Solving Maze Pathfinding Problem With A* Algorithm

A* Concept:

It is an informed search algorithm(heuristic search). It uses hints to guide the search toward the goal, allowing it to find solutions more efficiently than uninformed search algorithms (like BFS or DFS).
The heuristic function **h(n)**, estimates the cost from the current state to the goal state (it is not the actual cost **g(n)**).
Manhattan Distance heuristic function $|x_1 - x_2| + |y_1 - y_2|$ ***is used in this project because the maze allows movement only in four directions (up, down, left, and right).***
A* combines both the actual cost to reach a node and the estimated cost to the goal to determine the best path , So the total cost: **F(n)=g(n)+h(n)**
This combination ensures that A* efficiently balances exploring new nodes and moving toward the goal.

How A* works on this maze:

It starts with defining the start cell (which is **(0,0)**) and goal cell **(19,19)** in our 20×20 maze, and it initializes the open list (priority queue) with the start cell. For each cell, it calculates **h(n) =** $|x_1 - x_2| + |y_1 - y_2|$ this tells the algorithm how far the goal is from the current cell, but it doesn't consider the walls ,so it also calculate actual cost g(n). These are combined to calculate the total cost: **f(n) = h(n) + g(n).** This final equation determines the priority of each node in the open list. The node with the **lowest f(n)** value is selected to be explored. For each selected node, its neighbors in the four directions are examined. If a shorter path to a neighbor is found, its cost is updated, the parent of that neighbor is stored and it is added to the open list. All explored nodes are recorded during the search. When the algorithm reach the goal, it reconstructs the final path using the stored parent information, calculates its execution time, and visualizes the maze, highlighting both the explored nodes and the optimal path.

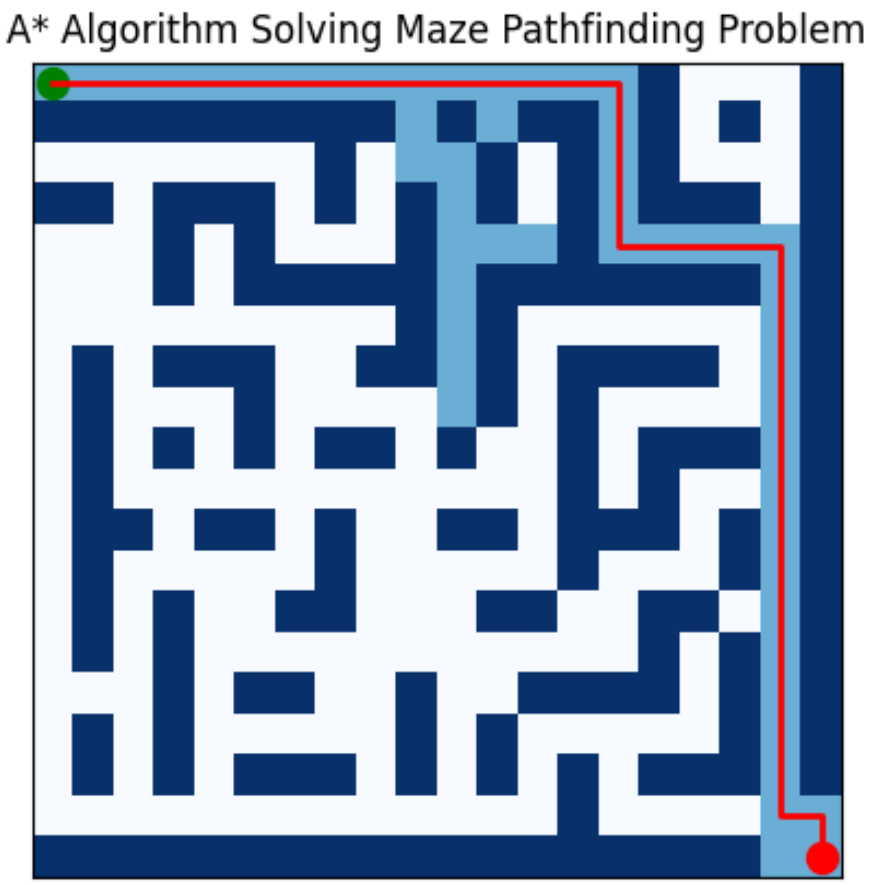
Analysis of heuristic effectiveness:

The Manhattan distance heuristic guides A* toward the goal efficiently in our 20×20 maze, allowing movement in four directions. It is admissible, so it never overestimates the true cost and guarantees the shortest path. Even though the heuristic doesn't directly consider walls, it helps the algorithm focus on the goal, exploring fewer unnecessary nodes than BFS or DFS. This makes the search quicker and more efficient.

Time and space complexity for A*:

Metric	Best Case	Worst Case
Time Complexity	O(L)	O(B^L)
Space Complexity	O(L)	O(B^L)

L = path length, **B** = branching factor, **N** = total number of cells in the maze.



The thin red line represents the actual path and the wide blue area shows the explored nodes.

Path length: 39
Explored nodes: 52
Time: 0.0003 seconds

Conditions for correct execution of the A* code:

- The maze must be represented as a 2D grid, where **0** represents free cell and **1** represents walls
- The start and goal must be **inside** the maze boundaries and **located on free cells**
- Each move has a **uniform cost** of 1 (unweighted grid)
- Must be **admissible** (never overestimates the actual cost)