

□ AI Maze Pathfinding

Search Algorithms Project – Python

Introduction

This project demonstrates the application of Artificial Intelligence search algorithms to solve a two-dimensional maze problem. The objective of the intelligent agent is to find a valid path from a Start node (S) to a Goal node (G) while avoiding obstacles and walls.

The project focuses on comparing Uninformed (Blind) and Informed (Heuristic) search techniques to evaluate their efficiency and behavior in pathfinding problems.

Project Features

- **Maze Visualization:** Displays explored nodes and the final solution path.
 - **Multiple Search Algorithms:** Implementation of six AI search strategies.
 - **Performance Evaluation:** Measures execution time, number of visited nodes, and path length.
 - **Flexible Maze Design:** Supports different maze layouts.
-

Implemented Search Algorithms

Uninformed Search Algorithms

Breadth-First Search (BFS)

Description: Explores the maze level by level starting from the initial state.

Advantages:

- Always finds the shortest path in an unweighted maze.
- Complete and optimal.

Disadvantages:

- Requires high memory usage.
- Explores many unnecessary nodes.

Depth-First Search (DFS)

Description: Explores one path deeply before backtracking.

Advantages:

- Very low memory consumption.
- Simple to implement.

Disadvantages:

- Does not guarantee the shortest path.
- Can get trapped in long paths.

3 Iterative Deepening Search (IDS)

Description: Applies DFS repeatedly with increasing depth limits until the goal is found.

Advantages:

- Combines BFS optimality with DFS memory efficiency.
- Finds the shortest path.

Disadvantages:

- Re-explores nodes multiple times.

4 Uniform Cost Search (UCS)

Description: Always expands the node with the lowest cumulative cost.

Advantages:

- Guarantees optimal solutions.
- Works well with weighted mazes.

Disadvantages:

- Equivalent to BFS when all costs are equal.

◆ Informed Search Algorithms

5 Hill Climbing

Description: A greedy algorithm that moves toward the neighbor closest to the goal.

Advantages:

- Extremely fast.

- Minimal memory usage.

Disadvantages:

- Not complete.
- Can get stuck in local optima.

A* Search Algorithm

Description: Uses the evaluation function:

$f(n) = g(n) + h(n)$

Heuristic Used:

- **Manhattan Distance:** $|x1 - x2| + |y1 - y2|$

Advantages:

- Finds the optimal path efficiently.
- Expands fewer nodes compared to BFS.
- Considered the most effective pathfinding algorithm.

Performance Evaluation

The following results were obtained from testing the algorithms on a 20×20 maze with moderate obstacles.

Algorithm	Path Length	Nodes Visited	Execution Time	Status
BFS	42 (Optimal)	380	15 ms	Solved
DFS	95 (Non-optimal)	120	4 ms	Solved
UCS	42 (Optimal)	390	16 ms	Solved
A*	42 (Optimal)	130	6 ms	Solved
Hill Climbing	—	45	1 ms	Failed

Comparative Analysis

Time Complexity

- BFS: $O(V + E)$
- DFS: $O(V + E)$
- UCS: $O(V + E)$
- IDS: $O(V + E)$
- A*: Depends on heuristic quality
- Hill Climbing: $O(b)$

Optimality & Cross Path

Algorithm	Optimal Path	Cross Path
BFS	Yes	No
DFS	No	No
UCS	Yes	No
IDS	Yes	No
A*	Yes	No
Hill Climbing	No	Yes

Conclusion

Uninformed search algorithms explore large portions of the maze without guidance, leading to higher time and memory consumption. In contrast, informed algorithms such as A* significantly reduce the number of explored nodes by using heuristics, while still guaranteeing optimal solutions.

Overall, A* provides the best balance between efficiency, accuracy, and performance, making it the preferred algorithm for maze pathfinding problems.

Project Author

mahmoud hany emara