Data Analysis and Algorithm

Practical 7

Write a program to implement Krushkals algorithm.

Date.: 19-10-21

Name – Yash Vasudeo Prajapati

Rollno - 022

MSc. Computer Science

Date ___ / ___ / ___

7) Write a Program to implement Kruskal's Algorithm.

Theory :- Kruskal's Algorithm is a Minimum Spanning tree algorithm that takes a graph as input & finds the subset of the edge of the graph which

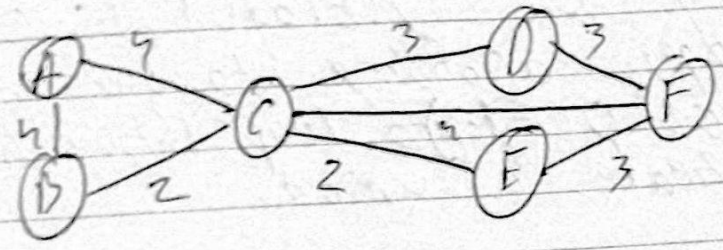
1) forms a tree that includes all vertex

2) has the minimum sum of weights among all the trees that can be formed.

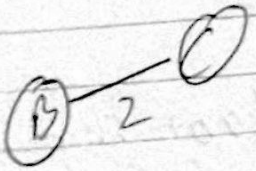
Kruskal's algorith is a greedy algorithm.

Example

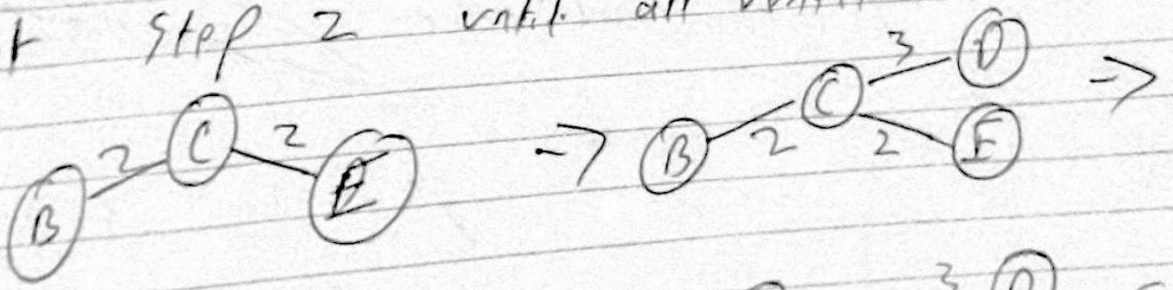
1) Start with a weighted graph.

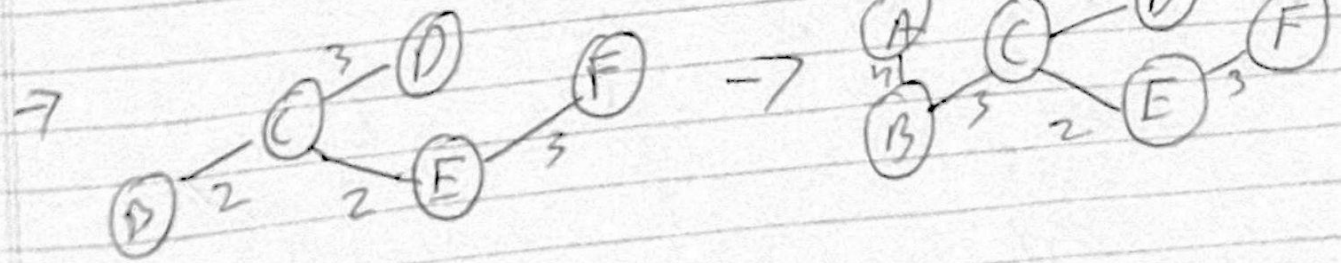


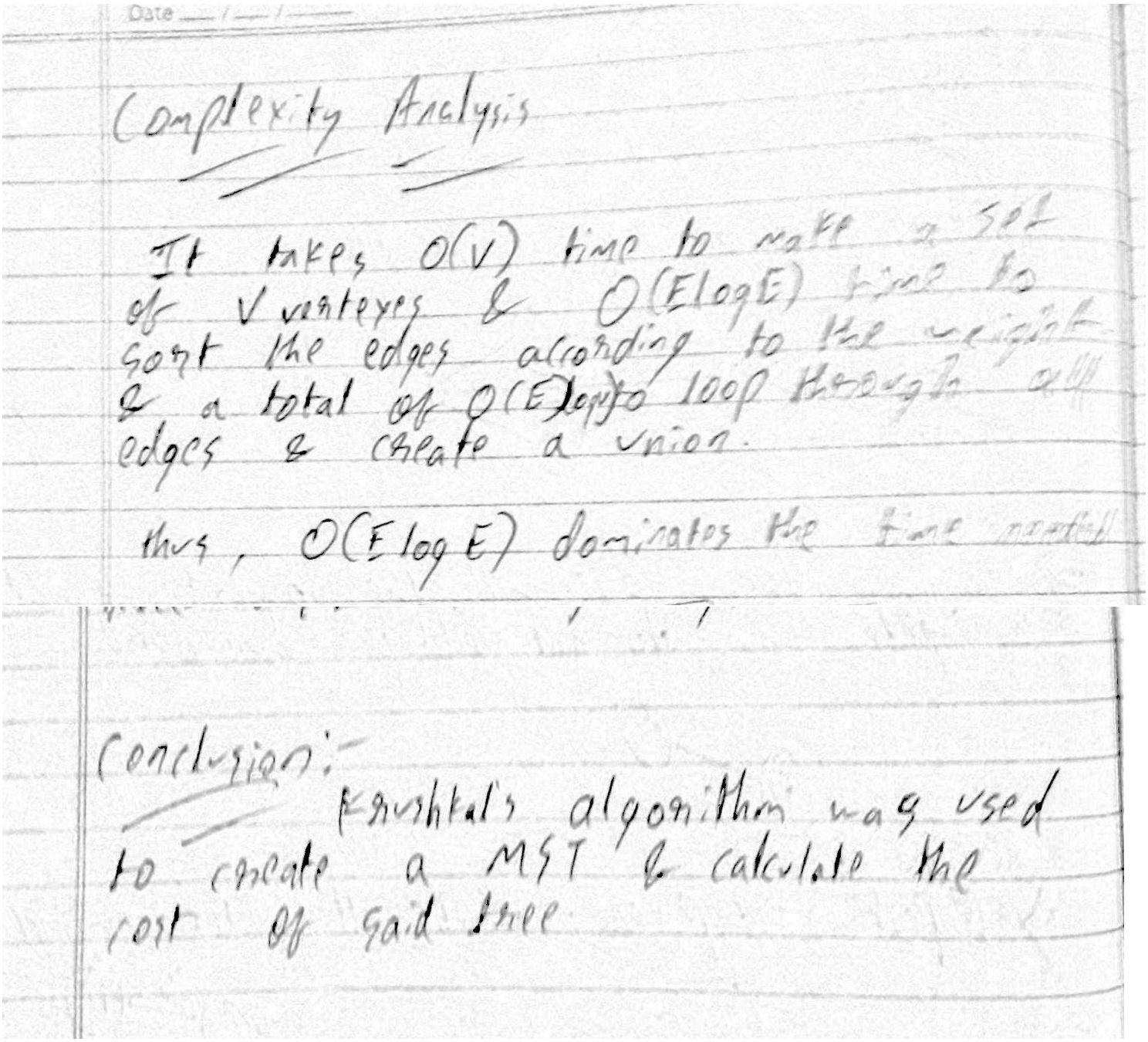
2) consider the edge with the lowest weight & add to the tree if it form a cycle then do not add & move on.



3) Repeat step 2 until all vertex are included







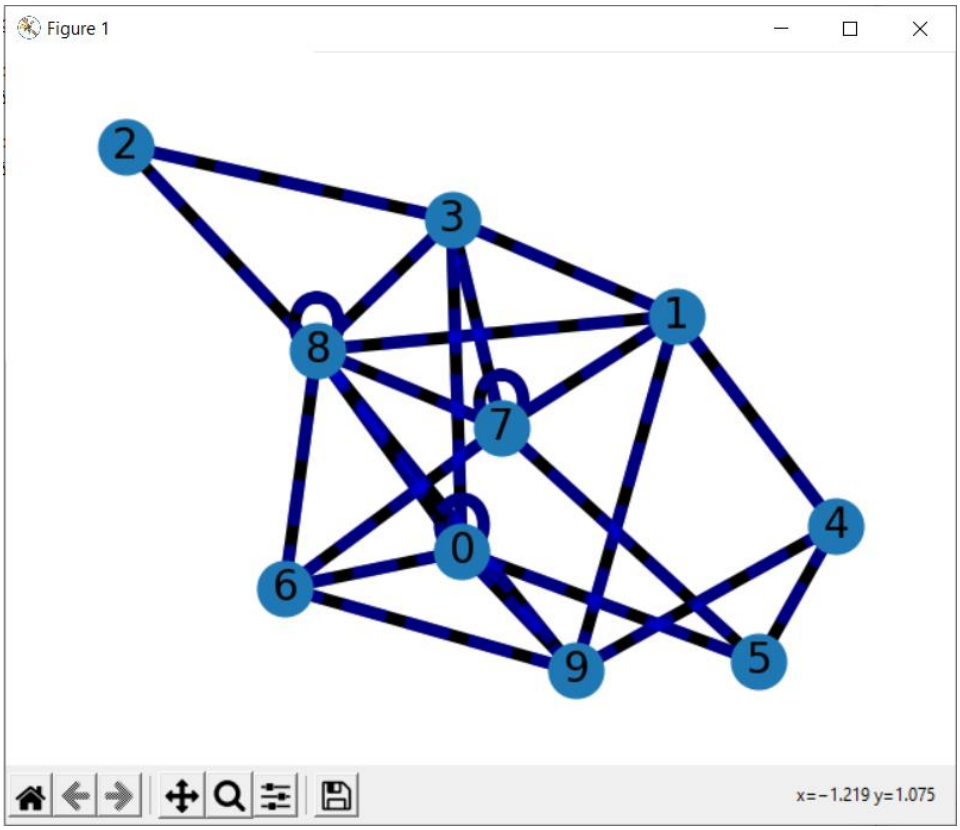
Program

```
import matplotlib.pyplot as plt
import networkx as nx
import random

class Graph:
    def __init__(self,vertex):
        self.v = vertex
        self.graph = []

    def add_edge(self, source, destination, weight):
        self.graph.append([source, destination, weight])

    def algo(self):
        parent, rank, result = [], [], []
        i,e = 0,0
        self.graph = sorted(self.graph, key=lambda item:item[2])
        for node in range(self.v):
            parent.append(node)
            rank.append(0)
```

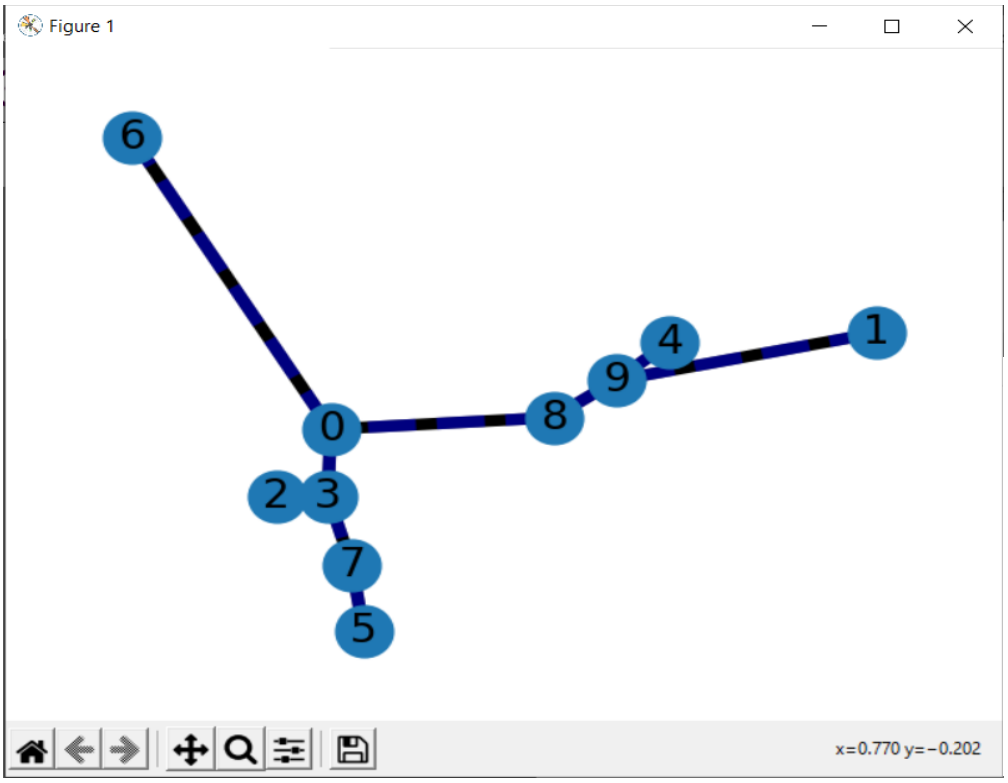
```
        while e < self.v -1:
            s, d, w = self.graph[i]
            i += 1
            x = self.find(parent, s)
            y = self.find(parent, d)
            if x != y:
                e += 1
                result.append([s, d, w])
                self.union(parent, rank, x, y)
        return result

    def find(self, p, i):
        if p[i] ==i:
            return i
        return self.find(p, p[i])


    def union(self, p, r, x, y):
        s = self.find(p, x)
        d = self.find(p, y)
        if r[s] < r[d]:
            p[s] = d
        elif r[s] > r[d]:
            p[d] = s
        else:
            p[d] = s
            r[s] += 1

def plot(G):

    pos = nx.spring_layout(G, seed=7)
    nx.draw_networkx_nodes(G, pos, node_size=700)


    # edges
    nx.draw_networkx_edges(G, pos, width=6)
    nx.draw_networkx_edges(
        G, pos, width=6, alpha=0.5, edge_color="b", style="dashed"
    )

    # labels
    nx.draw_networkx_labels(G, pos, font_size=20, font_family="sans-serif")

    ax = plt.gca()
    ax.margins(0.08)
    plt.axis("off")
    plt.tight_layout()
    plt.show()



if __name__ == "__main__":

    graph = Graph(10)
    for i in range(0 ,random.randint(19,50)):
        graph.add_edge(random.randint(0,9),random.randint(0,9), random.randint(0,9))
    result = graph.algo();

    G = nx.Graph()
    for u, v, w in graph.graph:
        G.add_edge(u, v, weight=w)

    plot(G)

    G = nx.Graph()
    for u, v, w in result:
        G.add_edge(u, v, weight=w)
        print("%d - %d: %d" % (u, v, w))
    plot(G)
```

Output:-

Original



MST:-




```
ushkals.py
8 - 0: 0
1 - 9: 0
6 - 0: 0
8 - 9: 1
9 - 4: 1
3 - 2: 2
3 - 0: 3
7 - 5: 3
7 - 3: 3
>>>
```