1.

def add\_vertex(v):

global graph

global vertices\_no

global vertices

if v in vertices:

print("Vertex", v, "already exists.")

else:

vertices\_no = vertices\_no + 1

vertices.append(v)

if vertices\_no >1:

for vertex in graph:

vertex.append(0)

temp = []

for i in range(vertices\_no):

temp.append(0)

graph.append(temp)

def add\_edge(v1, v2, e):

global graph

global vertices\_no

global vertices

if v1 not in vertices:

print("Vertex", v1, "does not exist.")

elif v2 not in vertices:

print("Vertex", v2, "does not exist.")

else:

index1 = vertices.index(v1)

index2 = vertices.index(v2)

graph[index1][index2] = e

vertices = []

vertices\_no = 0

graph = []

add\_vertex("A")

add\_vertex("B")

add\_vertex("C")

add\_edge('A', 'B', 5)

add\_edge('A', 'C', 6)

add\_edge('B', 'A', 10)

add\_edge('C', 'A', 3)

add\_edge('C', 'B', 2)

print(graph)

2.

class DisjointSet(dict):

def \_\_init\_\_(self, dict):

pass

def add(self, item):

self[item] = item

def find(self, item):

if self[item] != item:

self[item] = self.find(self[item])

return self[item]

def unionset(self, item1, item2):

self[item2] = self[item1]

def Kruskal(nodes, edges):

all\_nodes = nodes

used\_nodes = set()

MST = []

edges = sorted(edges, key=lambda element: element[2], reverse=True)

while used\_nodes != all\_nodes and edges:

element = edges.pop(-1)

if element[0] in used\_nodes and element[1] in used\_nodes:

continue

MST.append(element)

used\_nodes.update(element[:2])

return MST

def main():

nodes = set(list('ABCDEF'))

edges = [("A", "B", 6), ("A", "E", 10),

("A", "F", 12), ("B", "C", 3),

("B", "D", 5), ("B", "F", 8),

("C", "D", 7), ("D", "E", 9),

("D", "F", 11), ("E", "F", 16)]

print("The minimum spanning tree by Kruskal is : ")

print(Kruskal(nodes, edges))

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

main()