**IMPLEMENTATION – EX 1 :**

import sys

class Coords:

def \_\_init\_\_(self,x,y):

self.x = x

self.y = y

def dist(self,other):

s = (self.x - other.x)\*\*2 + (self.y - other.y)\*\*2

return s\*\*0.5

def \_\_str\_\_(self):

return '(' + str(self.x) + ',' + str(self.y) + ')'

class Graph:

def \_\_init\_\_(self,d,demand\_lst=None):

self.d = d

lst = [x for x in d.keys()]

self.lst = lst

if demand\_lst is None:

demand\_lst = [1] \* len(lst)

self.demand\_lst = demand\_lst

def min\_dist\_points(self,a,b):

mpoint = ''

mval = a.dist(b[0])

for i in b:

if mval < a.dist(b[i]):

mval = a.dist(b[i])

mpoint = b[i]

return mpoint

def adjmatrix(self):

adjmatrixlist = []

for i in self.lst:

sidelst = []

ct = 0

for j in self.lst:

if i != j:

sidelst.append(i.dist(j) \* self.demand\_lst[ct])

ct += 1

else:

sidelst.append(0)

adjmatrixlist.append(sidelst)

return adjmatrixlist

def printval(self,ind):

value = self.lst[ind]

return self.d[value]

def prim\_mst(self):

adj\_matrix = self.adjmatrix()

n = len(adj\_matrix)

parent = [0] \* n

key = [sys.maxsize] \* n

key[0] = 0

mst\_set = [False] \* n

for \_ in range(n):

min\_key = sys.maxsize

min\_index = -1

for v in range(n):

if key[v] < min\_key and not mst\_set[v]:

min\_key = key[v]

min\_index = v

mst\_set[min\_index] = True

for v in range(n):

if adj\_matrix[min\_index][v] and not mst\_set[v] and adj\_matrix[min\_index][v] < key[v]:

parent[v] = min\_index

key[v] = adj\_matrix[min\_index][v]

total\_cost = 0

dictionary = {}

for i in range(1, n):

val1 = a.printval(i)

val2 = a.printval(parent[i])

if str(val2) not in dictionary.keys():

dictionary[str(val2)] = [str(val1)]

else:

dictionary[str(val2)].append(str(val1))

total\_cost += adj\_matrix[i][parent[i]]

print(f"Distance between {val1} and {val2}: {adj\_matrix[i][parent[i]]}")

return (total\_cost,dictionary)

#driver code

if \_\_name\_\_ == '\_\_main\_\_':

#The code provided here will not be executed when imported

p1 = Coords(1,2)

p2 = Coords(3,4)

p3 = Coords(5,6)

p4 = Coords(0,2)

p5 = Coords(0,0)

p6 = Coords(1,1)

d = {p1 : 'Chennai', p2 : 'Cuddalore', p3 : 'Kanchipuram', p4 : 'Vellore', p5 : 'Kanyakumari',p6 : 'Madurai'}

a = Graph(d)

print()

#printing the shortest distances between each city and path

minimum\_cost,dictpaths = a.prim\_mst()

print()

#printing total cost of tree

print("Total cost is :",minimum\_cost)

print()

#printing all the paths

print("Paths to each point is :",dictpaths)

print()

**OUTPUT:**

**Distance between Cuddalore and Chennai: 2.8284271247461903**

**Distance between Kanchipuram and Cuddalore: 2.8284271247461903**

**Distance between Vellore and Chennai: 1.0**

**Distance between Kanyakumari and Madurai: 1.4142135623730951**

**Distance between Madurai and Chennai: 1.0**

**Total cost is : 9.071067811865476**

**Paths to each point is : {'Chennai': [' Cuddalore', 'Vellore', 'Madurai'], ' Cuddalore': ['Kanchipuram'], 'Madurai': ['Kanyakumari']}**