1.D.)Implement A* algorithm

AIM:

To implement the A* algorithm for finding the shortest path from a start node to a goal node, using both the actual cost (g) and a heuristic estimate (h).

CODE:

```
import heapq

def a_star(graph, start, goal, heuristic):
    open_set = [(0, start)]  # (f_score, node)
    came_from = {}
    g_score = {node: float('inf') for node in graph}
    g_score[start] = 0
    f_score = {node: float('inf') for node in graph}
    f_score[start] = heuristic(start, goal)

    while open_set:
        current_f, current_node = heapq.heappop(open_set)

        if current_node == goal:
            path = []
            while current_node in came_from:
                path.append(current_node)
                current_node = came_from[current_node]
            path.append(start)
            return path[::-1]  # Reverse the path to get it from start to
goal

        for neighbor, cost in graph[current_node].items():
            tentative_g_score = g_score[current_node] + cost
            if tentative_g_score < g_score[neighbor]:
                came_from[neighbor] = current_node
                g_score[neighbor] = tentative_g_score
```

```
                f_score[neighbor] = tentative_g_score + heuristic(neighbor,
goal)
                if (f_score[neighbor], neighbor) not in open_set:
                    heapq.heappush(open_set, (f_score[neighbor], neighbor))
    return None  # No path found


# Example usage:
def manhattan_distance(node1, node2):
    x1, y1 = node1
    x2, y2 = node2
    return abs(x1 - x2) + abs(y1 - y2)

graph = {
    (0, 0): {(0, 1): 1, (1, 0): 1},
    (0, 1): {(0, 0): 1, (0, 2): 1},
    (1, 0): {(0, 0): 1, (1, 1): 1},
    (1, 1): {(1, 0): 1, (0, 1): 1, (1,2):1},
    (0, 2): {(0, 1): 1, (1, 2): 1},
    (1, 2): {(1, 1): 1, (0,2):1}
}
start_node = (0, 0)
goal_node = (1, 2)


path = a_star(graph, start_node, goal_node, manhattan_distance)

if path:
    print(f"Path from {start_node} to {goal_node}: {path}")
else:
    print(f"No path found from {start_node} to {goal_node}")
```

OUTPUT:

Path from (0, 0) to (1, 2): [(0, 0), (0, 1), (0, 2), (1, 2)]

RESULT:

The code is executed as expected and the output have been verified successfully.