# A\* Algorithm Description

This project is about finding the shortest path between two points using the A\* algorithm. The program takes a small graph as input and tries to move from a start node to a goal node in the most efficient way possible.

## How It Works

The algorithm starts from a starting point (for example, node A) and looks at all the possible paths it can take. For each path, it checks two things:  
  
1. How much distance has already been covered from the start (called g cost).  
2. How much distance is still estimated to reach the goal (called h cost).  
  
Then it adds both values to get the total cost (f = g + h). The node with the lowest total cost is chosen next, meaning it seems closest to the goal.  
  
The program keeps repeating this process — moving to the node that looks best — until it finally reaches the goal node.

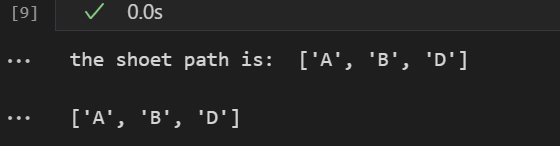
## Main Steps

- The graph is stored in a dictionary, where each place (node) has a list of nearby places and how far they are.  
- The algorithm starts from the start node and keeps a record of which nodes have been checked and which still need to be visited.  
- For every step, it picks the next node that has the smallest total cost.  
- When the goal is found, it goes backward from the goal to the start to show the full shortest path.

## Example

If the graph has nodes A, B, C, and D, and we want to go from A to D, the algorithm may find these possible paths:  
  
- A → D  
- A → B → D  
- A → C → D  
  
It compares all of them and prints the shortest one.  
In this case, it shows: “The shortest path is: ['A', 'B', 'D']”

**OUTPUT**

****

## Conclusion

The A\* algorithm is a smart way to find the shortest path because it combines both real distance and estimated distance to guide the search. It’s widely used in maps, games, and navigation systems to find quick routes efficiently.