91. Single Source Shortest Paths: Dijkstra's Algorithm

AIM: To find the shortest path by using the Djikstra’s Algorithm

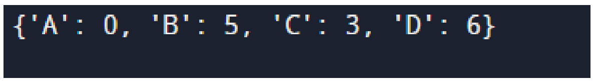
PROGRAM:

```
import heapq


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


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


        if current_distance > distances[current_node]:
            continue


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


            if distance < distances[neighbor]:
                distances[neighbor] = distance
                heapq.heappush(queue, (distance, neighbor))


    return distances
graph = {
    'A': {'B': 5, 'C': 3},
    'B': {'A': 5, 'C': 2, 'D': 1},
    'C': {'A': 3, 'B': 2, 'D': 4},
    'D': {'B': 1, 'C': 4}
}
```

start_node = 'A'

shortest_distances = dijkstra(graph, start_node)

print(shortest_distances)

OUTPUT: 

```
{'A': 0, 'B': 5, 'C': 3, 'D': 6}
```

TIME COMPLEXITY: O((V+E)log V)