G. Case Study 7: Shortest Path Algorithms - Delivery Services

6. Implement Dijkstra's algorithm in Python for the given scenario.

```
import heapq

def dijkstra(graph, start):
    # Initialize distances and priority queue
    distances = {node: float('infinity') for node in graph}
    distances[start] = 0
    priority_queue = [(0, start)]

    while priority_queue:
        current_distance, current_node = heapq.heappop(priority_queue)

        # Nodes can be added to the priority queue multiple times, we only
process the shortest distance
        if current_distance > distances[current_node]:
            continue

        for neighbor, weight in graph[current_node].items():
            distance = current_distance + weight

            # Only consider this new path if it's better
            if distance < distances[neighbor]:
                distances[neighbor] = distance
                heapq.heappush(priority_queue, (distance, neighbor))

    return distances

# Example graph
graph = {
    'A': {'B': 5, 'D': 3},
    'B': {'A': 5, 'C': 2, 'E': 1},
    'C': {'B': 2, 'F': 4},
    'D': {'A': 3, 'E': 2},
    'E': {'B': 1, 'D': 2, 'F': 3},
    'F': {'C': 4, 'E': 3, 'G': 1},
    'G': {'F': 1}
}
```

```
# Compute shortest paths from 'A'
shortest_paths = dijkstra(graph, 'A')
print(shortest_paths)
```