In [20]:

```python
import sys

class Graph:
    def __init__(self, vertices):
        self.V = vertices
        self.graph = [[0 for _ in range(vertices)] for _ in range(vertices

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

    def min_distance(self, dist, spt_set):
        min_dist = sys.maxsize
        min_index = -1

        for v in range(self.V):
            if dist[v] < min_dist and not spt_set[v]:
                min_dist = dist[v]
                min_index = v

        return min_index

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

        for _ in range(self.V):
            u = self.min_distance(dist, spt_set)
            spt_set[u] = True

            for v in range(self.V):
                if (not spt_set[v] and self.graph[u][v] > 0 and
                        dist[u] != sys.maxsize and
                        dist[u] + self.graph[u][v] < dist[v]):
                    dist[v] = dist[u] + self.graph[u][v]

        return dist

# Example usage:
g = Graph(9)
g.add_edge(0, 1, 4)
g.add_edge(0, 7, 8)
g.add_edge(1, 2, 8)
g.add_edge(1, 7, 11)
g.add_edge(2, 3, 7)
g.add_edge(2, 8, 2)
g.add_edge(2, 5, 4)
g.add_edge(3, 4, 9)
g.add_edge(3, 5, 14)
g.add_edge(4, 5, 10)
g.add_edge(5, 6, 2)
g.add_edge(6, 7, 1)
g.add_edge(6, 8, 6)
g.add_edge(7, 8, 7)

src = 0
```

```python
distances = g.dijkstra(src)
print("Shortest distances from source vertex", src)
for i, dist in enumerate(distances):
    print("Vertex", i, ":", dist)
```

```
Shortest distances from source vertex 0
Vertex 0 : 0
Vertex 1 : 4
Vertex 2 : 12
Vertex 3 : 19
Vertex 4 : 21
Vertex 5 : 11
Vertex 6 : 9
Vertex 7 : 8
Vertex 8 : 14
```

In [ ]: