Intro to AI  
Phase 2: Term project mid-status report  
Development results

8 Puzzle Problem Solver Using Algorithms

**1. Define Your Research Question**

The eight puzzle problem can be solved using search algorithms such as BFS, DFS and A\*. Which approach is more efficient solution-wise and computationally?

**2. Introduction**

The 8-puzzle is a sliding puzzle that consists of a frame of numbered square tiles in random order with one tile missing The goal is to shuffle these tiles to get it ready for some original configuration. This puzzle is a well-known test problem in AI, often used to explore state-space search and optimization.

The 8-puzzle is an important topic that enables the problem solver to understand how search algorithms are applied from a computational perspective in the real world, for instance, in robotics navigation, logistics, and automated problem-solving. A GUI-based solver for the puzzle using BFS, DFS and A\* which can be used to visualize how these algorithms work together with their pros and cons in solving.

**3. Related Literature**

**State-Space Search Techniques in AI**:

In this work we taken a look into the theory behind state-space search algorithms like BFS and A\* Yet it falls short on usefulness with visual implementations and a side-by-side behavioral comparison of algorithms.

**The Sliding Puzzle Problem and Heurist**:

PyHeuristics: Concentrates on heuristic functions (e.g., Manhattan distance with A\*). But it never seeks interactivity or visuals that put the user upfront.

**How My Work Differs:**

* A record of developing this project combines BFS, DFS, and A\* algorithms into a creating GUI that makes it natural to collaborate with.
* This also helps visualize the path of the solution and gives a sense of how well or poorly the algorithm performs.
* This implementation exposes the difficulties of each, like memory usage (DFS) and tuning heuristics (A\*).

**4. Achievements, Challenges, and Workarounds**

**Achievements:**

* Using BFS, DFS, and A\* to solve the 8-puzzle.
* Developed a GUI to visualize the solution step-by-step.
* Highlighted the efficiency and shortcomings of each algorithm.

**Challenges and Workarounds:**

* **Challenge:** DFS memory usage for deeper states.  
  **Workaround:** cutoff limits to prevent stack overflow.
* **Challenge:** Optimizing A\* on heuristic for complex configurations.  
  **Solution:** Fine-tuned the Manhattan distance heuristic to improve performance without increasing computational cost.

**5. GitHub Link:**

[**https://github.com/Neeraj1404nani/8-Puzzel-solver-using-search-algorithm**](https://github.com/Neeraj1404nani/8-Puzzel-solver-using-search-algorithm)