CSE-AI TY A div

Student Name: Prachee Prasad

Roll No.: 381060

Assignment 4

To Implement A* Algorithm for an Application

Problem Statement:

To implement the **A*** (A-Star) algorithm for finding the optimal path between two locations using heuristic-based search.
The algorithm should maintain **Open** and **Closed Lists** at each step and display the search process clearly.

Objective:

To understand how the **A*** search algorithm combines **heuristic** and **path cost** information to efficiently find the shortest or least-cost path in a search space.

Requirements:

- Input: Predefined nodes (e.g., Pune, Expressway, Mumbai) with their distances and heuristic estimates.
- Output: Step-by-step search process with Open and Closed lists, leading to the optimal path.
- Display: Each iteration showing nodes expanded, f(n) = g(n) + h(n) calculations, and final optimal path.

Operating System:

Windows / Linux / macOS

Libraries and Packages Used:

- **C++ iostream**, **map**, **vector**, **string**, and **algorithm** for managing data structures.
- No external packages are required.

Theory:

Definition:

The **A*** algorithm is an informed search algorithm that finds the optimal path to a goal node by considering both:

- **g(n):** The cost from the start node to the current node.
- **h(n):** The heuristic (estimated) cost from the current node to the goal.
- The total cost function is **f(n) = g(n) + h(n)**.

Structure:

- **Open List:** Stores nodes to be evaluated, sorted by their f(n) value.
- **Closed List:** Stores nodes already evaluated.
- **Heuristic Function:** Estimates cost to reach the goal (problem-specific).

Methodology:

1. Initialize Open List with the starting node.
2. While Open List is not empty:
    - Select the node with the lowest f(n).
    - Move it to the Closed List.
    - For each neighbor:
        - Calculate g(n), h(n), and f(n).
        - If not visited, add to Open List.

3. Repeat until the goal node is reached.
4. Reconstruct the optimal path by backtracking from the goal.

Advantages:

- Finds the **optimal path** if the heuristic is admissible (never overestimates cost).
- More **efficient** than uninformed search methods like BFS or DFS.
- Flexible for real-world pathfinding and routing problems.

Limitations:

- Performance depends heavily on the **quality of heuristic** used.
- Can consume significant **memory and computation** for large search spaces.
- May fail if heuristic is inconsistent or poorly designed.

Working / Algorithm:

Algorithm Steps:

1. Start with the **initial node (e.g., Pune)** in the Open List.
2. For each iteration:
    - Choose the node with the smallest f(n).
    - Expand it and move it to the Closed List.
    - Generate successors (neighboring paths).
    - Compute f(n) = g(n) + h(n) for each successor.
    - Add unvisited successors to the Open List.
3. Continue until the **goal (e.g., Mumbai)** is reached.

4. Display the sequence of Open and Closed Lists for each step.

5. Output the **Optimal Path** and **total cost**.

Conclusion:

The **A*** Algorithm efficiently combines both **path cost and heuristic knowledge** to find the shortest path in a search space.
It is one of the most widely used search algorithms in Artificial Intelligence, applied in navigation systems, robotics, and game development for optimal route planning.