Comparison of Pathfinding Algorithms: BFS, DFS, and A* with Different Heuristics

KASMI El Mehdi

October 6, 2024

Abstract

In this article, I compare three popular pathfinding algorithms: Breadth-First Search (BFS), Depth-First Search (DFS), and A* search. Additionally, we explore three variations of the A* algorithm using different heuristics: Euclidean distance, Manhattan distance, and Diagonal distance. This comparison highlights their performance, efficiency, and suitability for solving maze problems.. For the full implementation and GUI, visit my GitHub repository: `https://github.com/baldwin-sudo/graph_algorithms.git`.

1 Introduction

Pathfinding algorithms are essential in fields such as AI, robotics, and game development. In this article, I focus on three algorithms: BFS, DFS, and A*. Each has distinct properties in terms of search space, time complexity, and optimality. I also implement and compare three heuristics for A*: Euclidean distance, Manhattan distance, and Diagonal distance.

2 Algorithms Overview

2.1 Breadth-First Search (BFS)

Breadth-First Search (BFS) explores all nodes at the current depth level before moving on to nodes at the next level. It guarantees the shortest path in unweighted graphs, but its time complexity can be a limitation for larger spaces.

2.2 Depth-First Search (DFS)

Depth-First Search (DFS) explores each path as deeply as possible before backtracking. While DFS can be more efficient in certain scenarios, it does not guarantee the shortest path and may get stuck in deep paths in certain graph structures.

2.3 A* Search

A* combines the best of BFS and DFS by using a heuristic function to prioritize nodes. The heuristic estimates the cost from a node to the goal, helping A* find the shortest path more efficiently.

2.3.1 Heuristic 1: Euclidean Distance

The Euclidean distance heuristic computes the straight-line distance between two points, making it suitable for continuous spaces.

2.3.2 Heuristic 2: Manhattan Distance

The Manhattan distance heuristic sums the absolute differences between the coordinates of two points. It works best in grid-based environments where movement is constrained to horizontal and vertical directions.

2.3.3 Heuristic 3: Diagonal Distance

The Diagonal distance heuristic is useful in spaces that allow diagonal movement. It accounts for both horizontal and vertical movements, making it a better fit for scenarios with diagonal paths.

3 Visualization

The figures below depict the paths generated by BFS, DFS, and A* using different heuristics.

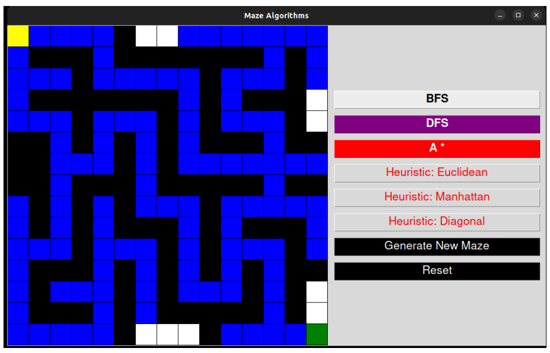

Figure 1: Nodes visited by BFS

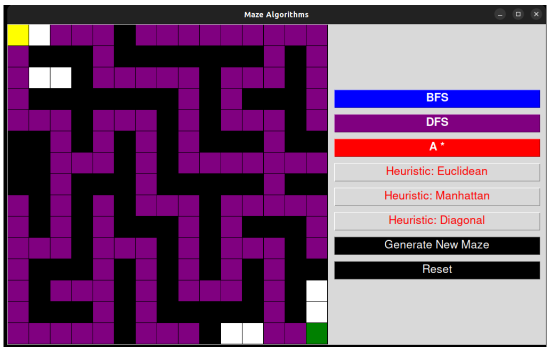

Figure 2: Nodes visited by DFS

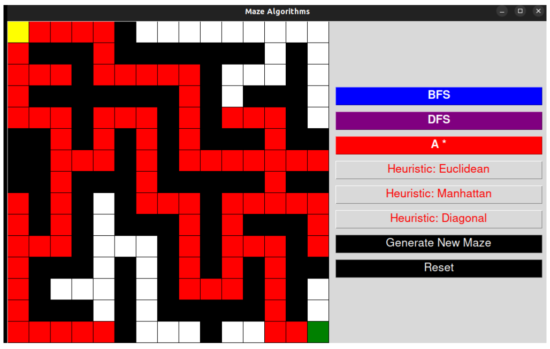

Figure 3: Nodes visited by A* (Euclidean Heuristic)

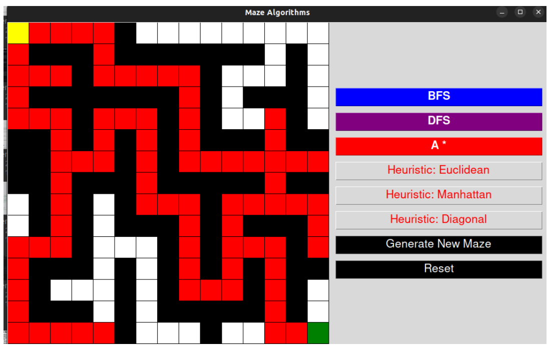

Figure 4: Nodes visited by A* (Manhattan Heuristic)

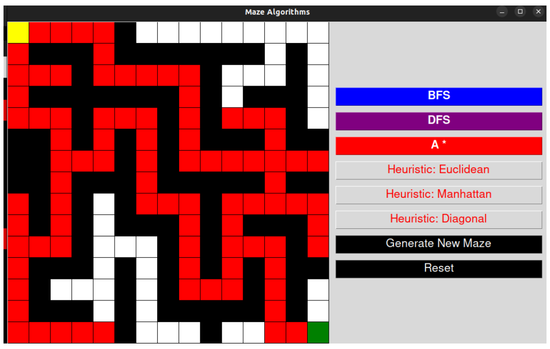

Figure 5: Nodes visited by A* (Diagonal Heuristic)

4 Results and Discussion

4.1 Comparison Criteria

I evaluate the algorithms based on the following criteria:

- **Time complexity:** The time taken by the algorithm to find a path.
- **Space complexity:** The memory usage during the search process.
- **Optimality:** Whether the algorithm guarantees the shortest path.
- **Heuristic Influence:** The effect of different heuristics on A* performance.

4.2 Time and Space Complexities of Selected Graph Algorithms

Algorithm	**Time Complexity**	**Space Complexity**	**Optimality**
Breadth-First Search (BFS)	$O(V+E)$	$O(V)$	Yes
Depth-First Search (DFS)	$O(V+E)$	$O(V)$ or $O(h)$	No
A* Search Algorithm	$O(E)$	$O(V)$	Yes (with admissible heuristic)

Table 1: Time and Space Complexities of Selected Graph Algorithms

Notes:

- V: Number of vertices in the graph.
- E: Number of edges in the graph.
- The complexities can vary based on the implementation details, such as the data structures used.
- For A*, optimality is guaranteed if the heuristic is admissible (never overestimates the true cost).

4.3 Heuristic Comparison

The heuristics have a significant influence on the A* algorithm's performance.

- Euclidean distance provides a balanced approach
- Manhattan distance is ideal for grid-like maps with vertical and horizontal movement.
- Diagonal distance is more efficient when diagonal movement is allowed.

5 Conclusion

The comparison between BFS, DFS, and A* reveals distinct strengths for each algorithm. BFS is guaranteed to find the shortest path but can be inefficient for large grids. DFS is faster but may not produce optimal paths. A* strikes a balance between efficiency and optimality, with different heuristics affecting its performance. In environments with diagonal movement, the Diagonal heuristic is most efficient, while Manhattan is ideal for grid-based, straight-line paths.