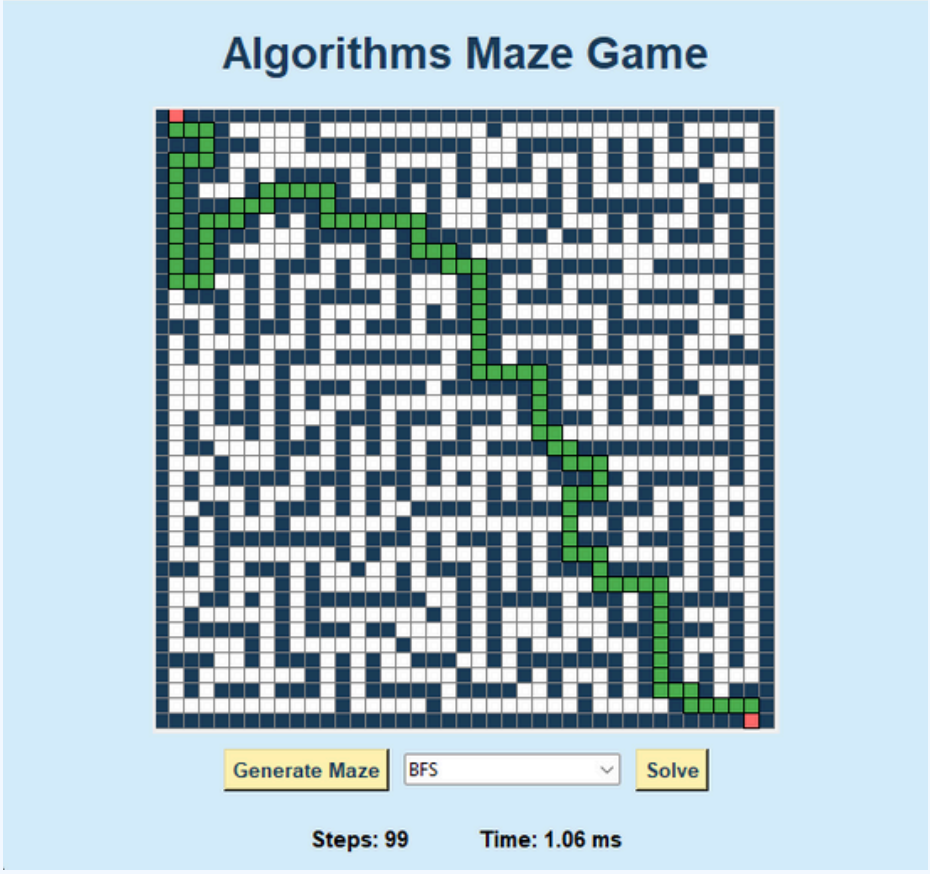
Project

Algorithms Maze Game
Intelligent Agent and Search Problem



STUDENT NAME : Hazem Mohamed Ali Mohamed

ID: 23014905

STUDENT NAME : Bola Eshak Faried Wahba

ID: 23013133

Course : Artificial Intelligence

Project Type : Intelligent Agent & Search Algorithms

Introduction

Purpose of the Project

The purpose of this project is to design and implement an intelligent agent that can solve a maze navigation problem using classical AI search algorithms. The agent perceives its environment, takes actions, and searches for an optimal path from a start state to a goal state.

Problem Overview

Maze navigation is a well-known search problem in Artificial Intelligence, where an agent must find a path through a grid-based environment containing obstacles. This problem is suitable for demonstrating state-space representation, problem formulation, and algorithmic comparison.

Project Objectives

- Design an intelligent agent capable of navigating a maze
- Formulate the problem using AI principles
- Implement and compare multiple search algorithms
- Visualize the agent's behavior using a graphical interface
- Analyze algorithm performance based on defined metrics

Intelligent Agent Description

Agent Type

The agent implemented in this project is a goal-based intelligent agent. It selects actions based on a predefined goal and uses search algorithms to determine the optimal sequence of actions.

Agent Behavior

- The agent starts from a fixed initial position
- It senses the maze structure (walls and free paths)
- It applies a selected search algorithm
- It moves step by step until the goal state is reached

Environment Interaction

The agent interacts with a grid-based maze environment, where it can move in four directions:

- Up - Down - Left - Right

PEAS Model

Component	Description
Performance Measure	Path length (steps), number of expanded nodes, execution time
Environment	Grid-based maze with walls and free cells
Actuators	Movement actions: Up, Down, Left, Right
Sensors	Maze grid information and agent's current position

ODESDA

Component	Description
Observable	The environment is fully observable because the agent has complete access to the maze layout, including walls and free cells.
Deterministic	The environment is deterministic since each action taken by the agent results in a predictable and consistent state transition.
Episodic - Sequential	The environment is sequential, as each action affects future decisions and outcomes.
Static - Dynamic	The environment is static, meaning that it does not change while the agent is planning or acting.
Discrete - Continuous	The environment is discrete, as the maze consists of a finite number of cells and actions.
Single Agent - Multi Agent	The environment is single-agent, since only one agent operates in the maze.

Environment Classification

Environment Classification

The maze environment can be classified as follows:

- Observable: Fully observable
- Deterministic: Yes
- Episodic: No (sequential)
- Static: Yes
- Discrete: Yes

Problem Formulation

Initial State

The agent starts at the maze entrance located at the top-left inside position of the grid.

State Space

Each state represents the agent's position *(row, column)* within the maze grid.

Actions

The agent can perform the following actions:

- Move Up
- Move Down
- Move Left
- Move Right
- (Only if the target cell is not a wall)

Transition Model

Each action moves the agent from one cell to an adjacent free cell.

Goal Test

The goal is reached when the agent arrives at the maze exit located at the bottom-right inside position.

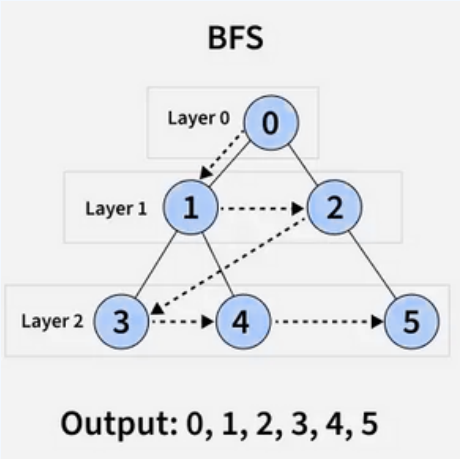
Path Cost

Each move has a uniform cost of 1.
Total path cost = number of steps taken.

Implemented Search Algorithms

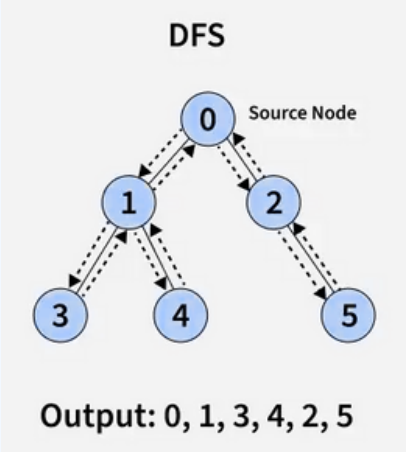
Breadth-First Search (BFS)

- Explores the state space level by level
- Guarantees the shortest path in unweighted graphs
- Expands many nodes



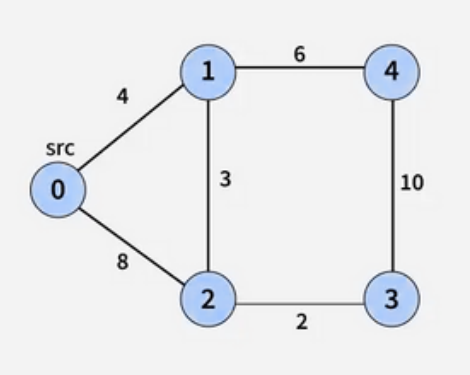
Depth-First Search (DFS)

- Explores deeply before backtracking
- Does not guarantee the shortest path
- Uses less memory



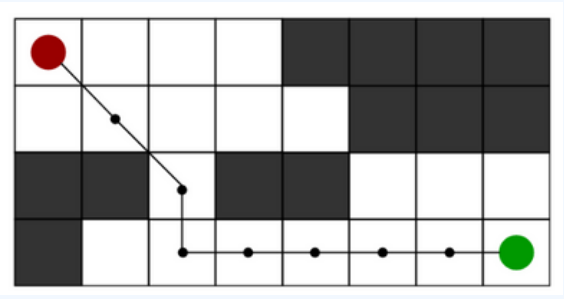
Uniform Cost Search (Dijkstra)

- Expands nodes based on cumulative path cost
- Guarantees optimal solution
- Equivalent to BFS when all costs are equal



A* Search

- Uses both path cost and heuristic
- Heuristic: Manhattan Distance
- Efficient and optimal



Graphical User Interface (GUI)

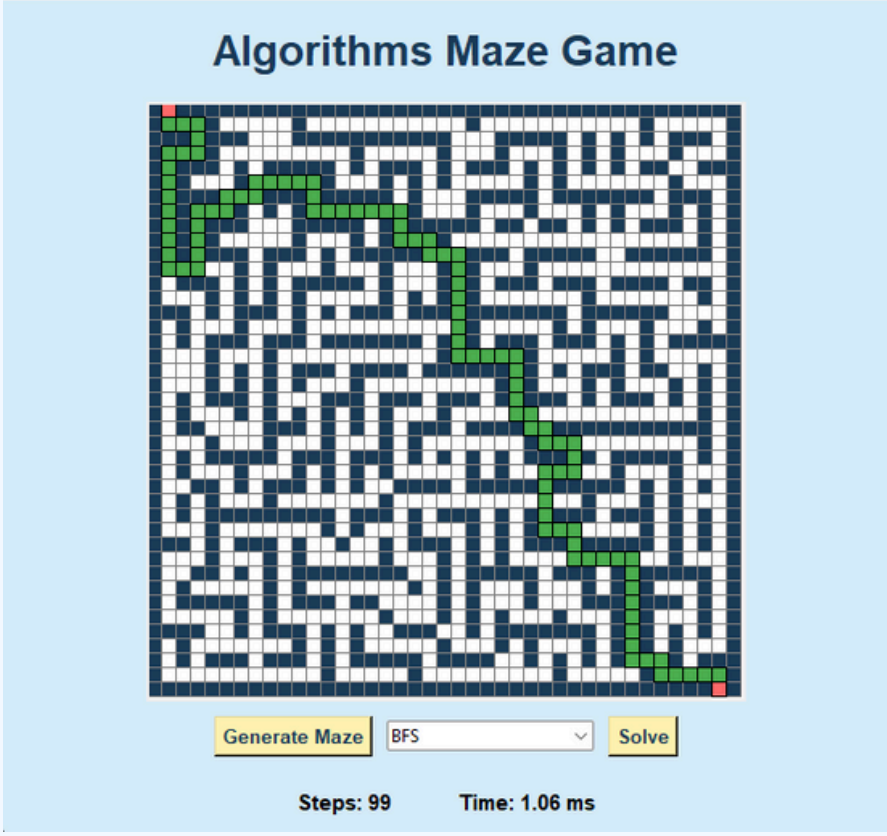
Interface Description

A graphical user interface was developed using Tkinter to visualize:

- Maze generation
- Algorithm execution
- Path traversal
- Performance metrics

Interface Features

- Algorithm selection menu
- Maze generation button
- Solve button
- Real-time display of steps and execution time
- Color-coded path visualization

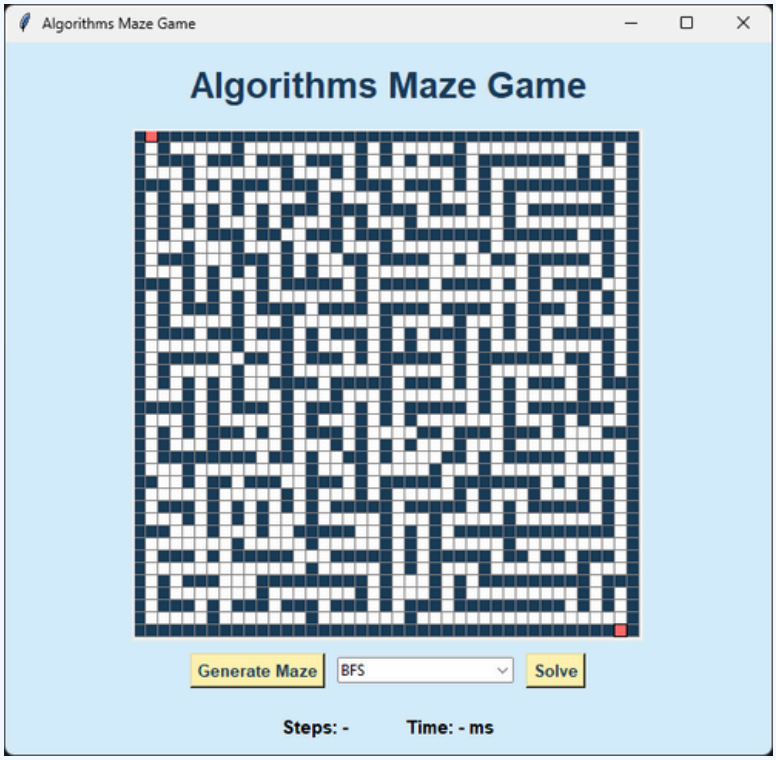


Experimental Results

Performance Metrics

The algorithms were evaluated based on:

- Path Length (Steps)
- Execution Time (ms)



Sample Results Table

Algorithm	BFS	DFS	UCS	A*
Path Length	89	229	89	89
Time (ms)	0.83	0.36	1.28	0.87
The Maze				

Final Thoughts

Discussion

- BFS and UCS guarantee optimal paths but may expand many nodes
- DFS is faster in some cases but produces longer paths
- A* offers the best balance between speed and optimality
- Execution time may vary slightly due to system conditions
- Path length is the most reliable performance metric

Conclusion

This project successfully demonstrates the application of intelligent agents and search algorithms in solving a maze navigation problem. The implementation highlights the strengths and weaknesses of different search strategies and provides a clear visualization of agent behavior.

The use of a graphical interface enhanced understanding and made algorithm comparison intuitive and effective.

Thank You!