Assignment3
Code Part

Shrinidhi M
192IT024

1. Degree Centrality

```
import operator
Degree_centrality=nx.degree_centrality(G1) # the fuction
ddegree_centrality(G) Compute the degree centrality for nodes and
returns Dictionary of nodes with degree centrality as the value.
#Degree_centrality #Dictionary that contains Degree Centrality
Values
minimum=sorted(Degree_centrality.keys(), key=lambda
key:Degree_centrality[key], reverse=False)[0]
maximum=sorted(Degree_centrality.keys(),key=lambda
key:Degree_centrality[key], reverse=True)[0]
print("Minimum degree centrality node and
value",minimum,Degree_centrality[minimum]) #minimum degree
centrality
print("Maximum degree centrality node and
value",maximum,Degree_centrality[maximum]) #maximum degree
centrality
print("Nodes in Decreasing order of Degree Centrality(top 5
nodes):\n")
sorted(Degree_centrality.items(),key=operator.itemgetter(1),reverse=
True)[0:5]
```

2. Closeness Centrality

```
Closeness_Centrality=nx.closeness_centrality(G1) #the function
closeness_centrality(G, u=None, distance=None, normalized=True)
Compute closeness centrality for nodes and Dictionary of nodes with
closeness centrality as the value.
#Closeness_Centrality #Dictionary of nodes with closeness centrality
as the value.
minimum=sorted(Closeness_Centrality.keys(), key=lambda
key:Closeness_Centrality[key], reverse=False)[0]
```

```
maximum=sorted(Closeness_Centrality.keys(), key=lambda
key:Closeness_Centrality[key], reverse=True)[0]
print("Minimum Closeness centrality node and
value",minimum,Closeness_Centrality[minimum]) #minimum Closeness
Centrality
print("Maximum Closeness centrality node and
value",maximum,Closeness_Centrality[maximum]) #maximum Closeness
Centrality
print("Modes in Decreasing order of Closeness Centrality(top 5
nodes):\n")
sorted(Closeness_Centrality.items(),key=operator.itemgetter(1),rever
se=True)[0:5]
```

3. Betweenness Centrality

```
Betweenness_Centrality=nx.betweenness_centrality(G1,normalized=True,
weight=None, endpoints=False, seed=None) #The funcion
betweenness_centrality(G, k=None, normalized=True, weight=None,
endpoints=False, seed=None) computes Compute the shortest-path
betweenness centrality for nodes and returns Dictionary of nodes
with betweenness centrality as the value.
#Betweenness_Centrality #Dictionary of nodes with betweenness
centrality as the value.
minimum=sorted(Betweenness_Centrality.keys(), key=lambda
key:Betweenness_Centrality[key], reverse=False)[0]
maximum=sorted(Betweenness_Centrality.keys(), key=lambda
key:Betweenness_Centrality[key], reverse=True)[0]
print("Minimum Betweenness centrality node and
value",minimum,Betweenness_Centrality[minimum])
print("Maximum Betweenness centrality node and
value",maximum,Betweenness_Centrality[maximum])
print("Nodes in Decreasing order of Betweenness Centrality(top 5
nodes):\n")
sorted(Betweenness_Centrality.items(),key=operator.itemgetter(1),rev
erse=True)[0:5]
```

4. Eigen Vector Centrality

```
Eigenvector_Centrality=nx.eigenvector_centrality(G1, max_iter=100,
tol=1e-06, nstart=None, weight=None) #The function
eigenvector_centrality(G, max_iter=100, tol=1e-06, nstart=None,
weight='weight') Compute the eigenvector centrality for the graph G
and return Dictionary of nodes with eigenvector centrality as the
value.
#Eigenvector_Centrality #Dictionary of nodes with eigenvector
centrality as the value.
minimum=sorted(Eigenvector_Centrality.keys(), key=lambda
key:Eigenvector_Centrality[key], reverse=False)[0]
maximum=sorted(Eigenvector_Centrality.keys(), key=lambda
key:Eigenvector_Centrality[key], reverse=True)[0]
print("Minimum degree centrality node and
value",minimum,Eigenvector_Centrality[minimum])
print("Maximum degree centrality node and
value",maximum,Eigenvector_Centrality[maximum])
print("Nodes in Decreasing order of Eigenvector Centrality(top 5
nodes):\n")
sorted(Eigenvector_Centrality.items(),key=operator.itemgetter(1),rev
erse=True)[0:5]
```

5. HITS- Hub Score

```
hubs1, authorities1 = nx.hits(G1, max_iter = 50, normalized = True)
maximum=sorted(hubs1.keys(), key=lambda key:hubs1[key],
reverse=True)[0]
minimum=sorted(hubs1.keys(), key=lambda key:hubs1[key],
reverse=False)[0]
print("Maximum hub score webpage and value",maximum,hubs1[maximum])
#maximum hub score webpage
print("Minimum hub score webpage and
value",minimum,hubs1[minimum])#minimum hub score webpage
print("Nodes in Decreasing order of hub score (top 5 nodes):")
print(sorted(hubs1.items(),key=operator.itemgetter(1),reverse=True)[
0:5])
```

6. HITS- Authoritative Score

```
maximum=sorted(authorities1.keys(), key=lambda
key:authorities1[key], reverse=True)[0]
minimum=sorted(authorities1.keys(), key=lambda
key:authorities1[key], reverse=False)[0]
print("Maximum authority score webpage and
value",maximum,authorities1[maximum])#maximum authority score
webpage
print("Minimum authority score webpage and
value",minimum,authorities1[minimum])#minimum authority score
webpage
print("Nodes in Decreasing order of autority score (top 5
nodes):\n")
sorted(authorities1.items(),key=operator.itemgetter(1),reverse=True)
[0:5]
```

7. Page Rank

```
pr1 = nx.pagerank(G1, alpha=0.85)
#print("page rank:G1",pr1) #page rank of all the webpages
maximum=sorted(pr1.keys(), key=lambda key:pr1[key], reverse=True)[0]
minimum=sorted(pr1.keys(), key=lambda key:pr1[key],
reverse=False)[0]
print("Maximum page rank webpage",maximum,pr1[maximum]) # maximum
page rank webpage
print("Minimum page rank webpage",minimum,pr1[minimum])#minimum page
rank webpage
print("Nodes in Decreasing order of Page Rank (top 5 nodes):\n")
sorted(pr1.items(),key=operator.itemgetter(1),reverse=True)[0:5]
```