

MAZE SOLVER

INTRODUCTION

The Maze Solver project is an application that integrates efficient pathfinding algorithms with an intuitive graphical user interface (GUI). Developed in C using the GTK library, this project demonstrates the implementation of A* and Dijkstra's Algorithm for solving mazes of varying sizes. It also incorporates login and registration functionality. This report outlines the project's objectives, methodology, implementation, and future scope.

PROJECT MOTIVATION

This project is chosen because it lets the fundamental concepts in computer science be applied in an extremely applicable and visually appealing manner. In maze generation and solving, key algorithms come into play, making it a perfect project for an Introduction to Computer Science course. The visual nature of mazes also lends their results to tangibility and accomplishment, which could help in bettering one's understanding of challenging algorithms.

KEY FEATURES

1. **LOGIN AND REGISTRATION:** using concept of file handling for account registration and login
2. **CUSTOMISABLE MAZE GENERATION:** option to input custom maze size
3. **INTERACTIVE GUI:** built using GTK
4. **MAZE SOLVING:** using Dijkstra and A*
5. **COMPARISON:** using runtimes of both the algorithms

SCREENSHOTS OF OUTPUT

Login and register

Maze Solver

Username

Password

Login

Don't have an account?

Register

Maze Solver

Username

Password

Register

Back to login

Maze Generation and Maze Solving

Maze Solver

Welcome, admin!

Logout

Maze Size: 100

Dijkstra Algorithm

Generate Maze

A Star Algorithm

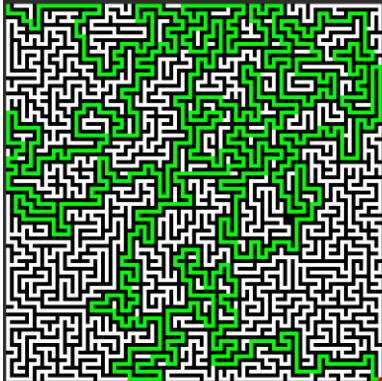
Time: Comparatively slower, explores all paths.
Complexity: $O((V + E) \log V)$
Path: Finds the shortest path, but may explore unnecessary paths.
Memory consumption: It is slightly more memory efficient.
Application: Suitable when there is no goal or when exploring all nodes is necessary.



Time: Comparatively Faster, uses a heuristic to prioritize paths.
Complexity: $O((V + E) \log V)$
Path: Finds the shortest path with a good heuristic, but can be longer with a bad one.
Memory consumption: A* stores information about each node. In big mazes, this can lead to significant memory usage.
Application: Ideal when a goal is known, as the heuristic helps focus the search toward the goal.

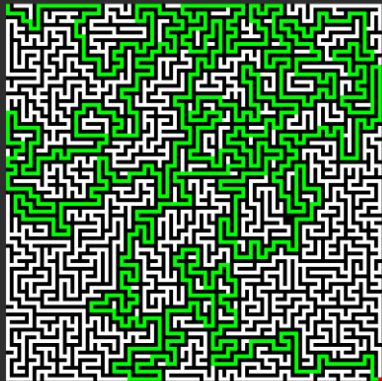
Solve Maze Using Dijkstra Algorithm

The program executes in 100000.000000000000 microseconds and path length is 1828 units using Dijkstra



Solve Maze using A Star Algorithm

The program executes in 360000.000000000000 microseconds and path length is 1828 units using A Star



METHODOLOGY

1. Algorithms

1.1 A* Algorithm: A heuristic-based approach using a priority queue for shortest-path calculation. Combines cost from the start (g-cost) and estimated cost to the goal (h-cost).

1.2 Dijkstra's Algorithm: Explores all possible paths from the start to the goal and guarantees the shortest path, Operates without heuristics

2. GUI Development: GTK (GIMP Toolkit) was used to build a cross-platform GUI. Features include Login and registration windows, Maze grid visualization.

3. File Handling: User credentials are stored in a file for authentication.

CHALLENGES

Integrating GUI into the application was a challenging task, considering there were very few tutorials for GTK available online and we had to go through the documentation. Making sure that the UI works hand-in-hand with the backend, generating arrays for mazes using pointers and heap memory also took effort. Tackling new algorithms and understanding them also was a challenging task.

FUTURE SCOPE

1. Add support for additional algorithms like Greedy First Search.
2. Develop a built-in maze editor for creating custom mazes.
3. Enable saving and loading mazes.
4. Improve visualization with animations and sound effects.
5. Integrate user profiles for personalized experiences.

CONCLUSION

The Maze Solver project has been a rewarding journey, blending algorithm design with GUI development. Implementing **A*** and **Dijkstra's Algorithm** enhanced our problem-solving and data structure skills, while working with GTK deepened our understanding of creating user-friendly applications.

This project was more than just solving mazes, it was about navigating challenges, from debugging to designing intuitive interfaces. Each step, though sometimes frustrating, brought the joy of creation and reinforced my passion for computer science.

Proud of what we've built, we see this as both a milestone and a stepping stone toward a future of learning and innovation.

Contributions

Akhil Dhyani: GUI development, backend integration

Saksham Panghal: Maze generation

Divyansh Yadav: Maze solving using Dijkstra's Algorithm

Harshil Agrawal: Maze solving using A* Algorithm