**PROJECT REPORT**

**Project Title:**

*Sliding Tile Puzzle with AI Solver (A/BFS Algorithm)*\*

**Submitted By:**  
Hamza Murtaza (22k-4540)  
Muhammad Fasih (22k-4486)  
Riyyan Siddiqui (22k-4452)

**Course:** Artificial Intelligence  
**Instructor:** [Instructor’s Name]  
**Submission Date:** [Date]

**1. Executive Summary**

**Project Overview:**  
This project implements a modified version of the classic Sliding Tile Puzzle, enhanced with an interactive user interface and an AI-based solver using either the A\* or Breadth-First Search (BFS) algorithm. The project focuses on real-time step-by-step visualization of the solving process to improve user understanding and experience.

**2. Introduction**

**Background:**  
The Sliding Tile Puzzle is a classic problem-solving game where tiles on a grid must be rearranged into a specified goal state by moving one tile at a time into an adjacent empty space. This game is often used in AI to test search algorithms. We selected it for its relevance to pathfinding and state-space search concepts, and modified it by adding real-time solving visualizations and improved user interaction.

**Objectives of the Project:**

* Develop an interactive UI for manual and automatic gameplay.
* Implement A\* and BFS algorithms to solve the puzzle.
* Provide visual, step-by-step representation of AI’s decision-making.
* Allow the user to switch between different puzzle sizes and reset functionality.

**3. Game Description**

**Original Game Rules:**  
The traditional game involves a grid of numbered tiles with one empty space. The objective is to rearrange the tiles into a specific order by sliding adjacent tiles into the empty space.

**Innovations and Modifications:**

* AI Solver using A\* or BFS algorithm.
* Visual, step-by-step representation of each AI move.
* Interactive UI with mouse click functionality.
* Support for multiple grid sizes like 3x3 and 4x4.
* Restart and randomization features.

**4. AI Approach and Methodology**

**AI Techniques Used:**  
We employed both A\* (A-Star) and BFS algorithms to explore the state space and solve the puzzle optimally or efficiently.

**Algorithm and Heuristic Design:**

* **A\***: Uses Manhattan distance as the heuristic to estimate the cost to reach the goal state.
* **BFS**: Explores all possible moves in a breadth-first manner to find the shortest solution path.
* States are represented as matrices; valid moves generate new states, which are explored using priority queues (A\*) or FIFO queues (BFS).

**AI Performance Evaluation:**

* Win rate is 100% for solvable puzzles.
* Average decision time for A\* is under 1 second for 3x3 puzzles.
* BFS performs well on smaller puzzles but slows down significantly for larger grids due to memory constraints.

**5. Game Mechanics and Rules**

**Modified Game Rules:**

* Player or AI can only slide tiles adjacent to the empty space.
* Multiple puzzle sizes (3x3, 4x4) are available.
* AI can be triggered at any time to solve the current configuration.
* The user can restart or shuffle the puzzle manually.

**Turn-based Mechanics:**  
The player can manually click tiles to move them. The AI plays its turn in a sequential step-by-step process upon request.

**Winning Conditions:**  
The game ends when the puzzle is arranged in the target order (e.g., ascending numbers from left to right, top to bottom, with the blank tile at the end).

**6. Implementation and Development**

**Development Process:**  
The project was implemented using Node.js for backend logic and integrated with a browser-based front end for visualization. Puzzles were encoded as matrices, and the AI solver executes on demand.

**Programming Languages and Tools:**

* **Language**: JavaScript (Node.js)
* **Libraries**: HTML5 Canvas, Node modules for state management
* **Tools**: GitHub (version control), VS Code

**Challenges Encountered:**

* Implementing efficient visualization without affecting performance.
* Managing state representation and ensuring proper backtracking.
* Preventing unsolvable puzzle generation.

**7. Team Contributions**

* **Hamza Murtaza (22k-4540)**: AI algorithm development (A\*, BFS), back-end logic
* **Muhammad Fasih (22k-4486)**: Game rule implementation, puzzle state design
* **Riyyan Siddiqui (22k-4452)**: UI development and integration, front-end visualization

**8. Results and Discussion**

**AI Performance:**

* A\* performs consistently well across multiple puzzle sizes.
* BFS is more memory-intensive but guarantees shortest paths for smaller puzzles.
* Visual feedback significantly improves understanding of AI steps.
* In tests, AI solved 3x3 puzzles in under 2 seconds with smooth animations.

**9. References**

* Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach*.
* GeeksforGeeks: A\* Algorithm & BFS in AI
* StackOverflow Discussions on Sliding Puzzle State Encoding
* GitHub Repos for Puzzle Game UIs