Project Title: AI Maze Solver Using Search Algorithms

Submitted By:

- M. Yousuf Rehan (22K-4457)
- Syed Azmeer Un Nabi (22K-4235)

Course: Artificial Intelligence
Instructor: [Instructor's Name]
Submission Date: 10th May 2025

---

1. Executive Summary

Project Overview:
In this project, we set out to build an intelligent maze solver. The idea was simple: create a maze, drop in an agent, and teach it how to get from point A to point B. But instead of giving it a map, we gave it brains—algorithms like Depth-First Search (DFS), Breadth-First Search (BFS), and A* to figure things out. Our main goal was to explore how these different algorithms behave when put to the test in a grid-based maze and visually observe how each one performs.

---

2. Introduction

Background:
Everyone has tried to solve a maze at some point—on paper, in a game, or even metaphorically. In AI, maze solving is a classic problem that tests how well algorithms can navigate and make decisions. We chose this project to explore how search algorithms work in practice, not just theory. Rather than using a fixed, step-by-step method, we let the AI decide its own path.

Objectives of the Project:

- Bring mazes to life with AI-driven decision-making.
- Implement and test DFS, BFS, and A* algorithms.
- Compare their logic, efficiency, and output.
- Create a user-friendly interface to visualize the agent's journey.

---

3. Game Description

Original Game Rules:
The maze is structured as a grid. There is a start point, a goal point, and lots of obstacles in between. The agent can move one step at a time—up, down, left, or right—but can't go through walls.

Innovations and Modifications:
We designed our own maze and introduced three different solving styles via AI. We didn't just want to get from start to goal—we wanted to do it smartly, and differently each time, depending on the algorithm in use.

---

4. AI Approach and Methodology

AI Techniques Used:

- **DFS** goes deep and explores one path until it hits a wall.
- **BFS** spreads out evenly and finds the shortest route, eventually.
- **A*** combines the best of both worlds by thinking ahead with a smart estimate (Manhattan Distance) to guide its decisions.

Algorithm and Heuristic Design:
For A*, we used the formula: **f(n) = g(n) + h(n)**

- **g(n)** is the actual path taken so far.
- **h(n)** is the guess on how much farther to go.
  This helped our agent make balanced decisions between speed and accuracy.

AI Performance Evaluation:
We didn't just run the algorithms—we watched them. We measured how fast they reached the goal and how long their path was. Some were quick but clumsy; others took their time and nailed the shortest route.

---

5. Game Mechanics and Rules

Modified Game Rules:

- The maze grid has static walls (can't pass).
- The agent can move in four directions.
- No diagonal moves or backtracking to the same spot unless necessary.

Turn-based Mechanics:
The agent thinks, moves one step, thinks again, and so on. It doesn't guess randomly—it calculates and moves.

Winning Conditions:
If the agent reaches the goal square, it wins. The faster and shorter the path, the better.

---

6. Implementation and Development

Development Process:
We used Python for everything. The maze layout, the AI logic, and the game window were all built using Pygame. Each algorithm was written separately, tested on the same maze, and then integrated with visuals so we could watch the magic happen in real time.

Programming Languages and Tools:

- **Language:** Python
- **Libraries:** Pygame, heapq (for priority queues in A*)
- **Tools:** GitHub for code collaboration and version control

Challenges Encountered:

- DFS sometimes got stuck in loops; we fixed it with visited checks.
- A* took a while to tune right. Choosing the right heuristic matters.
- The GUI froze when too many calculations were done too fast, so we added delays and frame updates.

---

7. Team Contributions

- **M. Yousuf Rehan:** Took charge of implementing DFS and BFS, worked on the basic maze layout and ensured smooth grid rendering.
- **Syed Azmeer Un Nabi:** Focused on A* search, built the heuristic, and led the GUI animations to show step-by-step decision-making.

---

8. Results and Discussion

AI Performance:
Here's how each algorithm performed on a 20x20 maze:

- **DFS:** Fast but sometimes took longer routes.
    - Time: ~0.12s
    - Path Length: Varies
- **BFS:** Always found the shortest path but took more time.
    - Time: ~0.25s
    - Path Length: Optimal
- **A*:** Balanced approach, performed best overall.
    - Time: ~0.18s
    - Path Length: Optimal

Watching the A* agent navigate felt like watching a smart player think ahead.

---

9. References

- Russell, S., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*
- Python Documentation: https://docs.python.org/3/
- Pygame Library: https://www.pygame.org/wiki/about
- Various GitHub examples and blog posts on pathfinding and heuristics