a good heuristic.
- **Greedy Search:** Very fast and memory efficient but does not guarantee shortest paths.

Conclusion

This project demonstrates the effectiveness of AI search algorithms in solving the 8-Puzzle problem. Among the implemented methods, A* achieved the best overall performance by combining speed and optimality. The results emphasize the importance of choosing the appropriate search strategy based on problem constraints.