**Dijkstra's CODE -**

# Dijkstra's Algorithm in Python

import sys

# Providing the graph

vertices = [[0, 0, 1, 1, 0, 0, 0],

            [0, 0, 1, 0, 0, 1, 0],

            [1, 1, 0, 1, 1, 0, 0],

            [1, 0, 1, 0, 0, 0, 1],

            [0, 0, 1, 0, 0, 1, 0],

            [0, 1, 0, 0, 1, 0, 1],

            [0, 0, 0, 1, 0, 1, 0]]

edges = [[0, 0, 1, 2, 0, 0, 0],

         [0, 0, 2, 0, 0, 3, 0],

         [1, 2, 0, 1, 3, 0, 0],

         [2, 0, 1, 0, 0, 0, 1],

         [0, 0, 3, 0, 0, 2, 0],

         [0, 3, 0, 0, 2, 0, 1],

         [0, 0, 0, 1, 0, 1, 0]]

# Find which vertex is to be visited next

def to\_be\_visited():

    global visited\_and\_distance

    v = -10

    for index in range(num\_of\_vertices):

        if visited\_and\_distance[index][0] == 0 \

            and (v < 0 or visited\_and\_distance[index][1] <=

                 visited\_and\_distance[v][1]):

            v = index

    return v

num\_of\_vertices = len(vertices[0])

visited\_and\_distance = [[0, 0]]

for i in range(num\_of\_vertices-1):

    visited\_and\_distance.append([0, sys.maxsize])

for vertex in range(num\_of\_vertices):

    # Find next vertex to be visited

    to\_visit = to\_be\_visited()

    for neighbor\_index in range(num\_of\_vertices):

        # Updating new distances

        if vertices[to\_visit][neighbor\_index] == 1 and \

                visited\_and\_distance[neighbor\_index][0] == 0:

            new\_distance = visited\_and\_distance[to\_visit][1] \

                + edges[to\_visit][neighbor\_index]

            if visited\_and\_distance[neighbor\_index][1] > new\_distance:

                visited\_and\_distance[neighbor\_index][1] = new\_distance

        visited\_and\_distance[to\_visit][0] = 1

i = 0

# Printing the distance

for distance in visited\_and\_distance:

    print("Distance of ", chr(ord('a') + i),

          " from source vertex: ", distance[1])

    i = i + 1

**OUTPUT –**

