Name: Usman Javed

Roll No: 023

A* Algorithm - Improved Code

```
import heapq

class Graph:
    def __init__(self, adjacency_list, heuristic):
        self.adjacency_list = adjacency_list
        self.heuristic = heuristic

    def get_neighbors(self, node):
        return self.adjacency_list.get(node, [])

    def a_star(self, start, goal):
        open_list = []
        heapq.heappush(open_list, (0, start))
        g_cost = {start: 0}
        came_from = {start: None}
        while open_list:
            current_f, current_node = heapq.heappop(open_list)
            if current_node == goal:
                return self.reconstruct_path(came_from, current_node)
            for neighbor, cost in self.get_neighbors(current_node):
                tentative_g = g_cost[current_node] + cost
                if neighbor not in g_cost or tentative_g < g_cost[neighbor]:
                    g_cost[neighbor] = tentative_g
                    f_cost = tentative_g + self.heuristic[neighbor]
                    heapq.heappush(open_list, (f_cost, neighbor))
                    came_from[neighbor] = current_node
        return None

    def reconstruct_path(self, came_from, current):
        path = []
        while current is not None:
            path.append(current)
            current = came_from[current]
        path.reverse()
        return path

adjacency_list = {
    'A': [('B', 1), ('C', 3), ('D', 7)],
    'B': [('D', 5)],
    'C': [('D', 12)],
    'D': []
}

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

graph = Graph(adjacency_list, heuristic)
path = graph.a_star('A', 'D')
print("Shortest Path:", path)
```