Report

DV2618: APPLIED ARTIFICIAL INTELLIGENCE

September 23, 2024

Student	Name	GURRAM VENKATA VISHNU VARDHAN REDDY
	E-Mail	vegu23@student.bth.se

1 Introduction

This report explores the implementation of three different algorithms Breadth-First search (BFS), Depth-First Search (DFS), and A* search—to solve a maze represented as a 2D grid. It covers implementation details, performance metrics, strengths and weaknesses of each algorithm, and provides insights into which algorithm is most suitable for this specific maze problem.

The goal of each algorithm is to find the shortest path from a starting point to an endpoint. A 2D array, where define the maze:

- 0 represents open paths.
- 1 represents walls.
- The start point is labeled as S.
- The endpoint is labeled as E.

2 Implementation Details

- **BFS:** BFS uses a queue data structure to keep track of nodes to visit. The algorithm starts from the start node and explores all its neighbors before moving to the next level of nodes.
- **DFS:** DFS uses a stack data structure to keep track of nodes to visit. The algorithm starts from the start node and explores each branch as far as possible before backtracking.
- **A*:** A* using a priority queue data structure to keep track of nodes to visit. The algorithm starts from the start node and first explores nodes with the lowest estimated total cost (heuristic + cost so far).

3 Performance Metrics

Algorithm	Path Length	Memory Usage
BFS	27	(1168, 4864)
DFS	31	(728, 2544)
A*	27	(1168, 6624)

Table 1: Performance Comparison of Algorithms

3.1 BFS:

Path: The algorithm finds the shortest path, which is 27 steps long.
Memory Used: BFS used 1168 bytes current and 2544 bytes peak.
Strengths: BFS is guaranteed to find the shortest path in an unweighted grid like this maze.
Weaknesses: The memory usage can be significant, especially for larger grids, as BFS explores many nodes at each level before proceeding deeper.

3.2 DFS:

Path: DFS found a long path of length 31, which is not optimal.
Memory Used: DFS was the most memory-efficient algorithm, using 728 bytes initially and peaking at 2544 bytes.
Strengths: DFS is memory-efficient in general but can consume more in deeply nested or complex mazes.
Weaknesses: DFS does not guarantee the shortest path. It may also return a significantly longer path in other configurations. It can also get stuck in deep branches, leading to increased computation time.

3.3 A*:

Path: A* found the shortest path, identical to BFS, with a length of 27 steps.
Memory Used: The memory usage was (1168, 6624) bytes, which is significantly lower than DFS and BFS had the same current memory usage, but A* had a higher peak due to due to the additional overhead from the heuristic calculations.
Strengths: A* is highly efficient in both path length and memory usage. The Manhattan distance heuristic effectively guides the search towards the goal, making it faster and less memory-intensive.
Weaknesses: A* requires a well-defined heuristic. In complex or unstructured problems, designing an effective heuristic can be challenging.

4 Comparative Analysis

Path Optimality:

- BFS and A* have the shortest path (27 steps).
- DFS found a longer path (31 steps), as expected due to its nature.

Memory Efficiency:

- DFS was the most memory-efficient.
- BFS and A* had the same current memory usage, but A* had a higher peak.

Algorithm Behavior:

- BFS and A* showed similar behavior, finding the same optimal path.
- DFS explored more unnecessary areas, resulting in a longer path.

5 Algorithm Suitability:

- BFS is suitable when finding the shortest path is essential, but for larger mazes, it can become memory-inefficient.
- DFS is more suitable when memory efficiency is important, and finding the shortest path is not a priority. It is less effective for grid-based problems like this maze where the shortest path is desired.
- A* strikes a balance between BFS and DFS. It guarantees the shortest path and performs efficiently, though with higher memory costs. For grid-based problems with a clear goal, A* is the most suitable choice.

6 Conclusion

The A* algorithm proves to be the best overall approach for solving this maze problem, as it guarantees the shortest path with reasonable memory usage, making it highly efficient in both small and large grids. BFS also guarantees the shortest path but at a higher memory cost. DFS is the most memory-efficient but does not guarantee an optimal solution, making it less suitable for this specific maze problem.