Ayush Mahant
Palak Goel

IR_ASSIGNMENT 3_75

Question 1 - [45 Points] Link Analysis

Pick a real-world network dataset (with number of nodes > 100) from here.

[2 points] Represent the network in terms of its 'adjacency matrix' as well as 'edge list'.

[28 points] Briefly describe the dataset chosen and report the following:

1. Number of Nodes
2. Number of Edges
3. Avg In-degree
4. Avg. Out-Degree
5. Node with Max In-degree
6. Node with Max out-degree
7. The density of the network

Further, perform the following tasks:

1. [5 points] Plot degree distribution of the network (in case of a directed graph, plot in-degree and out-degree separately).
2. [10 points] Calculate the local clustering coefficient of each node and plot the clustering-coefficient distribution of the network.

Dataset picked- wiki-Vote
https://snap.stanford.edu/data/wiki-Vote.html

Description of the dataset-

Dataset statistics	
Nodes	7115
Edges	103689
Nodes in largest WCC	7066 (0.993)
Edges in largest WCC	103663 (1.000)
Nodes in largest SCC	1300 (0.183)
Edges in largest SCC	39456 (0.381)
Average clustering coefficient	0.1409
Number of triangles	608389
Fraction of closed triangles	0.04564
Diameter (longest shortest path)	7
90-percentile effective diameter	3.8

Wikipedia is described as a free encyclopedia whose contributors include their administrators. These are specific users with additional tech features. If a specific user wants to be an administrator, he/she needs to sent a request for adminship. Thewikipedia community then decides through voting who needs to be promoted. About 2,794 and 103,663 votes with about 7066 users have participated in these elections. This network contains the wikipedia voting data from wikipeida, till Jan,2008. The nodes in the network are the users, and edge directed from node i to j, represents user i casted vote for user j.

Resource- https://snap.stanford.edu/data/wiki-Vote.html

1. Number of Nodes-
7115
Stored all the nodes from the file, both the FromNodeId and ToNodeId,
In two lists.
Stored all the unique values from both the lists, changed them into sets, as sets would return all the unique values, and simply performed logical OR operation on both the sets, And stored them in a set named ‘fin_nodes’.

The length of the fin_nodes is the total number of nodes in the graph.
Number of nodes= len(fin_nodes)

2. Number of Edges-

```
103689
```

The number of edges is simply the length of the dataset.
= len(data)

3. Avg In-degree-

```
14.573295853829936
```

Calculatted using two formulas-
i)

```
 len(data)/len(fin_nodes))
```

It is the number of edges divided by the number of nodes.
ii)

```
in_degree_average=((float)(in_degree_average))/(float)(len(fin_nodes))
```

Where in_degree_average stores the sum of total number of incoming edges for every node.

4. Avg. Out-Degree

```
14.573295853829936
```

Calculated using two formulas-
i)

```
 len(data)/len(fin_nodes))
```

It is the number of edges divided by the number of nodes.
ii)

```
out_degree_average=((float)(out_degree_average))/(float)(len(fin_nodes))
```

Where out_degree_average stores the sum of total number of outgoing edges for every node.

5. Node with Max In-degree

```
max in degree 457  Node: 4037
```

Calculated using two formulas-
i)

```
print("max in degree", max(np.sum(adjacency_matrix,axis=0)), "
Node:",(np.where(np.sum(adjacency_matrix,axis=0)==np.amax(np.sum(adjacency
_matrix,axis=0)))[0][0]))
```

Maximum indegree is the maximum number of incoming edges of a particular node.
iI)

Calculate the total number of incoming edges for individual nodes, and the maximum indegree is that of the node with a maximum number of edges incoming.

```
if(tot_incount> max_indegree):
    max_indegree=tot_incount
    max_indegree_node= node1
```

6. Node with Max out-degree

```
max out degree 893  Node: 2565
```

i)

```
print("max out degree", max(np.sum(adjacency_matrix,axis=1)), "
Node:",(np.where(np.sum(adjacency_matrix,axis=1)==np.amax(np.sum(adjacency
_matrix,axis=1)))[0][0]))
```

Maximum outdegree is the maximum number of outgoing edges of a particular node

ii)

Calculated the total number of outgoing edges for individual nodes, and the maximum outdegree is that of the node with a maximum number of edges outgoing.

```
if(tot_outcount> max_outdegree):
    max_outdegree=tot_outcount
    max_outdegree_node= node1
```

Maximum outdegree is the maximum number of outgoing edges of a particular node

7. The density of the network

```
print("The density is",len(data)/((len(fin_nodes)-1)*len(fin_nodes)) )
```

Density is calculated using the formula-

The total number of edges is divided by (number of vertices - 1) times (number of vertices) for directed graphs.

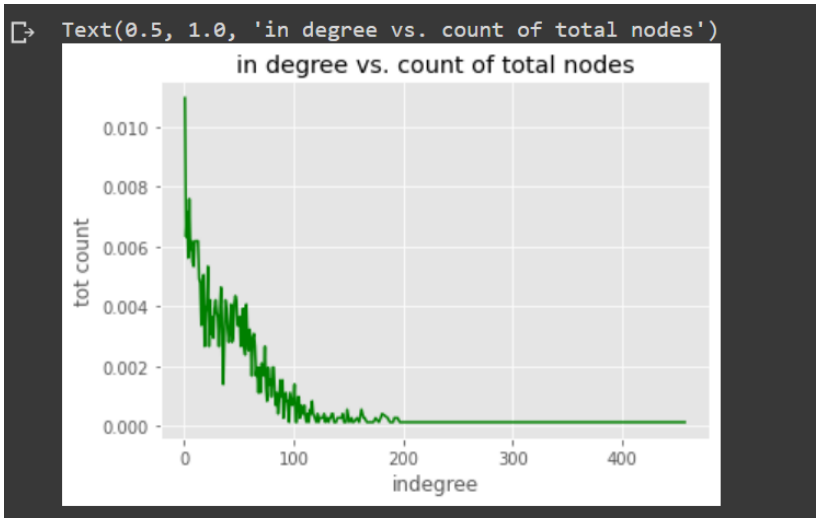
For an undirected graph, it is multiplied by 2

Further, perform the following tasks:

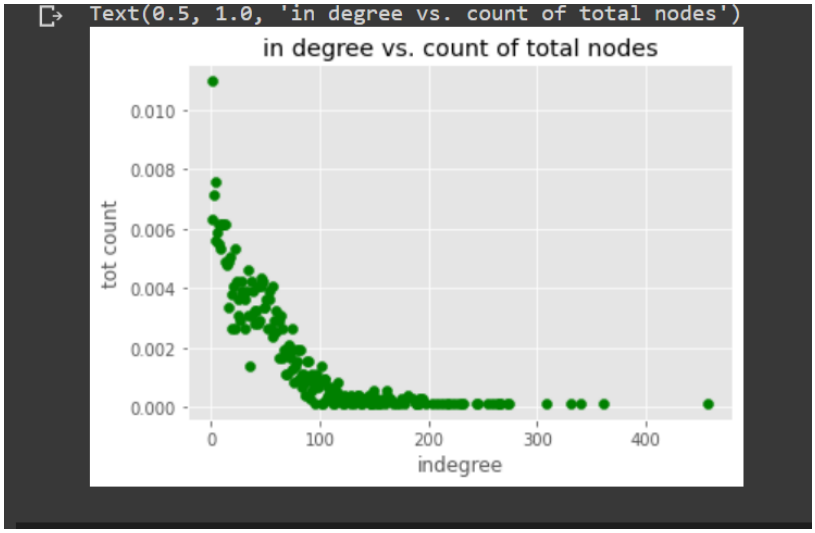
1. [5 points] Plot degree distribution of the network (in case of a directed graph, plot in-degree and

out-degree separately).

<u>In degree-</u>



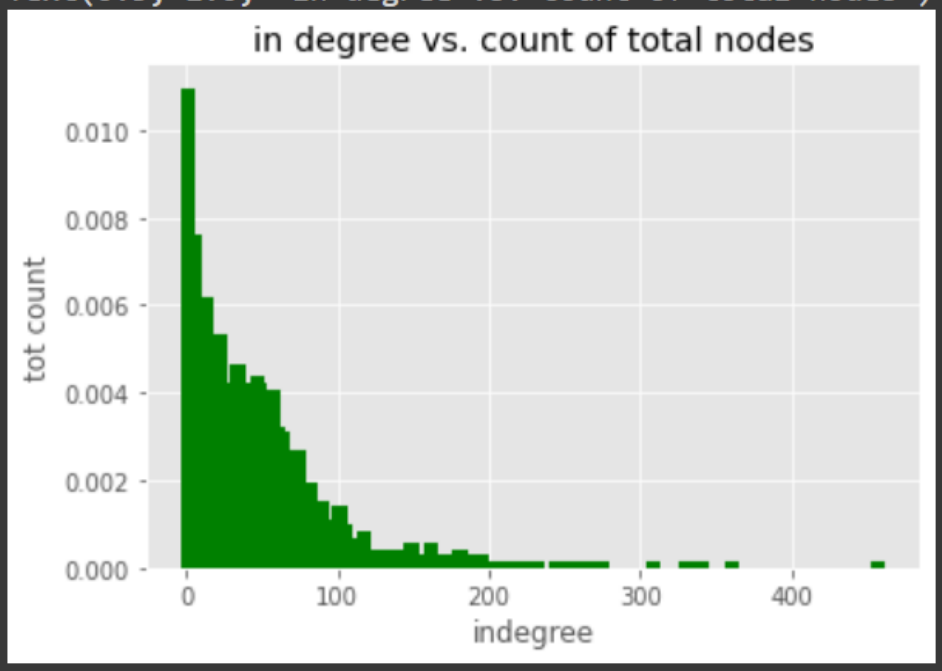
Scatter plot-



scatter plot

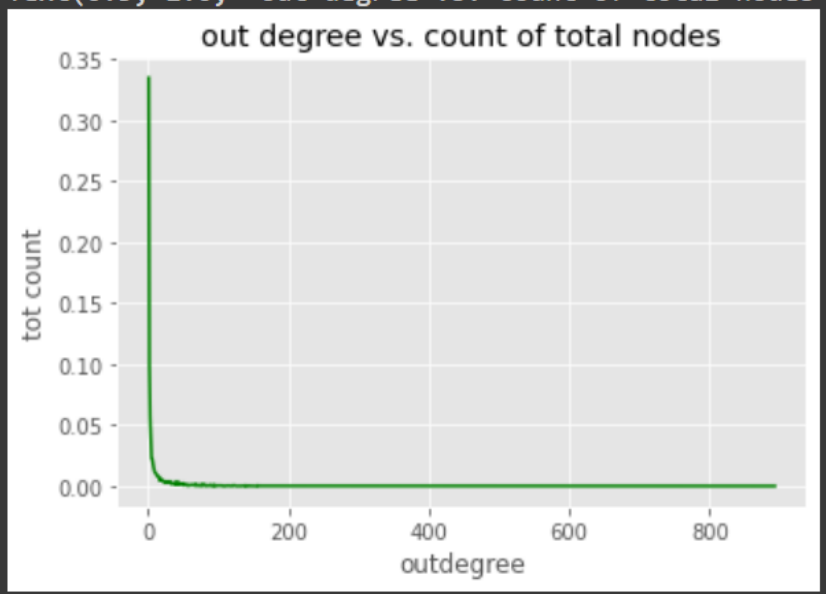
Bar plot-

```
Text(0.5, 1.0, 'in degree vs. count of total nodes')
```

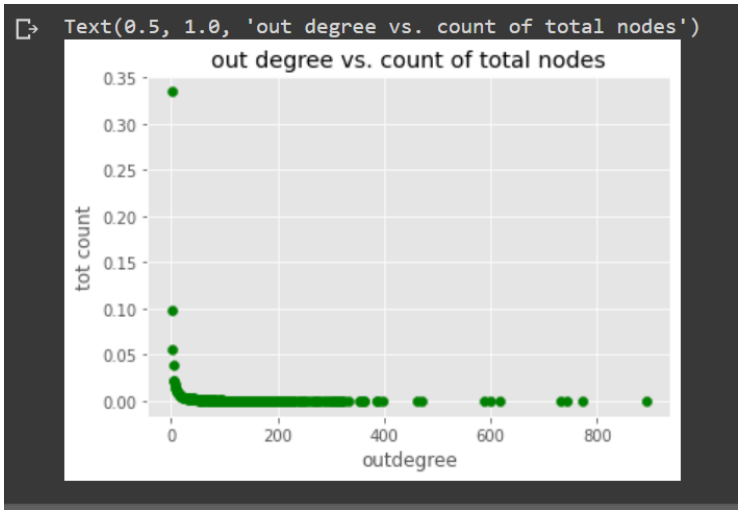


Out-degree-

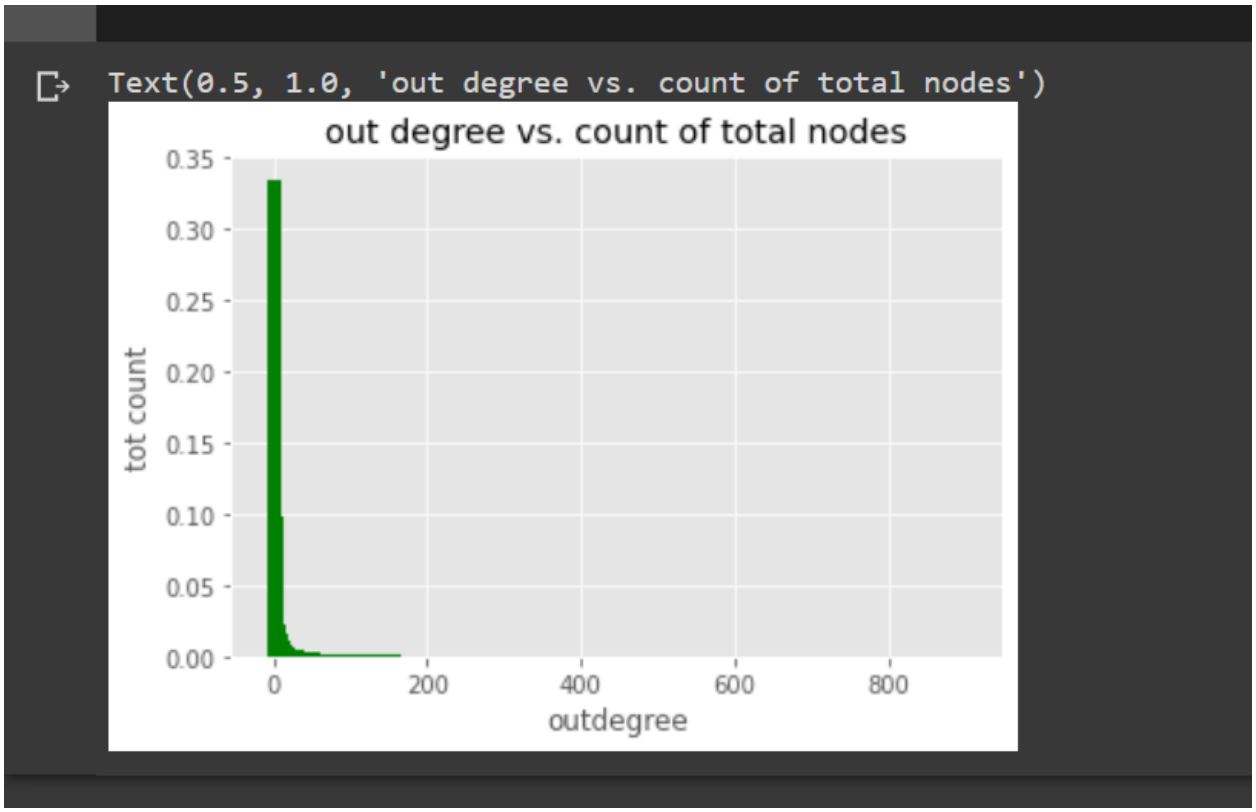
```
Text(0.5, 1.0, 'out degree vs. count of total nodes')
```



Scatter plot -



Bar plot -



2. [10 points] Calculate the local clustering coefficient of each node and plot the clustering-coefficient distribution of the network.

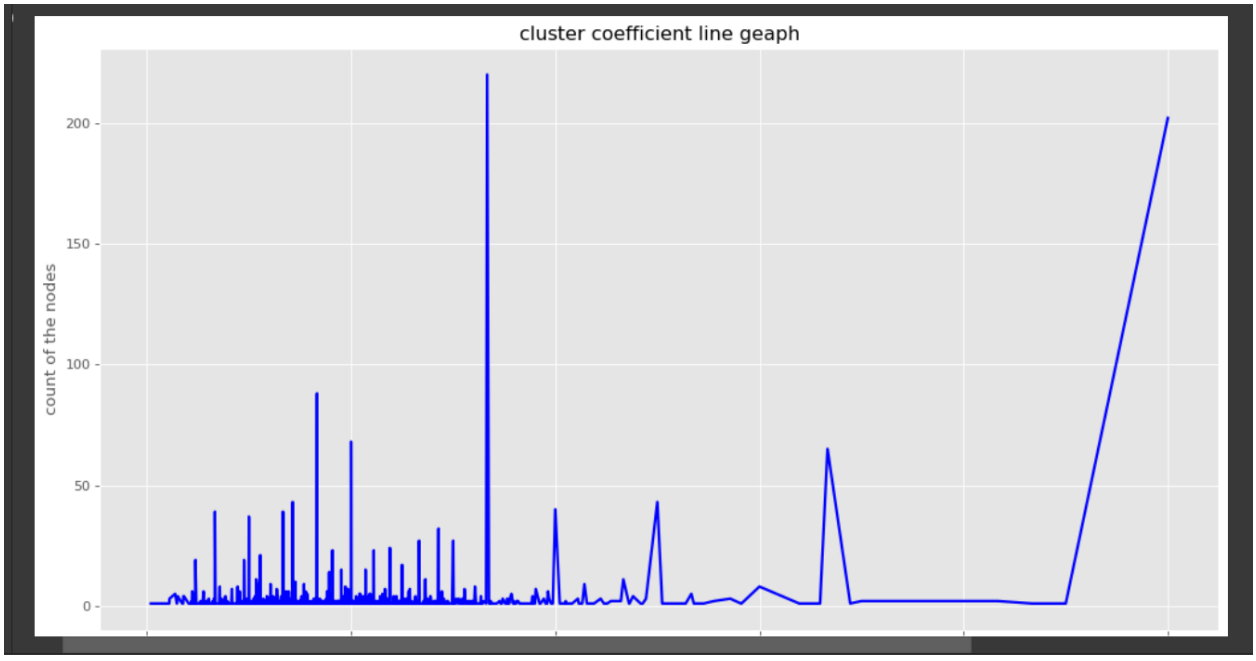
Converted the graph-adjacency matrix to undirected graph matrix for calculating the local coefficient of every node-
If a vertex v has k number of neighbors, then k(k-1)/2 edges can exist among vertices within the neighbors.
Local coefficient for undirected graphs-

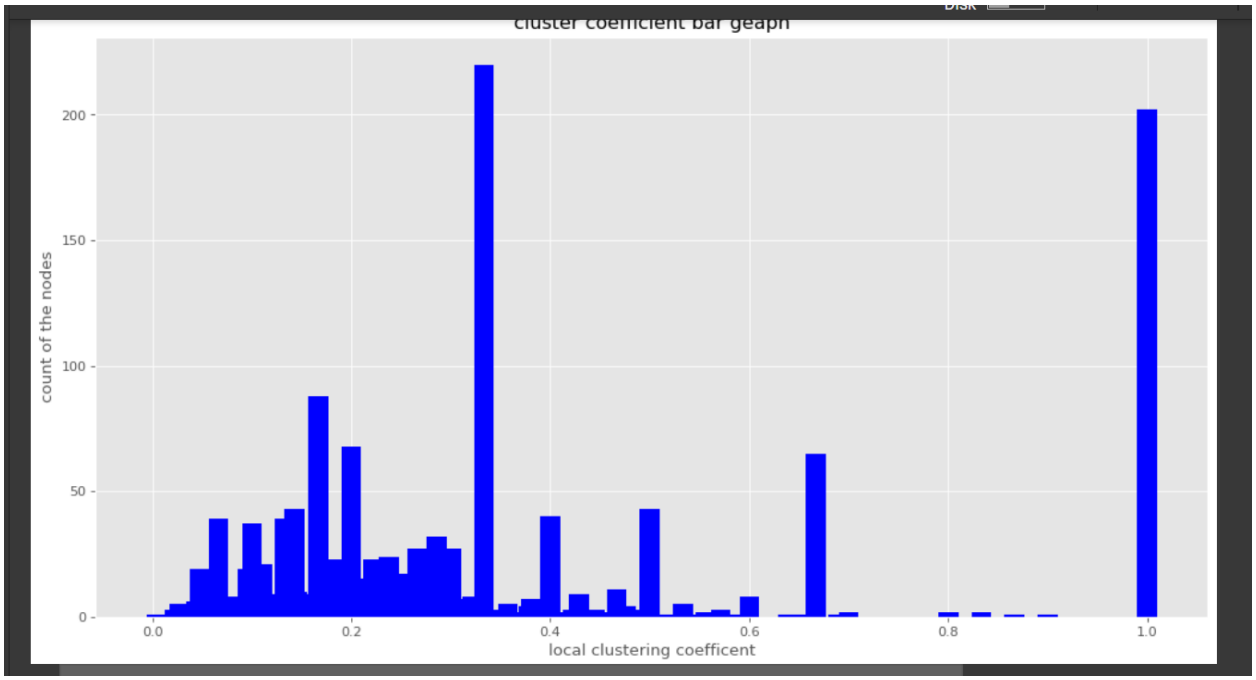
$$C_n = \frac{2T_n}{d_n(d_n - 1)}$$

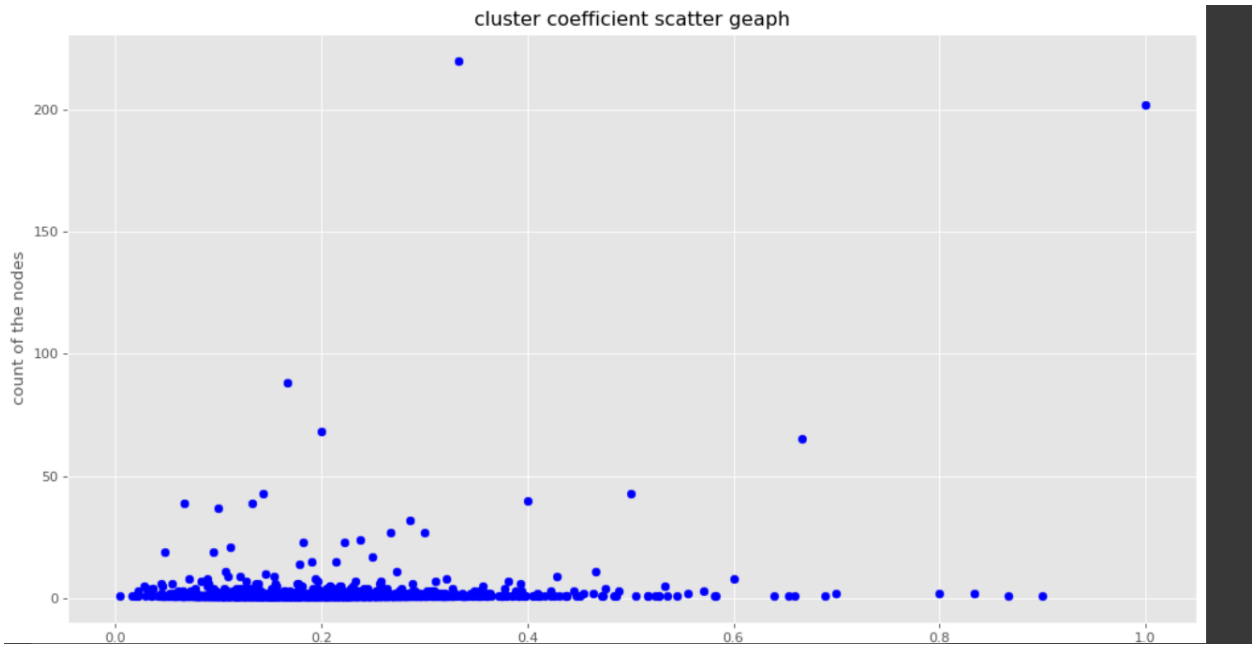
Where
Cn is the local clustering coefficient of node n.
Tn is the number of triangles a particular node is a part of .
Dn is the total degree of that particalu;lar node.

Few values-

```
{3: 0.2196078431372549, 4: 0.23399014778325122, 5: 0.34782608695652173, 6: 0.06395881239698012, 7: 0.15217391
```




cluster coefficient bar geaph
count of the nodes
local clustering coefficent
0
50
100
150
200
0.0
0.2
0.4
0.6
0.8
1.0


cluster coefficient scatter geaph
count of the nodes
0
50
100
150
200
0.0
0.2
0.4
0.6
0.8
1.0

Question 2 - [35 points] PageRank, Hubs and Authority

For the dataset chosen in the above question, calculate the following:

1. [15 points] PageRank score for each node
2. [15 points] Authority and Hub score for each node

[5 points] Compare the results obtained from both the algorithms in parts 1 and 2 based on the node scores.

HINT: Note that PageRank computes a ranking of nodes in the graph based on the structure of the incoming links. On the other hand, the HITS algorithm computes the authority score for a node based on the incoming links and computes the hub score based on outgoing links.

NOTE: You CAN use libraries like networkx to solve this question.

You are allowed to subsample the dataset in case it is not processable on your machine. Ensure that you use an approach like random walk to subsample the nodes so that you get a connected network.

1)

Calculated using

```
page_rank=nt.pagerank(digraph1,alpha=0.8)
page_rank
```

Some values are-

```
{3: 0.00020192490435020787,
 4: 5.522683582556362e-05,
 5: 5.522683582556362e-05,
 6: 0.00030768520129828417,
 7: 5.522683582556362e-05,
 8: 0.00032475006415766 46,
 9: 5.522683582556362e-05,
 10: 0.00040156839441441 09,
 11: 5.522683582556362e-05,
 12: 5.522683582556362e-05,
 13: 5.522683582556362e-05,
```

ii)

```
hubs,auth=nt.hits(digraph1)
```

```
print(hubs)
```

```
{3: 4.0210316397776376e-05, 4: 7.319607685824178e-05, 5: 3.501788474433638e-05, 6: 0.0010539872861763616,
```

```
print(auth)
```

```
{3: 9.50117185846073e-05, 4: -0.0, 5: -0.0, 6: 6.398065594290155e-05, 7: -0.0, 8: 0.000187766801949214, 9
```

[5 points] Compare the results obtained from both the algorithms in parts 1 and 2 based on the node scores.

Ans)

HIghest page rank -

```
(4037, 0.0045192264338374 81)
```

Top 20 page ranks-

```
(4037, 0.004519226433837481)
(15, 0.003542574133664394)
(6634, 0.003219620121295294)
(2625, 0.0031132124801420877)
(2470, 0.002534998738509616)
(2237, 0.0024801691288450353)
(2398, 0.002444997985780813)
(4191, 0.0021656867122448833)
(5254, 0.0020651885536163467)
(7553, 0.00205061803681289)
(1186, 0.0020393364656671 29)
(2328, 0.0019534065380085296)
(7620, 0.0018431652962861342)
(1297, 0.001841599949043567)
(4335, 0.0018140365096805504)
(4875, 0.0017943596060918613)
(7632, 0.00177292160346536)
(5412, 0.0017709255275136955)
(2654, 0.001731686613775378)
(3352, 0.0016975220947357606)
```

Highest authority value:

```
(2398, 0.0025801471780088738)
```

Top 20 authority values-

```
for i in range(0,20): # top 20 highest authority values with nodes
  print(z2[i])
```

```
(2398, 0.0025801471780088738)
(4037, 0.002573241124229792)
(3352, 0.002328415091497682)
(1549, 0.0023037314804571782)
(762, 0.0022558748562871394)
(3089, 0.002253406688451163)
(1297, 0.0022501446366627216)
(2565, 0.0022235641039536117)
(15, 0.0022015434925655806)
(2625, 0.0021978968034030723)
(2328, 0.0021723715453407385)
(2066, 0.002107040939609974)
(4191, 0.002081194130528987)
(3456, 0.0020504355215107775)
(737, 0.0020393826293356693)
(3537, 0.001957956707591016)
(2576, 0.0019547902768889494)
(4712, 0.0018716357520568287)
(5412, 0.001869411316148911)
(2535, 0.0018680659041295823)
```

Highest hubs value:

```
(2565, 0.007940492708143142)
```

Top 20 hubs values -

```
(2565, 0.007940492708143142)
(766, 0.00757433529750125)
(2688, 0.006440248991029863)
(457, 0.006416870490261073)
(1166, 0.006010567902411203)
(1549, 0.005720754058269245)
(11, 0.004921182063808106)
(1151, 0.00457204070175641)
(1374, 0.004467888792711109)
(1133, 0.003918881732057349)
(2485, 0.0037844608130803746)
(2972, 0.0035176739768147175)
(3449, 0.003503558110460446)
(3453, 0.0034494148611122102)
(4967, 0.0034433407418341284)
(3352, 0.003381423106344999)
(2871, 0.0032390167017277093)
(5524, 0.0031957811103467994)
(3642, 0.003156068703698416)
(1608, 0.0031218439181332205)
```

Printing the common nodes between i)authority,page rank
ii) authority, hubs
iii)hubs, page ranks

```
[183] print(common_member(high_auth1,high_page1))

      {3352, 2625, 5412, 4037, 15, 1297, 2328, 2398, 4191}

[184] print(common_member(high_auth1,high_hub1))

      {3352, 1549, 2565}

[185] print(common_member(high_hub1,high_page1))

      {3352}
```

Observation-

Page rank is highest for node 4037, and the node with the highest indegree, is 4037, stating that page rank is based on the incoming edges. In directed graphs, page ranks works on the link's quality with quantity based on incoming edges to an individual node.

Hubs is highest for node 2565, and the node with the highest outdegree, is 2565, stating that hubs is based on Hits algorithm whic is based on the outcoming edges. In directed graphs, hubs works on the link's quality with quantity based on outgoing edges of an individual node.

Authority is highest for node 2398, 4037 have the highest incoming links, thus works same as page rank.

Nodes with high value of outgoing links or edges, will have higher authority and page rank. Those with high value of incoming links will have higher hubs values.

Stating few differences between page rank and hits-

i) page ranks works on incoming links, while hits work on outgoing links.
ii) page rank is based on random surfer and hits works on hubs and authorities recursively.
iii) page ranks calculate highest incoming links from good incoming nodes, while hits Calculate highest outgoing links to a good node.
iv) page rank is applied to the whole graph and then search is done, while hits is appleid to subgraphs after searching .
v) Page rank- Google's visualRank , hits- spam detection