*A Pathfinding Algorithm Implementation**

1. Introduction

The A* (A-star) algorithm is one of the most popular and efficient pathfinding algorithms used in artificial intelligence, robotics, and game development. It finds the shortest path between a start node and a goal node in a graph while considering both cost and heuristic estimations.

2. Objective

The goal of this project is to implement the A* algorithm in Python and apply it to a grid-based environment to find an optimal path from a given start position to a specified goal.

3. Methodology

- *Node Representation:* Each node in the grid is represented as a class with attributes for position, parent, and cost values (g, h, f).

- *Heuristic Function:* The Manhattan distance heuristic is used to estimate the cost from the current node to the goal.

- *Algorithm Implementation:*

  - Open and closed lists are used to manage nodes.

  - The algorithm selects the node with the lowest total cost (f = g + h).

  - Neighboring nodes are evaluated, and the shortest path is determined.

- *Grid Representation:* The environment is represented as a 2D list where 0 indicates open paths and 1 represents obstacles.

4. Implementation Details

The Python implementation consists of the following:

- A Node class to store position, parent, and cost values.

- A heuristic function to compute the Manhattan distance.

- The astar function to process the grid and determine the optimal path.

- A sample 5x5 grid with obstacles to test the algorithm.

5. Results

Upon running the implementation, the algorithm successfully finds the shortest path from the start to the goal while avoiding obstacles. The output displays the sequence of coordinates representing the optimal path.

6. Conclusion

The A* algorithm is an efficient and optimal pathfinding technique. Its balance between cost evaluation and heuristic estimation makes it suitable for various applications, including robotics, navigation systems, and game AI.

7. Future Enhancements

- Implementing A* on larger and dynamic grids.

- Exploring different heuristic functions for performance comparisons.

- Extending the implementation to 3D space for real-world navigation.