Graph Theory Project Report

Topic: Dijkstra's Algorithm vs. A\* Algorithm

**Section: BCS-5C**

**Group members:**

**21K-3333 21K-3232 21K-3272**

**Objective:**

The primary objective of this project is to compare the performance of Dijkstra's algorithm and A\* algorithm in solving a pathfinding problem. Both algorithms are widely used for finding the shortest path in graphs, but they differ in their approach and efficiency.

**Algorithms Overview:**

1. Dijkstra's Algorithm:

- Methodology: Dijkstra's algorithm explores all possible paths from the source node to all other nodes, updating the shortest distances as it progresses.

- Pros:

- Guarantees the shortest path.

- Suitable for graphs with uniform edge weights.

- Cons:

- Inefficient for graphs with varying edge weights.

- May explore unnecessary paths.

2. A\* Algorithm:

- Methodology: A\* algorithm combines the benefits of Dijkstra's algorithm and heuristic approaches. It uses a heuristic function to estimate the cost of reaching the goal from the current node and guides the search towards the most promising paths.

- Pros:

- Generally more efficient than Dijkstra's algorithm.

- Well-suited for graphs with varying edge weights and heuristics.

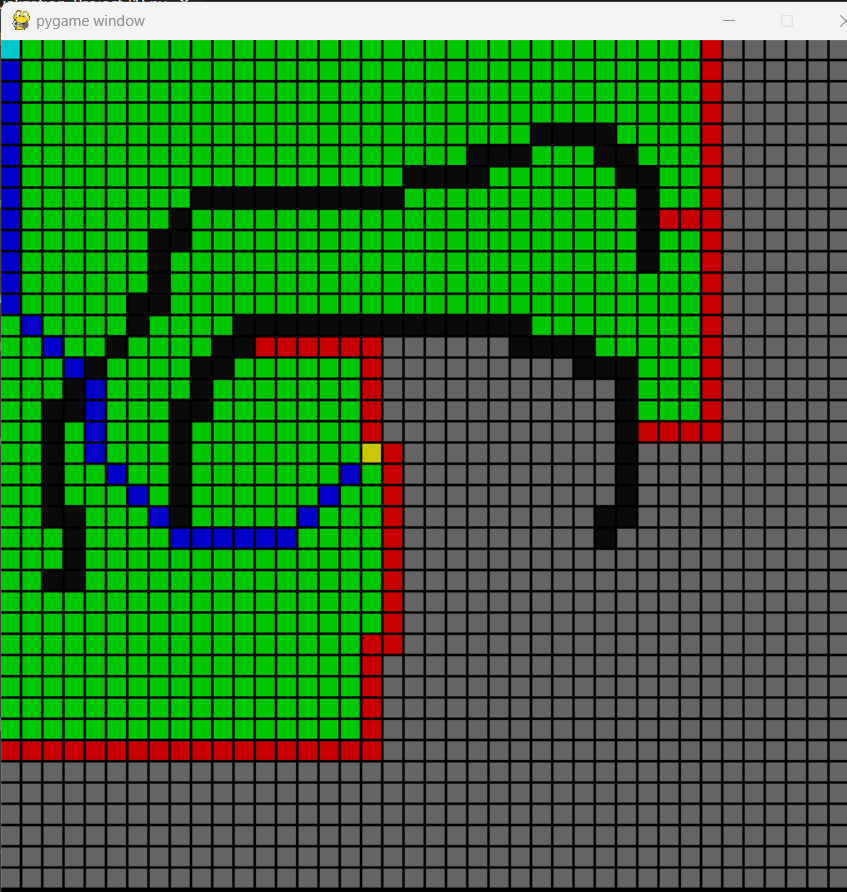
- Cons:

- May not guarantee the shortest path in certain scenarios.

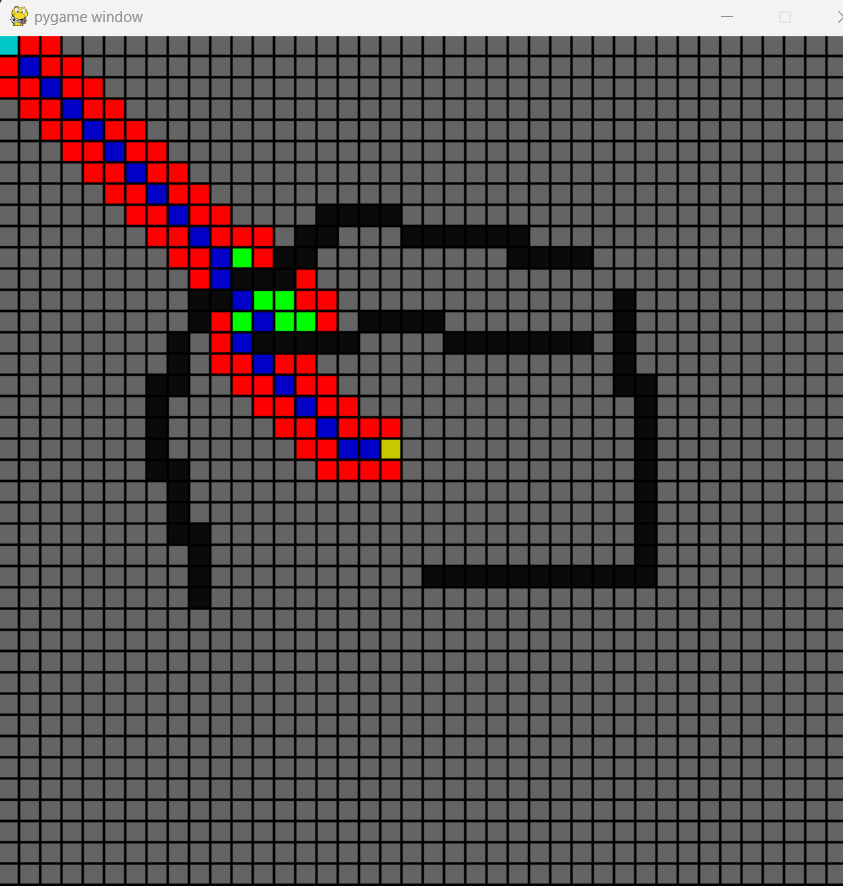
**Implementation:**

Both algorithms were implemented in Python, and the same set of input data and graph structures were used for a fair comparison. The code includes functionalities to visualize the graph, input nodes, and display the computed paths.

**Results:**

1. Dijkstra's Algorithm Output:

2. A\* Algorithm Output:



**Performance Analysis:**

1. Dijkstra's Algorithm:

- Successfully finds the shortest path.

- Path may not be optimal in graphs with varying edge weights.

2. A\* Algorithm:

- Generally more efficient than Dijkstra's.

- Takes advantage of heuristics for faster convergence.

**Conclusion:**

The choice between Dijkstra's algorithm and A\* algorithm depends on the characteristics of the graph. For scenarios where edge weights are uniform, Dijkstra's algorithm suffices. However, A\* algorithm proves advantageous in graphs with varying edge weights, providing a more efficient solution.

Note: The attached screenshots illustrate the output of the algorithms for better visualization.