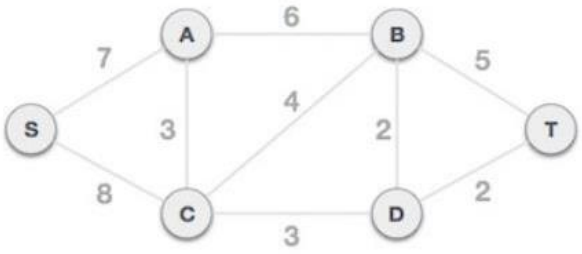
S Abhishek AM.EN.U4CSE19147

Data Structures and Algorithms – Assignment 9

Graph Applications

1. Implement Kruskal's minimum spanning tree algorithm. Use the following graph as input.




```
class s:
    def __init__(self, v):
        self.V = v
        self.s = []

    def add(self, u, v, w):
        self.s.append([u, v, w])

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

    def merge(self, root, count, x, y):
        x = self.find(root, x)
        y = self.find(root, y)
        if count[x] < count[y]:
            root[x] = y
        elif count[x] > count[y]:
            root[y] = x
```

```
        else:
            root[y] = x
            count[x] += 1

    def min_span(self):
        arr = []
        i, j = 0, 0
        self.s = sorted(self.s, key=lambda item: item[2])
        root = []
        count = []
        for node in range(self.V):
            root.append(node)
            count.append(0)
        while j < self.V - 1:
            u, v, w = self.s[i]
            i = i + 1
            x = self.find(root, u)
            y = self.find(root, v)
            if x != y:
                j = j + 1
                arr.append([u, v, w])
                self.merge(root, count, x, y)
        for u, v, val in arr:
            print("\t\t Edge : (", u, v, ")", end=" ")
            print("-", val)


if __name__ == '__main__':
    root = s(6)
    root.add(0, 1, 7)
    root.add(0, 3, 8)
    root.add(1, 2, 6)
    root.add(1, 3, 3)
    root.add(2, 4, 2)
    root.add(2, 5, 5)
    root.add(3, 2, 4)
    root.add(3, 4, 3)
```

```
root.add(4, 5, 2)

print("\n\t\tMinimum Spanning Tree\n")
root.min_span()
```

```
Minimum Spanning Tree

 Edge : ( 2 4 ) - 2
 Edge : ( 4 5 ) - 2
 Edge : ( 1 3 ) - 3
 Edge : ( 3 4 ) - 3
 Edge : ( 0 1 ) - 7
```

2. Read up on the travelling salesman problem and describe in your own words an algorithm to solve this – you need not code this. Just a short algorithmic writeup is needed.

- This is the problem facing a salesman who needs to travel to a number of cities and get back home.
- Given the cities (and their locations), the challenge is to find the shortest possible route that he can follow to visit each city exactly once and return to the origin city.
- For Example: **You will start to visit from your work place to all the villages in your city. So, guess what data I have while visiting a particular village?**
- Yes, the distance from the office to first village where you are going to visit and the other distances, you are going to visit next.

- So, like this from first village to other you will know all the distances.
- Hence, in this way you can take multiple routes to reach from one village to other, the motion is random.
- So, our main goal is to find the shortest path. The lines joined is the final shortest path from your office to all the villages you covered.
- We can use any particular algorithm like Dynamic Program to trace out the optimal route of the salesman in order to get the data.
- It has many applications in Computer Science and corporates to get data.

One Drive: Click Me!!

Thankyou!