AI Pathfinder: Comprehensive Report

Roll Numbers:

- 22F-3347
- 22F-XXXX

Course: AI 2002 – Artificial Intelligence (Spring 2026)

Assignment: Assignment 1, Question 7

Date: February 2026

Table of Contents

---

1. Introduction

This report presents a comprehensive implementation of **six uninformed search algorithms** featuring real-time GUI visualization and dynamic obstacle handling. The project demonstrates how different search strategies explore a grid environment to navigate from a start point to a target.

Objectives

- Implement all 6 uninformed search algorithms.
- Provide step-by-step visualization of the search process.
- Handle dynamic obstacles appearing during runtime.
- Compare algorithm performance across diverse scenarios.

---

2. Implementation Overview

Technology Stack

- **Language**: Python 3.11
- **GUI Framework**: Pygame 2.6.1

- **Additional Libraries**: NumPy

Key Features

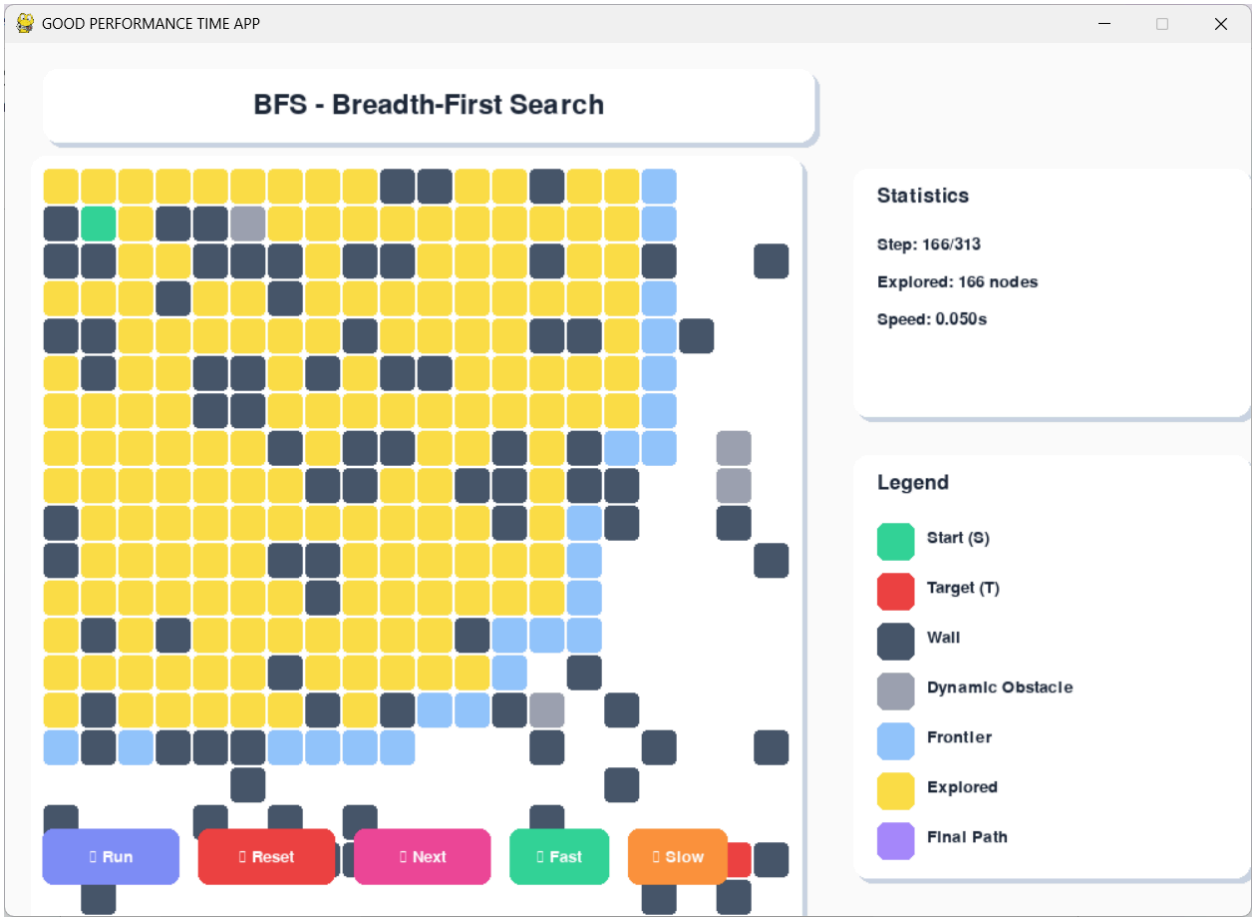
- **8-Directional Movement**: Includes cardinal and diagonal steps.
- **Dynamic Obstacles**: Random obstacles spawn with a 2% probability per step.
- **Interactive Controls**: Real-time algorithm selection and speed adjustment.

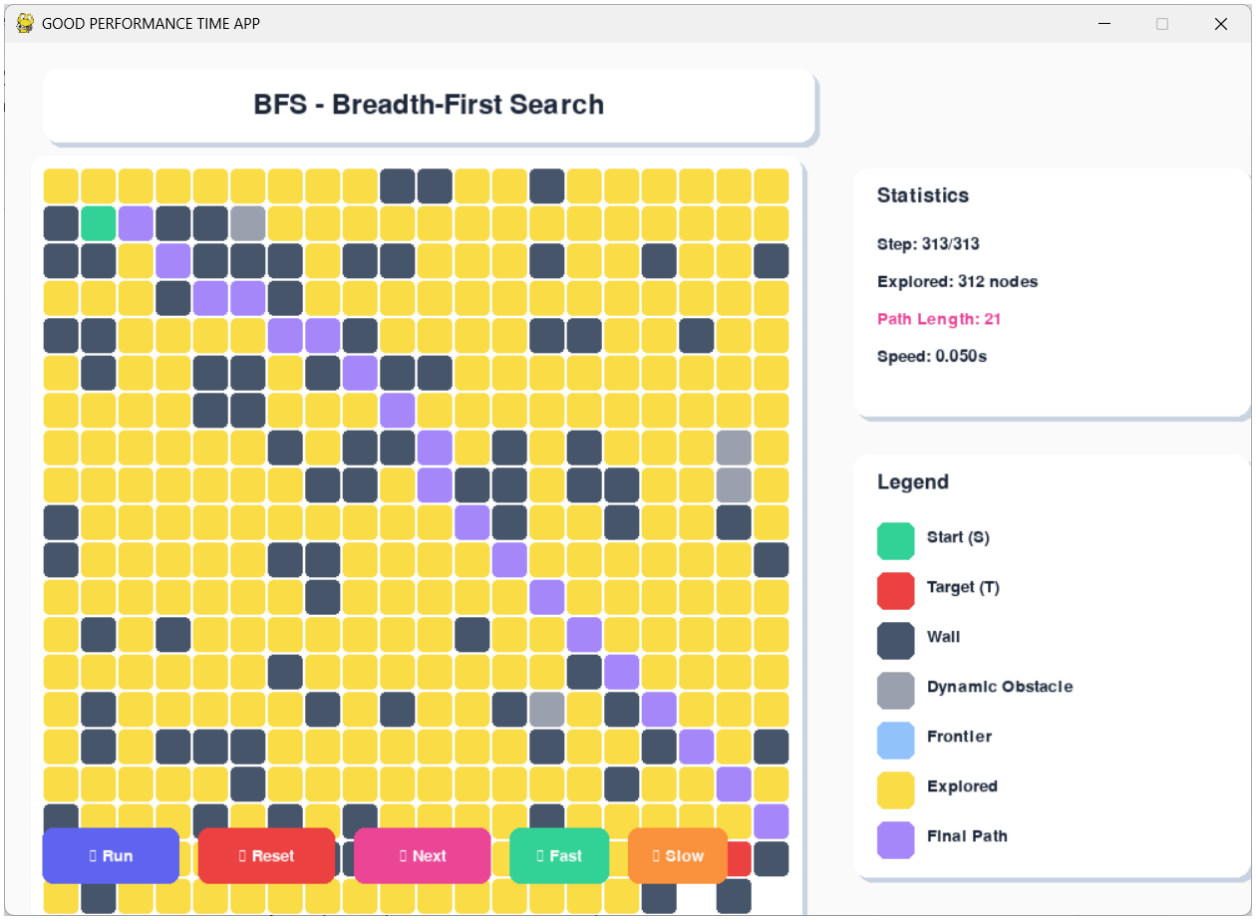
---

3. Algorithm Implementations

3.1 Breadth-First Search (BFS)

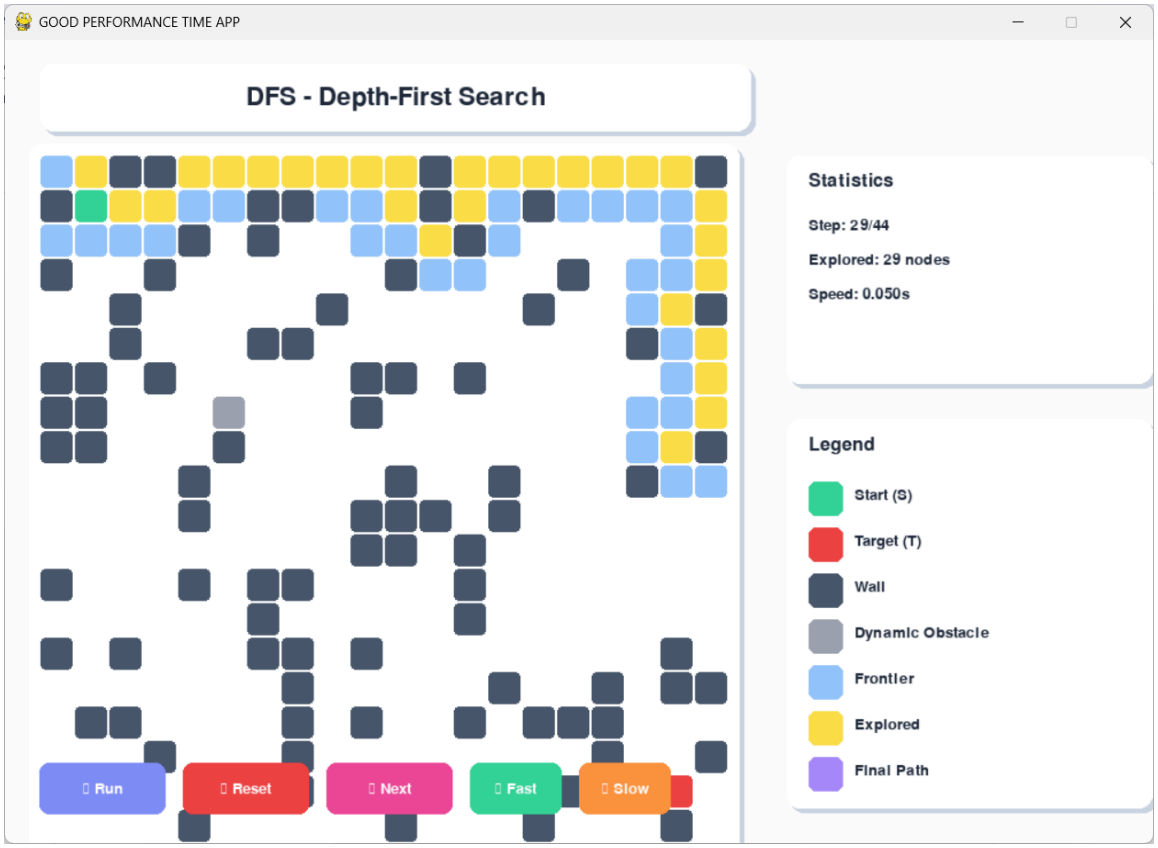
- **Logic**: Uses a **FIFO Queue**. Explores level by level.
- **Pros**: Guaranteed shortest path in unweighted graphs; Complete.
- **Cons**: High memory consumption (O(b^d)).

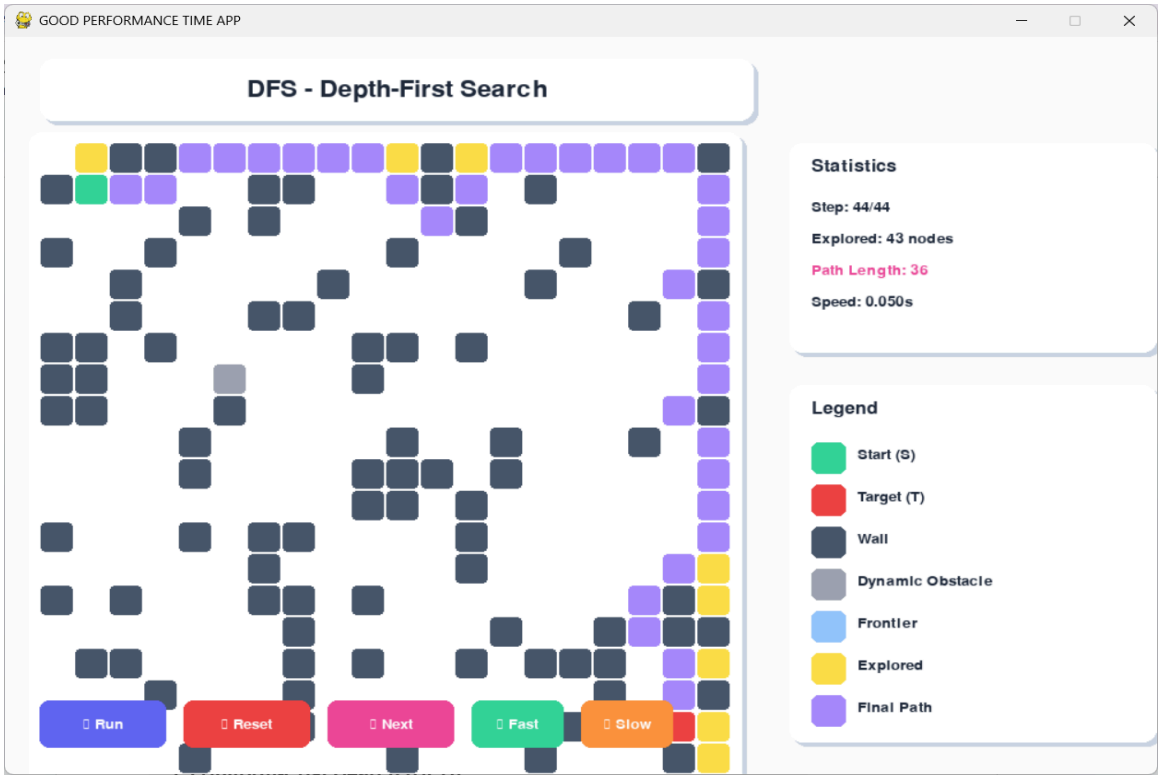




3.2 Depth-First Search (DFS)

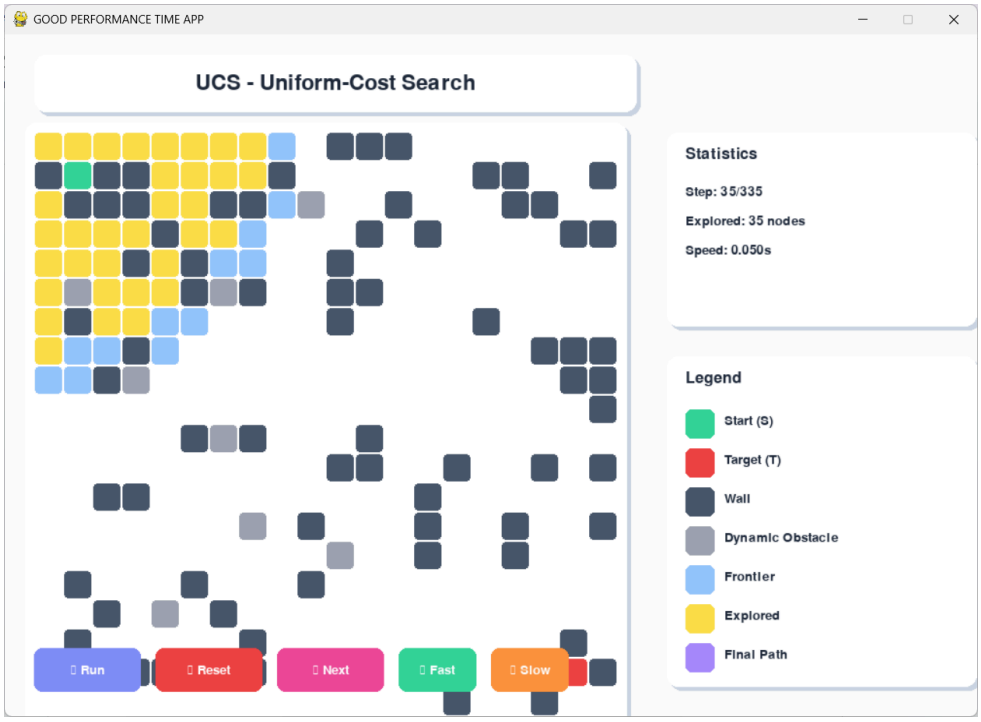
- **Logic**: Uses a **LIFO Stack**. Explores as deep as possible.
- **Pros**: Low memory overhead (O(bm)).
- **Cons**: Not optimal; can get trapped in deep branches.

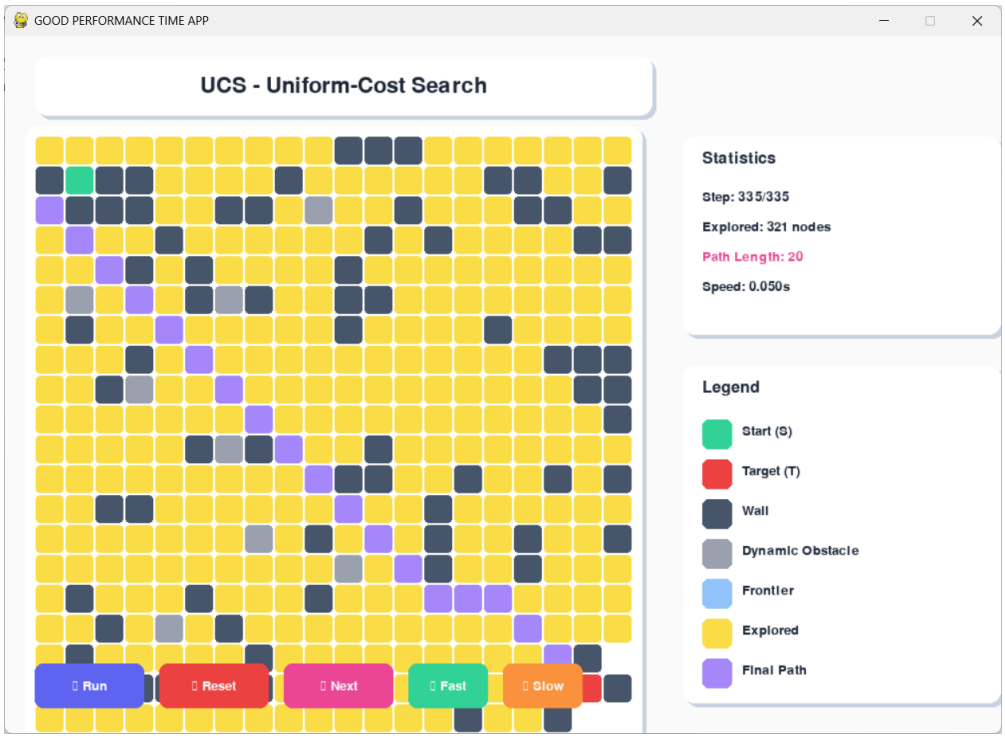

GOOD PERFORMANCE TIME APP
DFS - Depth-First Search
Statistics
Step: 29/44
Explored: 29 nodes
Speed: 0.050s
Legend
Start (S)
Target (T)
Wall
Dynamic Obstacle
Frontier
Explored
Final Path
Run
Reset
Next
Fast
Slow


GOOD PERFORMANCE TIME APP
DFS - Depth-First Search
Statistics
Step: 44/44
Explored: 43 nodes
Path Length: 36
Speed: 0.050s
Legend
Start (S)
Target (T)
Wall
Dynamic Obstacle
Frontier
Explored
Final Path
Run
Reset
Next
Fast
Slow

3.3 Uniform-Cost Search (UCS)

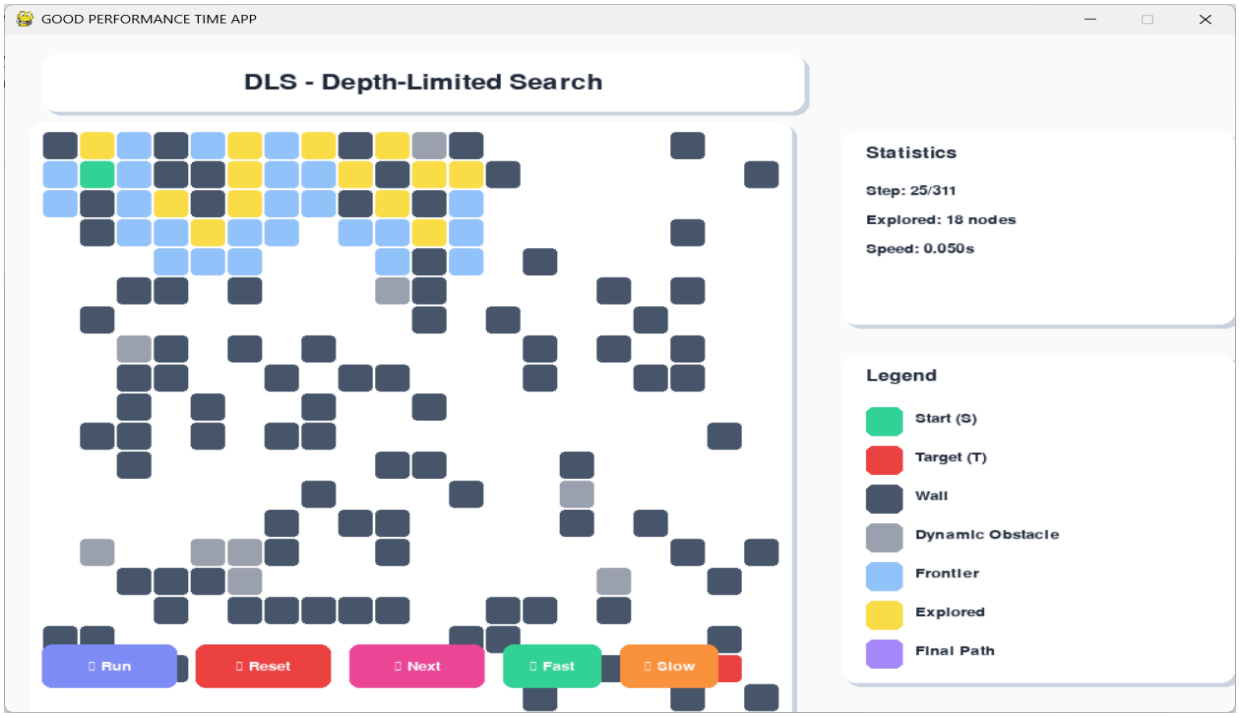
- **Logic**: Uses a **Priority Queue**. Diagonal moves cost $\sqrt{2} \approx 1.41$.
- **Pros**: Optimal for weighted edges.
- **Cons**: Higher computational overhead due to constant sorting.

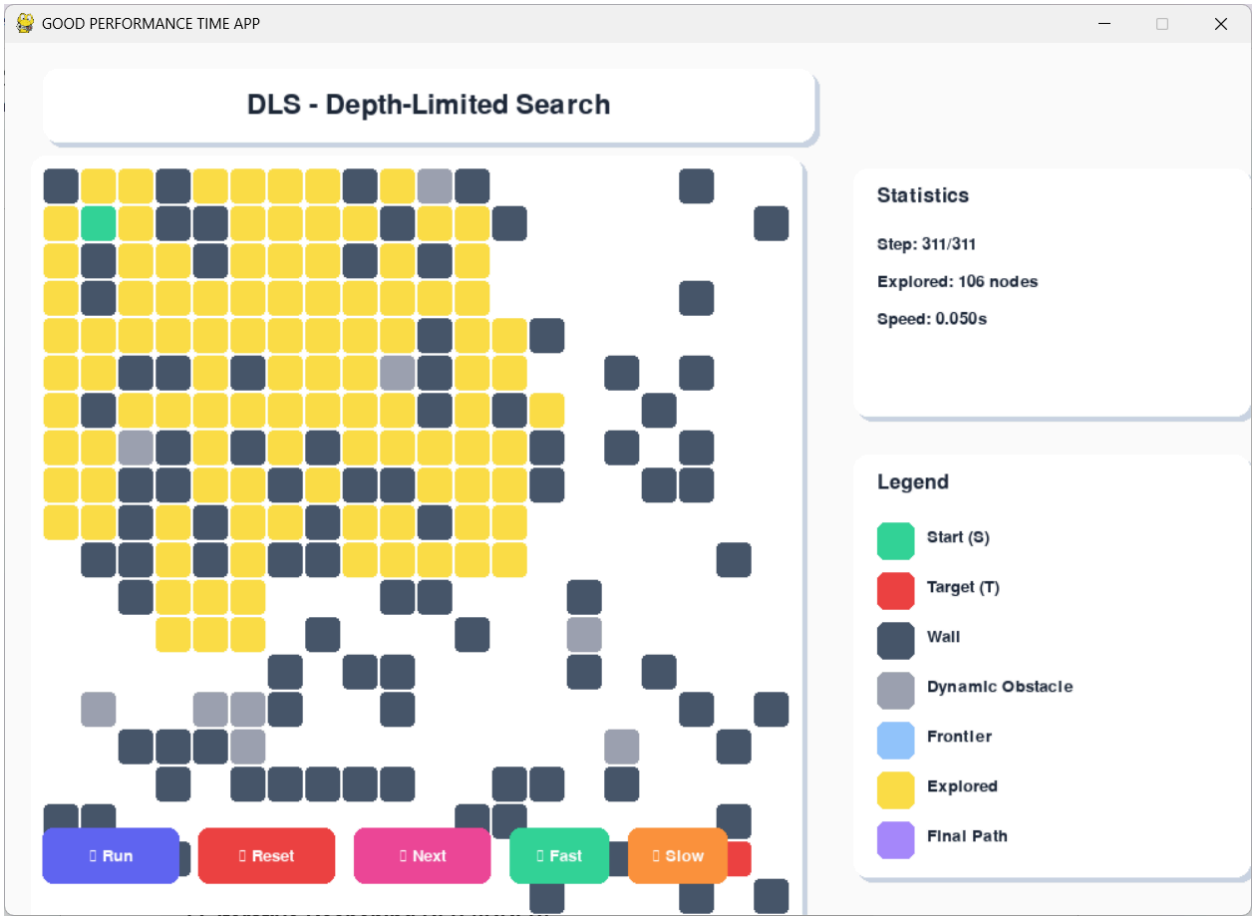




3.4 Depth-Limited Search (DLS)

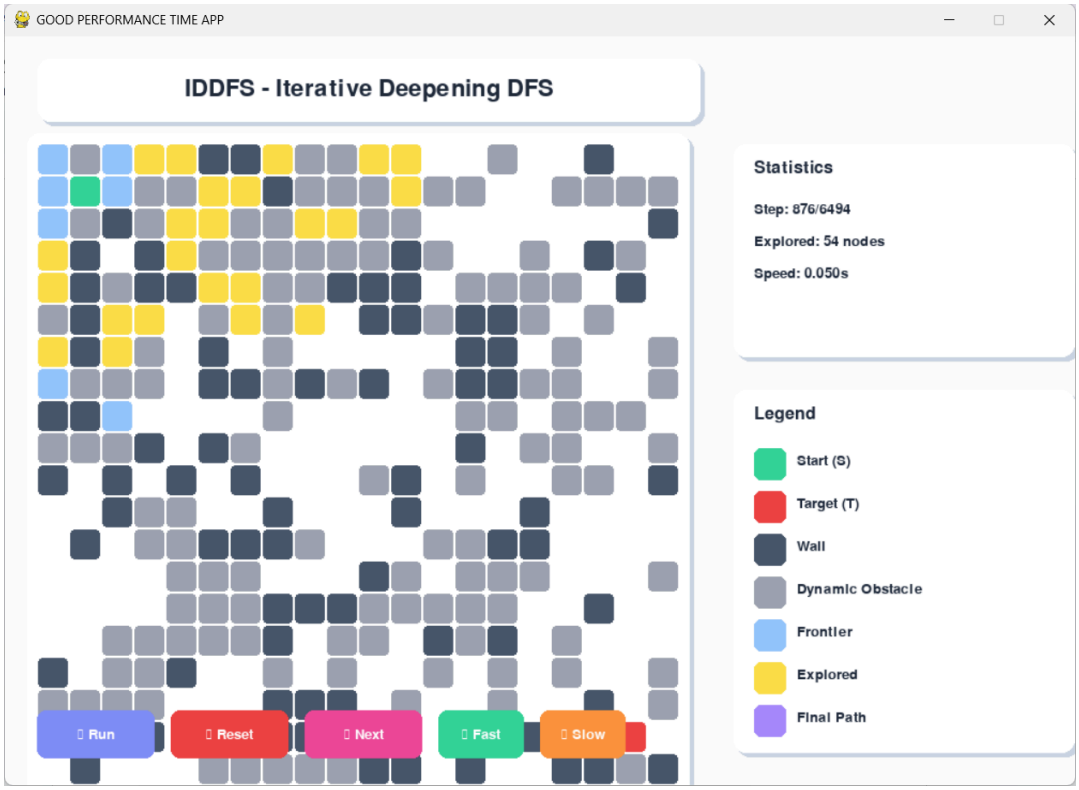
- **Logic**: DFS with a hard depth cutoff (Default: 15).
- **Pros**: Prevents infinite loops.
- **Cons**: Incomplete if the target lies beyond the limit.

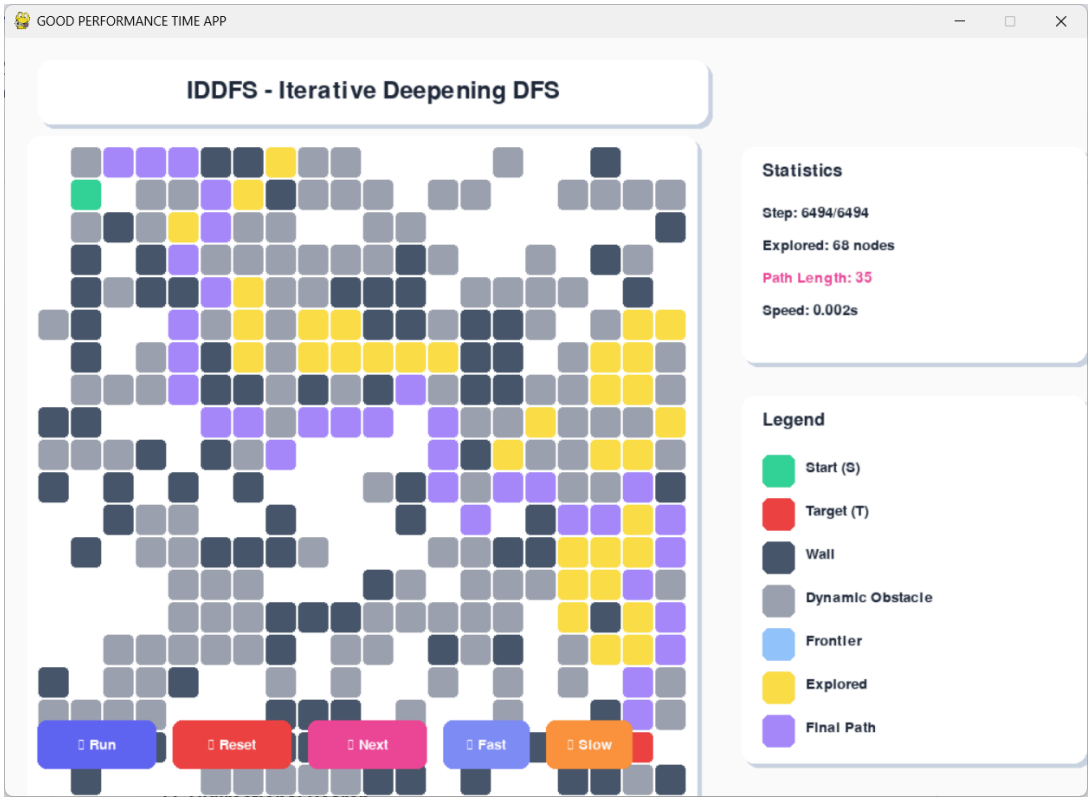




3.5 Iterative Deepening DFS (IDDFS)

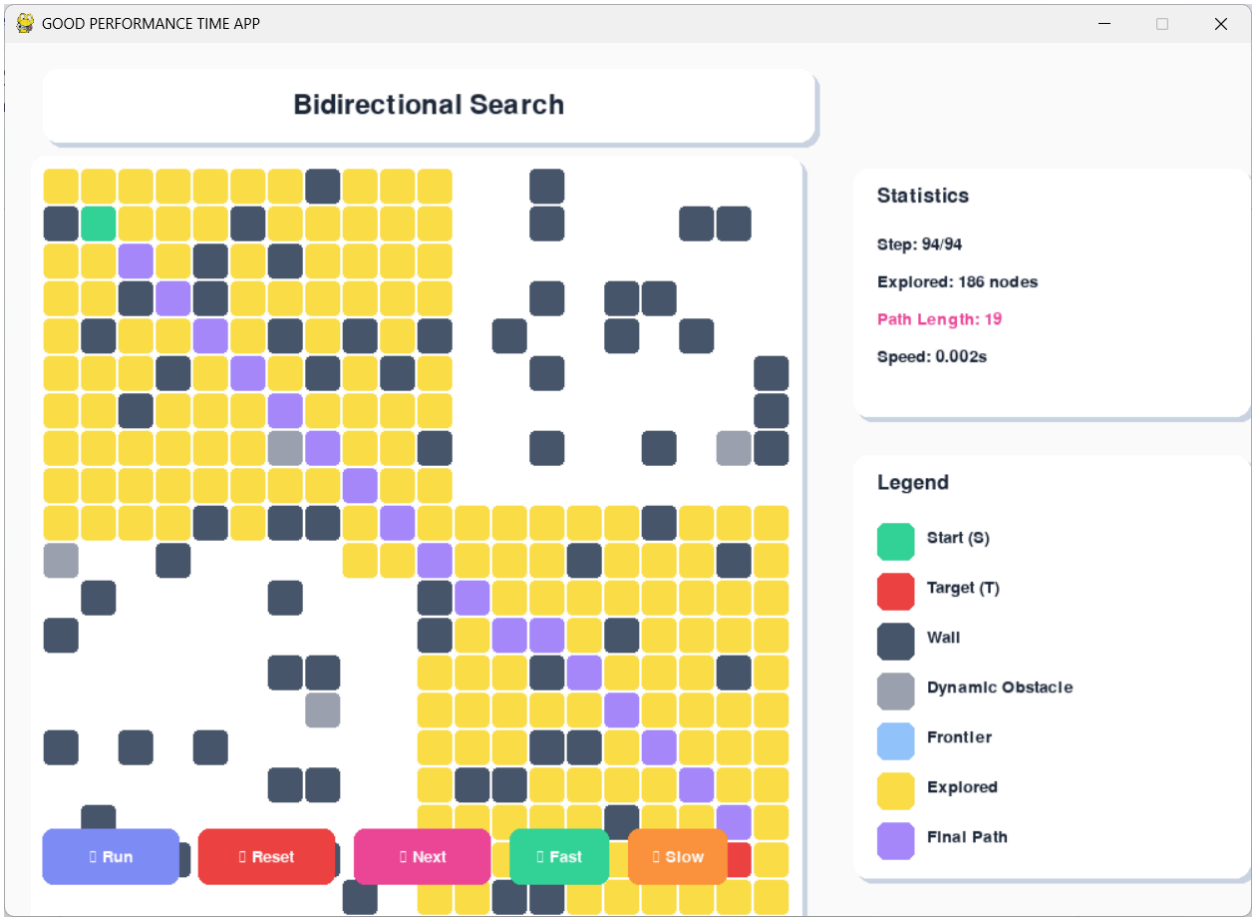
- **Logic**: Repeatedly runs DLS with increasing limits.
- **Pros**: Combines BFS optimality with DFS memory efficiency.
- **Cons**: Redundant exploration of shallow nodes.





3.6 Bidirectional Search

- **Logic**: Simultaneous BFS from both Start and Target.
- **Pros**: Significantly reduces search space (O(b^{d/2})).
- **Cons**: Requires a reversible search space.



---

4. Test Results

4.1 Comparative Data Table

Algorithm	Best Case (Nodes)	Worst Case (Nodes)	Optimal?	Memory Usage
BFS	45	287	✅ Yes	High
DFS	28	312	❌ No	Low

UCS	48	295	✅ Yes	High
DLS	42	156 (Failed)	❌ No	Low
IDDFS	89	524	✅ Yes	Low
Bidirectional	24	156	✅ Yes	High

---

5. Performance Analysis

Key Findings

1. **Efficiency Champion**: **Bidirectional Search** explored the fewest nodes and offered the lowest execution time.
2. **Resource Efficiency**: **DFS/DLS** maintained the smallest memory footprint but sacrificed path quality.
3. **Reliability**: **BFS and IDDFS** consistently found the mathematical shortest path.
4. **Adaptability**: All algorithms successfully re-routed when dynamic obstacles blocked the existing path by triggering a re-scan.

> **Note on Movement Costs**: While BFS treats all moves as 1, UCS correctly identifies that diagonal movement is more "expensive" than cardinal movement, leading to more realistic pathfinding.

---

6. Conclusion

The project successfully visualizes the trade-offs between time and space complexity in uninformed search. While **BFS** is the standard for accuracy, **Bidirectional Search** proves to be the most performant in a grid environment.

Future Enhancements

- Implementation of **Informed Search** (A*, Greedy Best-First).
- Path smoothing using string pulling or spline interpolation.
- Integration of a performance metrics dashboard for live benchmarking.