A* Search

Aim:

To implement A* search algorithm for finding shortest path b/w two nodes

Algorithm:

1. Start
2. Initialize list with start node and its f_cost = 0
3. Create g_cost to track the lowest known cost for start node to each node
4. Store parent of each node
5. Loop until list is empty
6. Print solution
7. Stop.

```
import heapq

def a_star_search(graph, start, goal, heuristic):
    open_list = []
    heapq.heappush(open_list, (0, start))
    g_cost = {start: 0}
    came_from = {start: None}
    while open_list:
        current_f_cost, current_node = heapq.heappop(open_list)
        if current_node == goal:
            return reconstruct_path(came_from, current_node)

        for neighbor, cost in graph[current_node]:
            tentative_g_cost = g_cost[current_node] + cost
            if neighbor not in g_cost or
                tentative_g_cost < g_cost[neighbor]:
                g_cost[neighbor] = tentative_g_cost
                f_cost = tentative_g_cost
                heapq.heappush(open_list, (f_cost, neighbor))
                came_from[neighbor] = current_node
    return None

def reconstruct_path(came_from, current):
    total_path = [current]
    while current in came_from and came_from
        [current] is not None:
        current = came_from[current]
        total_path.append(current)
    return total_path[::-1]
```

```
if __name__ == "__main__":
    graph = {
        'A': [('B', 1), ('C', 4)],
        'B': [('A', 1), ('D', 2), ('E', 5)],
        'C': [('A', 4), ('F', 3)],
        'D': [('B', 2)],
        'E': [('B', 5), ('F', 1)],
        'F': [('C', 3), ('E', 1)]
    }

    heuristic = {
        'A': 7,
        'B': 6,
        'C': 2,
        'D': 6,
        'E': 1,
        'F': 0
    }

    start_node = 'A'
    goal_node = 'F'
    path = a_star_search(graph, start_node,
                         goal_node, heuristic)

    if path:
        print(f"Path found: {' -> '.join(path)}")
```

Output:

Path found: A —> C —> F

Result:

The program has been executed successfully and the output has been verified.