**Assignment 10**

**Name: Mudit Mohit**

**Enrolment No.: BT21GCS057**

**Section: B2**

**Question:** Implement the Dijkstra algorithm to solve the problem of finding the shortest path in a graph from a single source node to all other nodes (ex-single source shortest path problem). Assume that there are n nodes in the graph and that they have associated costs (which may be positive or negative). Analyze the complexity of the algorithm and calculate the optimal path and its cost value.

Ans.

Code:

import sys

class Graphes():

    def \_\_init\_\_(self,vertex):

        self.v = vertex

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

                      for row in range(vertex)]

    def print\_Solution(self, weight):

        print("Vertex \tSource Distance")

        for node in range(self.v):

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

    def min\_Distance(self, weight, sptset):

        min = sys.maxsize

        for u in range(self.v):

            if weight[u] < min and sptset[u] == False:

                min = weight[u]

                min\_index = u

        return min\_index

    def dijktras(self, src):

        weight = [sys.maxsize]\* self.v

        weight[src] = 0

        sptset = [False]\*self.v

        for cout in range(self.v):

            x = self.min\_Distance(weight, sptset)

            sptset[x] = True

            for y in range(self.v):

                if self.graph[x][y] > 0 and sptset[y] == False and \

                        weight[y] > weight[x] + self.graph[x][y]:

                    weight[y] = weight[x] + self.graph[x][y]

        self.print\_Solution(weight)

if \_\_name\_\_ == "\_\_main\_\_":

    g = Graphes(4)

    g.graph = [[0,8,0,0],

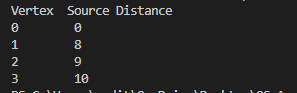
               [8,0,1,6],

               [0,1,6,1],

               [0,0,1,4]]

    g.dijktras(0)

Output:



Analysis:

Time-complexity: O(V^2)

Space complexity: O(V)