

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 goal is to move tiles to reach a target configuration.

Objectives:

- Solve the 8-Puzzle using AI algorithms.
- Compare performance in terms of steps, memory usage, and execution time.

Algorithms Used:

1. Breadth-First Search (BFS):

- Explores all nodes at the current depth before moving deeper.
- Guarantees the shortest solution.

2. Depth-First Search (DFS):

- Explores as deep as possible along each branch before backtracking.
- May not find the shortest path and can use less memory.

3. Uniform Cost Search (UCS):

- Expands the node with the lowest path cost.
- Guarantees optimal solution if cost is uniform.

4. A* Search:

- Uses heuristics to estimate the cost to goal.
- More efficient; finds the optimal solution faster.
- Common heuristics: misplaced tiles, Manhattan distance.

Project Structure:

8-Puzzle-AI/

- bfs.py
- dfs.py
- ucs.py
- a_star.py
- puzzle.py
- utils.py
- report.pdf
- summary.pdf

Performance Comparison:

- BFS: shortest path, high memory usage.
- DFS: may take longer, less memory.
- UCS: optimal, slower than A*.
- A*: fastest, optimal with good heuristic.

Conclusion:

The project successfully solves the 8-Puzzle using multiple AI algorithms. BFS and A* are most efficient in terms of finding the shortest solution, while DFS uses less memory. The report and summary provide clear documentation for understanding the algorithms and their performance.

End of Report