

**Name:**  Muhammad Awais Asghar

**Roll Number:** SU92-BSAIM-F24-065

**SUBJECT:** Artifical Intelligence (LAB)

**Task No -7**

**Question 1:**

**Why this code was made:**

The A (A-star) algorithm\*, which finds the quickest and most effective path between two points (or nodes) in a graph, is implemented by this code.  
  
This code's primary goal is:  
to examine every path that could lead from a start node to a destination node.  
to determine the least expensive route (minimum total cost or distance).  
In order to do this intelligently, A\* employs a heuristic function that determines which nodes are most likely to lead to the goal more quickly rather than blindly pursuing every path.  
  
A\* is frequently utilized in:  
NPC pathfinding in game development  
systems for navigation (such as GPS)  
Robotics (moving on its own)  
Issues with AI planning and route optimization.

**How this code works:**

**1.Display of Graphs**  
An adjacency list, which lists each node's neighbors and the cost of reaching them, is used to represent the graph.  
For instance, 'A': [('B', 1), ('C', 3)] indicates that A → B costs 1 and A → C costs 3.  
  
**2. Setup**  
open\_list → a queue (using deque) that stores nodes that are yet to be fully explored.

closed\_list → a collection of previously investigated nodes.  
The actual cost from the start node to each node is stored in a dictionary called g.  
parents → a map from which the ultimate shortest path is subsequently reconstructed.

**3. Choosing the Next Node**  
The algorithm selects the node with the lowest value of f = g + h from the open\_list at each step:  
g → the true cost from the beginning to the present node  
h → heuristic (estimated cost between the goal and the current node)  
Because of this combination, A\* is "smart" and investigates nodes that appear to be closer to the objective first.  
  
**4. Objective Verification**  
If the goal node and the current node are equal:  
By returning from the goal to the beginning using the parents dictionary, the algorithm recreates the route.  
This provides the shortest route along with its overall cost.

**5. Investigating Adjacent People**  
If the objective has not yet been accomplished:  
For every neighbor of the node in question:  
Determine the tentative cost (tentative\_g) by adding the edge cost to the cost thus far.  
Update g and parents if this path is shorter than any known path.  
If the neighbor hasn't been investigated yet, add it to the open\_list.  
  
**6. Repetition of the Procedure**  
The algorithm is repeated:  
selecting the f = g + h node that is the next lowest.  
neighbor expansion.  
Until the open\_list is empty or the goal is located.  
There is no path between start and goal if open\_list is empty and the goal has not been reached.  
  
**7. Outcome**  
If it works:  
gives back the shortest route (as a node list).  
reimburses the entire cost of the trip.  
If not successful:  
"Path does not exist!" is printed.

