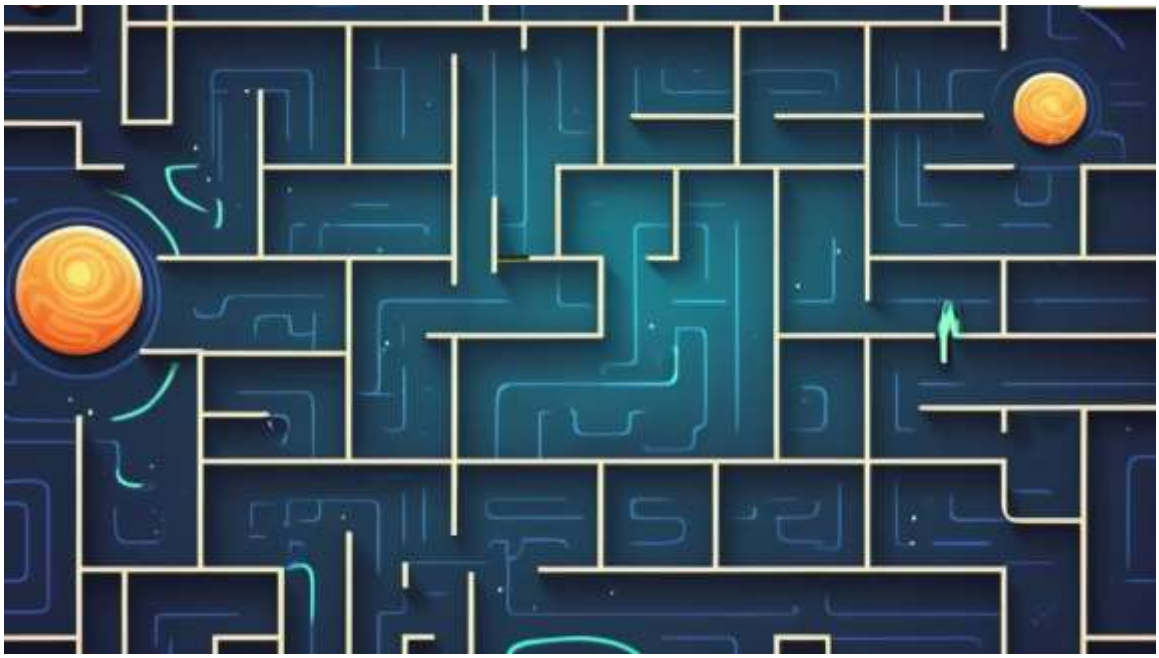


Artificial Intelligence Project

MAZE SOLVING ALGORITHM

TEAM MEMBERS

ABDULLAH TAREEN - 210690

AYESHA SHAHZAD - 210680

AFSHEEN ABRAR - 210697

Artificial Intelligence Project: Maze Solver

Introduction

This project demonstrates a Maze Solver program implemented in Python using the Pygame library. The program ***generates a maze using a randomized depth-first search algorithm*** and ***solves it using a depth-first search algorithm.***

Features

1. Maze generation using randomized depth-first search.

2. Maze solving using depth-first search.

3. Visual representation of the maze and the solving process.

4. Dynamic maze dimensions and customizable visualization settings.

Algorithm Details

Maze Generation

The maze generation algorithm employs a randomized depth-first search approach. Starting from a random cell, it explores unvisited neighbors recursively, creating paths by linking cells. If no unvisited neighbors are available, it backtracks to the previous cell and continues until all cells are processed.

Maze Solving

The maze solving algorithm uses depth-first search. Starting from the top-left cell, it recursively explores paths towards the bottom-right cell, marking visited cells to avoid revisits. The solution path is stored upon reaching the target and highlighted as Blue Dots on the screen, while Greyed area is considered as the visited places.

Code Explanation

The program is structured into classes and functions for modularity and readability. Key components include the Cell class representing individual maze cells, the Maze class managing maze generation and solving, and the main function for executing the program.

Highlights of the code include:

- The Cell class encapsulates attributes and methods for managing individual cells, such as neighbors and visit status.

- The Maze class implements the logic for maze generation, solving, and visualization using Pygame.

- The main function initializes the maze and handles program execution.

Visualization

Pygame is used for rendering the maze, solution path, and visited cells. Colors are used to differentiate between walls, paths, and solution steps, providing a clear and interactive visualization.

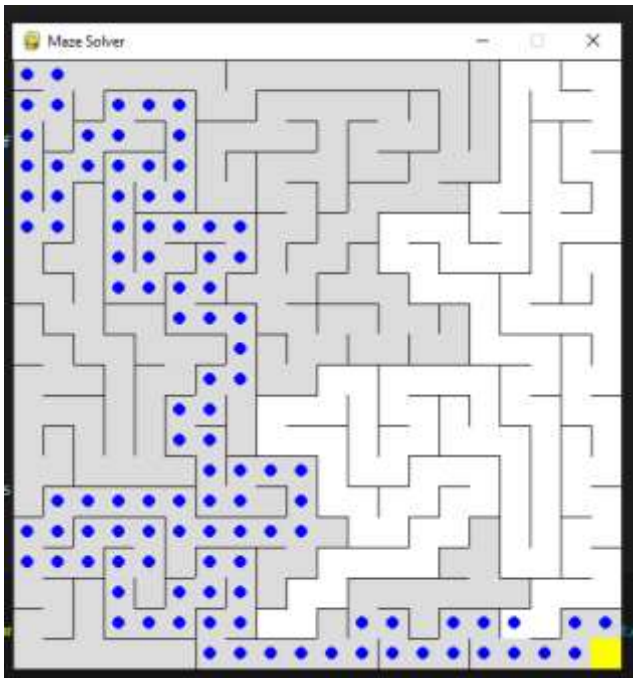
Customization

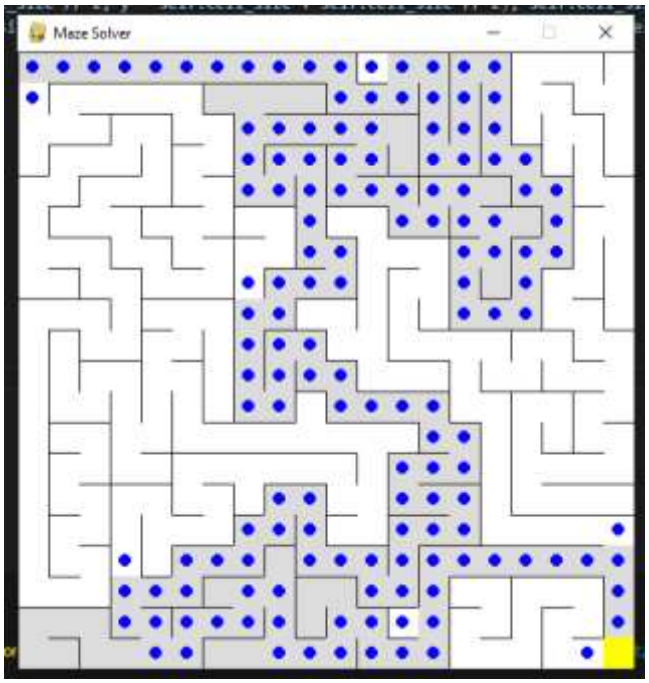
The program allows customization of maze dimensions, cell sizes, and update intervals for visualization. These can be adjusted by modifying the corresponding parameters in the code.

Conclusion

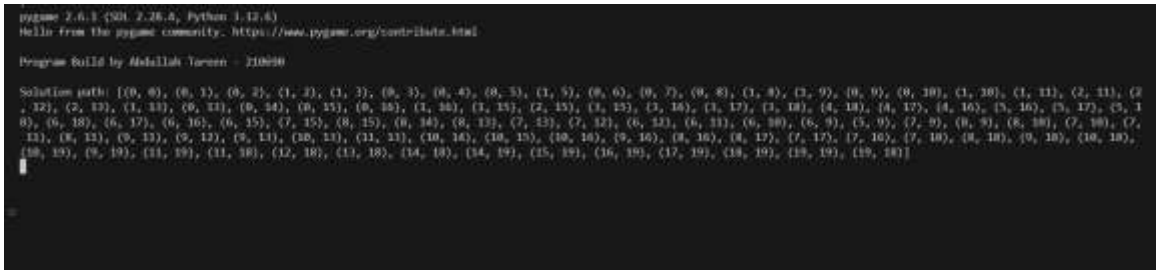
The Maze Solver program is a practical demonstration of algorithms and visualization techniques in Python. It showcases efficient problem-solving approaches and dynamic rendering capabilities using Pygame.

Visualization:





Console Output:



Prerequisites:

1. npm install Python
2. pip install random
3. pip install pygame

And to run the code:
Python <filename>