199) Given a graph represented by an edge list, implement Dijkstra's Algorithm to find the shortest path from a given source vertex to a target vertex. The graph is represented as a list of edges where each edge is a tuple (u, v, w) representing an edge from vertex u to vertex v with weight w.

Test Case 1:
Input:
n = 6
edges = [(0, 1, 7), (0, 2, 9), (0, 5, 14), (1, 2, 10), (1, 3, 15),
(2, 3, 11), (2, 5, 2), (3, 4, 6), (4, 5, 9)]
source = 0
target = 4
Output: 20
Test Case 2:
Input:
n = 5
edges = [(0, 1, 10), (0, 4, 3), (1, 2, 2), (1, 4, 4), (2, 3, 9), (3, 2, 7), (4, 1, 1), (4, 2, 8), (4, 3, 2)]
source = 0
target = 3
Output: 8

AIM: To write a pythin program for the The graph is represented as a list of edges where each edge is a tuple (u, v, w) representing an edge from vertex u to vertex v with weight w.

PROGRAM :

```
import heapq

def dijkstra(n, edges, source, target):
    # Create the adjacency list
    adj_list = {i: [] for i in range(n)}
    for u, v, w in edges:
        adj_list[u].append((v, w))
        adj_list[v].append((u, w))  # Assuming the graph is undirected

    distances = {i: float('inf') for i in range(n)}
    distances[source] = 0

    # Priority queue to process vertices
    pq = [(0, source)]  # (distance, vertex)
    heapq.heapify(pq)
```

```
    while pq:
        current_distance, current_vertex = heapq.heappop(pq)

        if current_vertex == target:
            return current_distance

        if current_distance > distances[current_vertex]:
            continue

        for neighbor, weight in adj_list[current_vertex]:
            distance = current_distance + weight

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

    return float('inf')

n1 = 6
edges1 = [(0, 1, 7), (0, 2, 9), (0, 5, 14), (1, 2, 10), (1, 3, 15),
          (2, 3, 11), (2, 5, 2), (3, 4, 6), (4, 5, 9)]
source1 = 0
target1 = 4
print("Test Case 1 Ou
tput:", dijkstra(n1, edges1, source1, target1))
```

OUTPUT:

```
Test Case 1 Output: 20
```

TIMECOMPLEXITY:

$O((V+E)\log V)$ $O((V+E)\log V)$.