class Graph():

    def \_\_init\_\_(self, vertices):

        self.V = vertices

        self.graph = [[0 for column in range(vertices)]

                      for row in range(vertices)]

    def printSolution(self, dist):

        print("Vertex \t Distance from Source")

        for node in range(self.V):

            print(node, "\t\t", dist[node])

    # A utility function to find the vertex with

    # minimum distance value, from the set of vertices

    # not yet included in shortest path tree

    def minDistance(self, dist, sptSet):

        # Initialize minimum distance for next node

        min = 1e7

        # Search not nearest vertex not in the

        # shortest path tree

        for v in range(self.V):

            if dist[v] < min and sptSet[v] == False:

                min = dist[v]

                min\_index = v

        return min\_index

    # Function that implements Dijkstra's single source

    # shortest path algorithm for a graph represented

    # using adjacency matrix representation

    def dijkstra(self, src):

        dist = [1e7] \* self.V

        dist[src] = 0

        sptSet = [False] \* self.V

        for cout in range(self.V):

            # Pick the minimum distance vertex from

            # the set of vertices not yet processed.

            # u is always equal to src in first iteration

            u = self.minDistance(dist, sptSet)

            # Put the minimum distance vertex in the

            # shortest path tree

            sptSet[u] = True

            # Update dist value of the adjacent vertices

            # of the picked vertex only if the current

            # distance is greater than new distance and

            # the vertex in not in the shortest path tree

            for v in range(self.V):

                if (self.graph[u][v] > 0 and

                   sptSet[v] == False and

                   dist[v] > dist[u] + self.graph[u][v]):

                    dist[v] = dist[u] + self.graph[u][v]

        self.printSolution(dist)

# Driver program

g = Graph(9)

g.graph = [[0, 4, 0, 0, 0, 0, 0, 8, 0],

           [4, 0, 8, 0, 0, 0, 0, 11, 0],

           [0, 8, 0, 7, 0, 4, 0, 0, 2],

           [0, 0, 7, 0, 9, 14, 0, 0, 0],

           [0, 0, 0, 9, 0, 10, 0, 0, 0],

           [0, 0, 4, 14, 10, 0, 2, 0, 0],

           [0, 0, 0, 0, 0, 2, 0, 1, 6],

           [8, 11, 0, 0, 0, 0, 1, 0, 7],

           [0, 0, 2, 0, 0, 0, 6, 7, 0]

           ]

g.dijkstra(0)