A* Search Algorithm

Aim:

To implement the **A*** (**A-star**) search algorithm in Python to find the **shortest path** from a start node to a goal node in a weighted graph, using both path cost and a heuristic function to guide the search.

Code:

```
from queue import PriorityQueue


def a_star(start, goal, graph, h):
    open_list = PriorityQueue()
    open_list.put((0, start))
    came_from = {}
    g_score = {node: float('inf') for node in graph}
    g_score[start] = 0


    while not open_list.empty():
    _, current = open_list.get()
    if current == goal:
    path = []
    while current in came_from:
        path.append(current)
        current = came_from[current]
    path.append(start)
    path.reverse()
    return path
    for neighbor, cost in graph[current]:
```

```
            tentative_g_score = g_score[current] + cost
            if tentative_g_score < g_score[neighbor]:
                came_from[neighbor] = current
                g_score[neighbor] = tentative_g_score
                f_score = tentative_g_score + h[neighbor]
                open_list.put((f_score, neighbor))
        return None

    graph = {
        'A': [('B', 1), ('C', 3)],
        'B': [('D', 3), ('E', 1)],
        'C': [('F', 5)],
        'D': [],
        'E': [('F', 2)],
        'F': []
    }

    heuristic = {'A': 6, 'B': 4, 'C': 4, 'D': 2, 'E': 2, 'F': 0}
    path = a_star('A', 'F', graph, heuristic)
    print("Path found by A*:", path)
```

Result:

```
    Path found by A*: ['A', 'B', 'E', 'F']
```