Ex: 4

Aim:

To implement A* Search using python

ALGORITHM:

Step 1: Start

Step 2: Input graph as adjacency list where node is connected to the neighbours with given weight

Step 3: Initialize two set: open set for nodes with evaluated and closed set for nodes.

Step 4: Choose the open set, choose the node with lowest f-score (best estimated cost to goal

Step 5: If the current node is goal node terminate the search and reconstruct the path

Step 6: The goal is reached. Trace back the node to the start node to find the optimal path

Step 7: Stop

PROGRAM:

```
import heap
def a_star(graph, start, goal, heuristic):
    open_list = []
    heapq.heappush(open_list, (0, start))
    g_score = {node: 0}
    come_from = {start: None}
    f_costs = {start: heuristic(start)}
    while not open_list.empty():
    current = open_list.get()[1]
    if current == goal
    return reconstruct_path(come_from, current)
    for neighbour, cost in graph[current]
        tentative_g_cost = g_cost(current)
```

Date:

```
f_cost = tentative_g_cost + heuristic(neighbor)
f_cost[neighbor] = f_cost
open_list.put((f_cost, neighbor))
came_from[neighbor] = current
return None
def reconstruct_path(came_from, current):
    path = []
    while current:
        path.append(current)
        current = came_from[current]
    path.reverse()
    return path
start_node = 'A'
goal_node = 'G'
path = a_star(graph, start_node, goal_node)
print("Shortest path:", path)
```

Output

Shortest path ['A', 'E', 'D', 'G']

Result:

Thus A* search is implemented and the output is verified.