Lab 7: Implementing A Search Algorithm*

Learning Objective

To implement the A* (A-Star) Search Algorithm and understand its working mechanism using Python.

A* Search

- A* Search is an **informed search** technique (also known as **best-first search**).
- It combines features of **Uniform Cost Search** and **Greedy Best-First Search**.
- It evaluates nodes using the function:

$$f(n) = g(n) + h(n)$$

  - **g(n)**: Actual cost from the start node to the current node.
  - **h(n)**: Heuristic estimate of the cost from the current node to the goal.
- A* finds the lowest cost path to the goal if **h(n)** is **admissible** (never overestimates) and **consistent**.

Why A* is Important?

- Finds the **optimal path** using both costs so far and estimated future cost.
- Commonly used in **navigation systems**, **game AI**, **robot path planning**, and more.

Key Properties of A*

- Complete
- Optimal (with admissible heuristic)
- Uses both path cost and heuristic to guide search
- Uses priority queue (open set) to store frontier nodes

Description of A* Components

1. openSet

- A collection of discovered nodes that are yet to be evaluated.
- Initially contains only the start node.
- Nodes are picked based on the lowest fScore.

2. cameFrom

- A map tracking the most efficient previous node for each explored node.
- Helps reconstruct the final path once the goal is reached.

3. gScore

- A map storing the cost of the shortest path from the start node to every other node.
- Initialized with ∞ for all nodes except the start node (set to 0).

4. fScore

- Estimated total cost from start to goal through a node: f(n) = g(n) + h(n).
- Combines known cost and estimated cost to guide the search.

5. h(n)

- The heuristic function that estimates the cost from a node to the goal.
- Must be admissible and consistent for A* to guarantee the shortest path.

6. reconstruct_path()

- Builds the final path from goal to start using cameFrom.

A* Pseudocode

```
function A_Star(start, goal, h):
   openSet = {start}                  // Initialize the open set with the start node
   cameFrom = empty map          // Tracks the best previous step to each node
    gScore = map with default ∞       // Cost from start to each node
   gScore[start] = 0
   fScore = map with default ∞       // Estimated total cost from start to goal through each node
fScore[start] = g(start) + h(start) = 0 + h(start)
fScore[start] = h(start)
   while openSet is not empty:
      current = node in openSet with lowest fScore[]     // Select node with the lowest estimated cost
```

```
if current == goal:
    return reconstruct_path(cameFrom, current)      // Goal reached, reconstruct the path
remove current
```

Lab Task

Write a Python program to:

- Implement a graph using dictionaries.
- Define a heuristic function for each node.
- Use the A* algorithm to find the shortest path between two nodes.
- Print the final path and cost.

Note: The efficiency of A* highly depends on the heuristic function used.