DAA – Lab7

Vishal Gauba

1410110501

**Input:**

* N (number of nodes)
* M (number of edges)

**Output:**

* Randomly generated connected weighted graph
* Minimum Spanning Tree (using prim’s)

**Screenshots:**

**Source Code:**

from random import randint

def generate\_graph(n, m):

connected=0

while(connected==0):

cost = [[1000 for i in range(n)] for j in range(n)]

j=0

while(j<m):

fromm = randint(0,4)

to = randint(0,4)

if(fromm==to or cost[fromm][to] != 1000):

j = j-1

continue

cost[fromm][to] = randint(3,10)

cost[to][fromm] = cost[fromm][to]

j = j + 1

for i in range(n):

if(cost[i]==[1000]\*n):

connected=0

break;

connected=1

print (cost)

return cost

# returns index at which min value is in the array

def findmin(arr):

min = 0; pos = -1;

for x in range(5):

if(arr[x] < arr[min]):

min = x

return min;

# returns index of min array at which min value is in the cost array

def findmin2(arr):

min = 0; pos = 0;

for x in range(5):

if(cost[x][arr[x]] < cost[min][arr[min]]):

min = x

return min;

'''

Until all nodes are added to the tree,

find the minimum weight attached to all nodes already in the tree,

then find minimum of those and add to the tree.

'''

def prims(cost, n):

traversed = [0]\*n

traversed[0] = 1

count=1

total\_weight=0

while(count<n):

min = [0]\*n

for i in range(n):

if(traversed[i]==1):

min[i] = findmin(cost[i])

fromm = findmin2(min)

to = min[fromm]

if(traversed[to]==1):

cost[fromm][to]=1000

cost[to][fromm]=1000

if(traversed[to]!=1):

print("edge: "+str(fromm)+" -> "+str(to)+ ", weight="+str(cost[fromm][to]))

total\_weight = total\_weight + cost[fromm][to]

traversed[to] = 1

count = count + 1

cost[fromm][to] = 1000

cost[to][fromm] = 1000

print(total\_weight)

/\* Main \*/

n=5

m=10

cost = generate\_graph(n,m)

prims(cost, n)

#print(generate\_graph(5,5))