A* Algorithm: Navigating the Grid

Mohamed Zaghloul

Abstract and Real-World Applications:

The A* pathfinding algorithm is a powerful and widely-used method for finding the shortest path in a grid or map, navigating around obstacles to reach a goal. This project focuses on implementing a simplified version of the A* algorithm to solve the problem of navigating through a 2D grid, where obstacles may block the path and the objective is to find the most efficient route from a start point (S) to an end point (E).

In the real world, A* has broad applications:

- **Robotics**: For autonomous navigation in environments with obstacles.
- **Video Games**: To control character or NPC movement in dynamic environments.
- **Navigation Systems**: Used in GPS for calculating optimal routes, avoiding traffic, or detours.
- **Logistics**: Optimizing delivery routes to minimize time and costs.

By learning and implementing the A* algorithm, this project provides insight into essential problem-solving techniques and optimization methods used across multiple fields, including AI, game development, and real-world navigation systems.

Project Objective:

The goal of this project is to implement the A* algorithm to find the shortest path between a start point (S) and an end point (E) on a grid. The grid is represented as a 2D array, with obstacles (0) and walkable paths (1). The project will visually display the shortest path found by the algorithm using asterisks (*).

Approach and Methodology:

Map Representation:

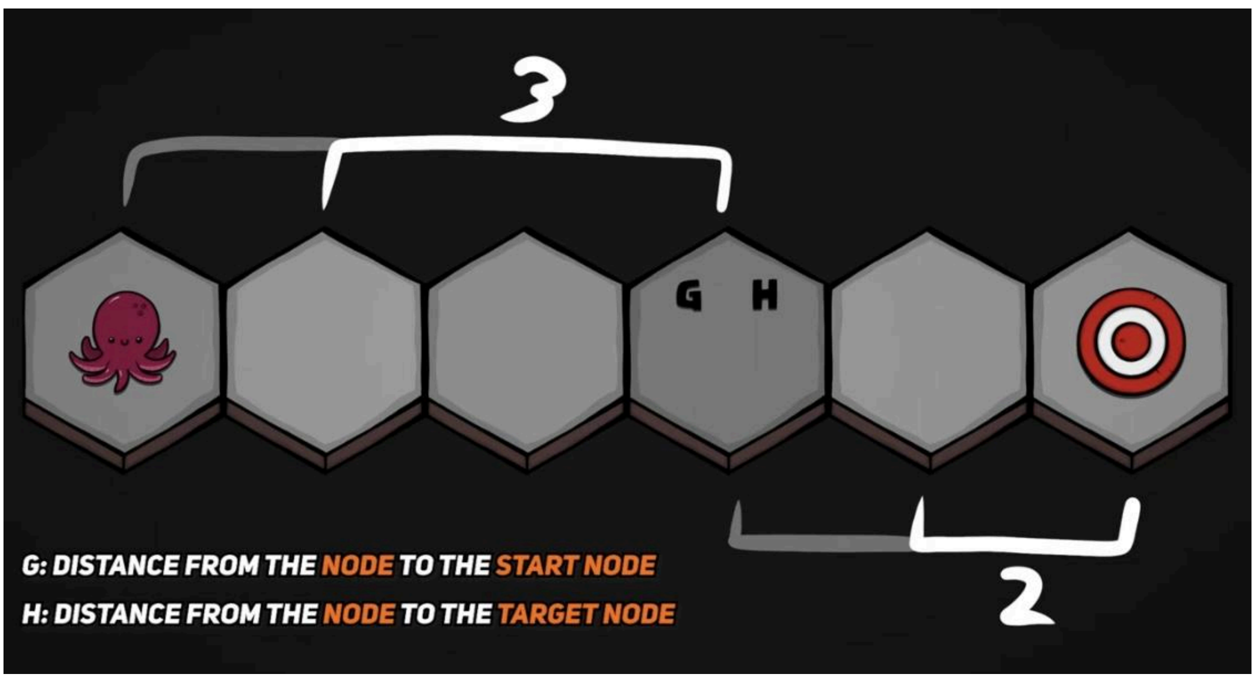
The grid will be represented as a 2D array of integers or characters, where:

- '0' represents obstacles (blocked paths),
- '1' represents walkable paths (free spaces),
- 'S' is the start point,
- 'E' is the end goal.

Note: There are 3 map levels (Easy, Medium, Hard) used for testing the algorithm

A Algorithm Overview:

- **Heuristic Function (h)**: The heuristic used will be the **Manhattan distance**, which calculates the sum of the horizontal and vertical distances between the current node and the goal.
- **Cost Function (g)**: The cost will simply count the number of steps from the start point.



- **Path Reconstruction**: After reaching the goal, the algorithm will backtrack from the goal to the start by following parent nodes, marking the path with '*'.

```
0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0
E * * * * * 0 1 1 1 0 * * * * * S
0 0 0 1 0 * 0 0 0 0 0 * 0 1 0 0 0
0 0 0 1 0 * * * * * * * 0 1 0 0 0
1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Goal Reached
```

Skills and Learning Outcomes:

This project will help develop essential programming skills such as:

- **Problem-solving**: Developing an understanding of how to break down a complex algorithm like A* into manageable steps.
- **Algorithm design**: Learning the fundamentals of pathfinding algorithms and heuristics.
- **Code organization**: Structuring the code for readability and maintaining modularity.
- **Visualization**: Generating outputs that help understand how the algorithm works.

Algorithm Implementation:

- **Node Structure**: Holds the current position, costs, and parent node for path reconstruction.
- **Heuristic Function**: Manhattan distance between current node and goal.
- **Priority Queue**: Used to always expand the node with the lowest f cost. Path
- **Reconstruction**: Traces back from the goal to start using the parent nodes and marks the path in the map

Output:

```
1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1
1 0 0 1 0 0 0 1 0 1 0 0 0 1 0 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 1
1 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1
0 0 0 1 0 0 0 1 0 1 0 0 0 1 0 0 0
0 0 0 1 0 1 1 1 1 1 1 1 0 1 0 0 0
0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0
E * * * * * 0 1 1 1 0 * * * * * S
0 0 0 1 0 * 0 0 0 0 0 * 0 1 0 0 0
0 0 0 1 0 * * * * * * * 0 1 0 0 0
0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0
1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1
1 0 0 1 0 0 0 1 0 1 0 0 0 1 0 0 1
1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1
0 1 0 1 0 1 0 0 0 0 0 1 0 1 0 1 0
1 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1
1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Goal Reached
```

Please find the three maps (easy, medium, and hard) along with their solutions and the main C++ file attached. If there are any edits needed in the code or further adjustments required, feel free to reach out.