In [5]:

```python
import heapq
import sys

class Graph:
    def __init__(self, vertices):
        self.V = vertices
        self.graph = [[] for _ in range(vertices)]

    def add_edge(self, u, v, weight):
        self.graph[u].append((v, weight))

    def dijkstra(self, src):
        distance = [sys.maxsize] * self.V
        distance[src] = 0

        pq = [(0, src)]

        while pq:
            dist_u, u = heapq.heappop(pq)

            if dist_u > distance[u]:
                continue

            for v, weight in self.graph[u]:
                if distance[v] > distance[u] + weight:
                    distance[v] = distance[u] + weight
                    heapq.heappush(pq, (distance[v], v))

        return distance

# Example usage:
g = Graph(5)
g.add_edge(0, 1, 2)
g.add_edge(0, 3, 6)
g.add_edge(1, 2, 3)
g.add_edge(1, 3, 8)
g.add_edge(1, 4, 5)
g.add_edge(2, 4, 7)
g.add_edge(3, 4, 9)

src = 0
distances = g.dijkstra(src)

print(f"Shortest distances from source vertex {src}:")
for i, dist in enumerate(distances):
    print(f"Vertex {i}: {dist}")



```

```
Shortest distances from source vertex 0:
Vertex 0: 0
Vertex 1: 2
Vertex 2: 5
Vertex 3: 6
Vertex 4: 7
```