

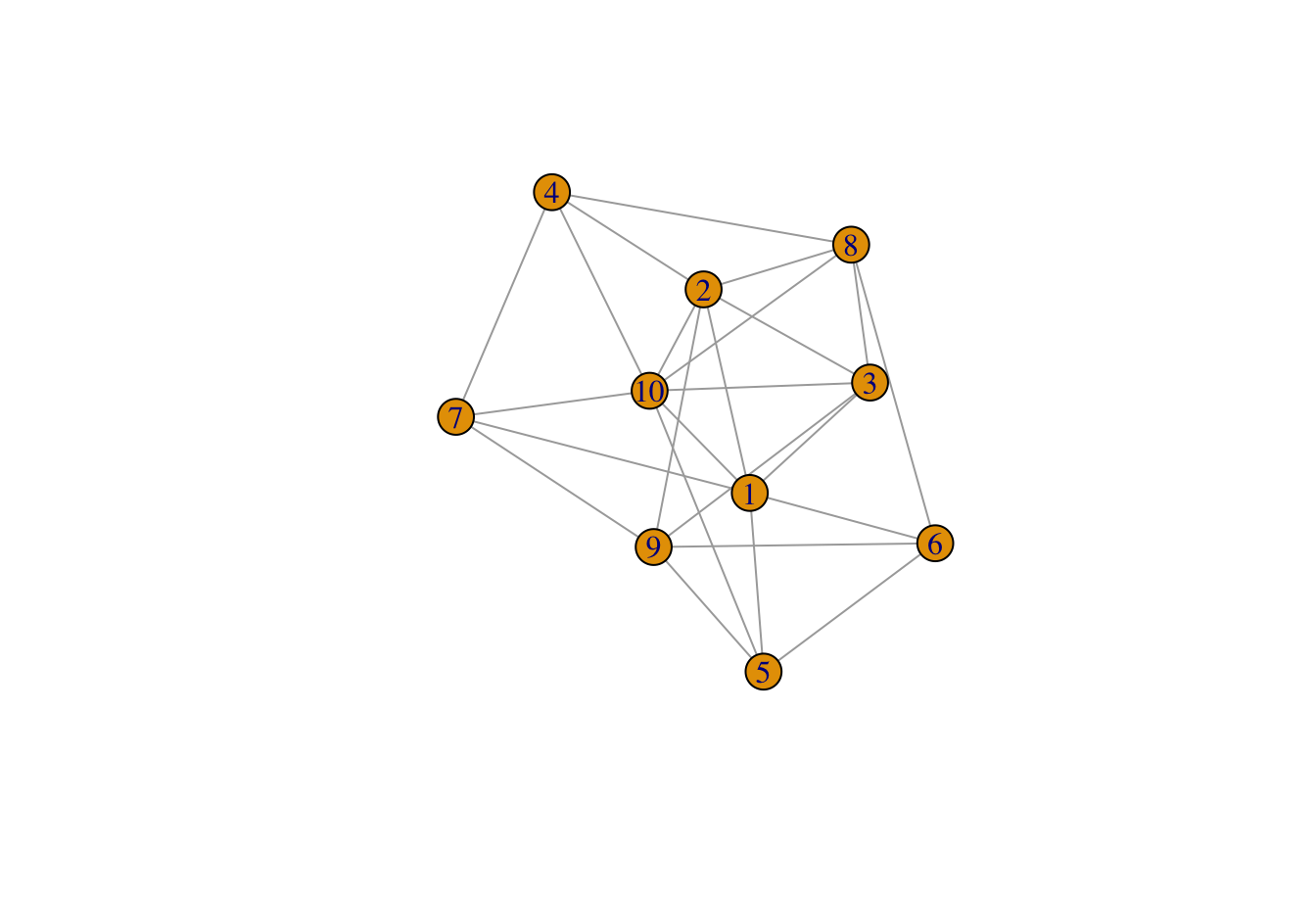
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Programme | : | **B.Tech** | Semester | : | **Win Sem 21-22** |
| Course | : | **Web Mining** | Code | : | **CSE3024** |
| Faculty | : | **Dr.Bhuvaneswari A** | Slot | : | **L7+L8** |
| Date | : | **8-03-2022** | Marks | : | **10 Points** |

NAME: RAHUL GARG  
REGISTER NUMBER: 19BCE1431

**Exercise 8:CENTRALITY METRICS – SOCIAL NETWORK ANALYSIS**

**GOOGLE COLLAB LINK: https://colab.research.google.com/drive/10FrxFD91ctlhA1Q84LFqQD4KUSgvzkmY?usp=sharing**

**Consider the following Facebook friendship network (mutual connection)**



1. Find the degree centrality of all nodes
2. Find the neighbors of node 2.
3. Find the average degree of graph
4. Find the density of the graph
5. Find the closeness centrality of Node 10
6. Find all the paths to reach 4 from 6
7. Find the longest shortest path between any two nodes
8. Find the shortest path between any two nodes
9. Find the Betweenness centrality of Node 1.
10. Find the person who has maximum number of connections (friends)

CODE:

import networkx as nx

import matplotlib.pyplot as plt

from collections import Counter

import statistics

G = nx.DiGraph()

G.add\_edges\_from([('1', '2'), ('1', '3'), ('1', '5'), ('1', '6'), ('1', '7'), ('1', '10'),

 ('2', '3'), ('2', '4'),('2', '8'),

 ('2', '9'), ('2', '10'),('3', '8'),

 ('3', '10'), ('3', '2'),('3', '9'),

 ('4', '2'), ('4', '10'),('4', '7'),

 ('4', '8'), ('5', '1'),('5', '6'),

 ('5', '9'), ('5', '10'),

 ('6', '1'), ('6', '5'),('6', '8'),

 ('6', '9'), ('7', '9'),('7', '10'),('8', '10')])

G = G.to\_undirected()

plt.figure(figsize =(10, 10))

nx.draw\_networkx(G, with\_labels = True)

# DEGREE CENTRALITY

d=nx.degree\_centrality(G)

mean\_arr = []

for i,k in enumerate(d):

 mean\_arr.append(d[k])

 print('Degree centrality of the nodes are : ')

print(d)

print()

#neighbours of node 2

print('neighbours of node 2 are : ')

print(list(G.neighbors('2')))

print()

#average degree of the graph

x = statistics.mean(mean\_arr)

print("average degree is :", x)

print()

#density of the graph

print('density of the graph : ')

density = nx.density(G)

print(density)

print()

#closeness centrality of node 10

print('closeness centrality of node 10 is : ')

closeness\_centrality = nx.closeness\_centrality(G)

print(closeness\_centrality['10'])

print()

#path to reach from node 4 to 6

print('path to reach from node 4 to 6 : ')

paths = nx.all\_simple\_paths(G, source='4', target='6')

print(list(paths))

print()

#the longest shortest path between any two nodes

#print('shortest path between any 2 nodes : ')

#print(nx.longest\_path(G, source='4', target='6'))

print('the longest shortest path between any two nodes : ')

paths = list(nx.all\_simple\_paths(G, source='4', target='6'))

pathLengths = [len(i) for i in paths]

index = pathLengths.index(max(pathLengths))

print(paths[index])

print()

#shortest path between any 2 nodes

print('shortest path between any 2 nodes : ')

print(nx.shortest\_path(G, source='4', target='6'))

print()

#betweenness centrality of node 1

print('betweenness centrality of node 1 : ')

betweenness\_centrality = nx.betweenness\_centrality(G)

print(betweenness\_centrality['1'])

print()

#person with maximum number of connections

degreeView = G.degree()

degree\_counts = Counter(dict(degreeView ))

max\_degree\_node = degree\_counts.most\_common(1)

print('person with maximum number of connectionsv: ')

print(max\_degree\_node)

print()

OUTPUT:

