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 3x3 board with 8 numbered tiles and one empty space. The objective is to reach the goal configuration using valid moves.

Objectives

- Solve the 8-Puzzle using different AI search algorithms.
- Compare performance in terms of steps, 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 optimal solution.
- Uniform Cost Search (UCS): Expands lowest-cost node and guarantees optimality.
- A* Search: Uses heuristics (Manhattan Distance & Misplaced Tiles) to efficiently find optimal solutions.
- Greedy Best-First Search: Uses heuristic only to choose the next node, faster but not optimal.

Project Structure

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

Performance Comparison

- BFS: Shortest path, very high memory usage.
- DFS: Lower memory usage, may take long time and non-optimal solution.
- UCS: Optimal solution but slower compared to A*.
- A*: Fastest and optimal when using a good heuristic.
- Greedy Search: Very fast, uses less memory, but does not guarantee shortest path.

Conclusion

This project demonstrates the effectiveness of AI search algorithms in solving the 8-Puzzle problem. A* achieved the best balance between speed and optimality, while Greedy Search showed high speed but lacked optimal solutions. The comparison highlights the importance of choosing the right algorithm.