Calculation of various graph parameters and centrality features using SNAP

Assignment-1
Social Computing - CS60017

Name : N Krithick Santhosh
Roll No : 17AE30009

Date : 07/09/2019
Context : Assignment usage documentation
Indian Institute Of Technology, Kharagpur,
WB 721302, India

System specifications:

All the code given in the files attached are run on Python 2.7 in Ubuntu 18.04 environment with 8GB ram and NVidea GeForce 920MX GPU. All the code runtimes are specified only if they take more than 30 seconds to complete.

Problem 1:

Task 1.1:

All the graphs given in table 1.1 were generated and stored in the subgraphs folder. To generate the subgraphs required, run the following commands:

```
krithick@krithick-Lenovo-ideapad-320-15IKB:~/Desktop/SocialcompProj$ cd graphGeneratingScripts_Task1.1
krithick@krithick-Lenovo-ideapad-320-15IKB:~/Desktop/SocialcompProj/graphGeneratingScripts_Task1.1$ python 1.1
1.1a.py  1.1b.py  1.1c.py
krithick@krithick-Lenovo-ideapad-320-15IKB:~/Desktop/SocialcompProj/graphGeneratingScripts_Task1.1$ python 1.1a.py
```

Text: cd graphGeneratingScripts_Task1.1 && python 1.1a.py

Graph	Generating script
soc-Epinions1.txt	1.1a.py
Cit-HepPh.txt	1.1b.py
Email-Enron.txt	1.1c.py
p2p-Gnuetella.txt	Original graph

Upon running the following scripts, task 1.1 would be completed. Here's how the core part of script 1.1a.py looks.

```
G1=snap.LoadEdgeList(snap.PUNGraph,"../subgraphs/soc-Epinions1.txt",0,1)
G2=snap.TUNGraph.New()

for NI in G1.Nodes():
    if NI.GetId()%2:
        G2.AddNode(NI.GetId())

for EI in G1.Edges():
    if EI.GetSrcNId()%2 and EI.GetDstNId()%2:
        G2.AddEdge(EI.GetSrcNId(),EI.GetDstNId())
```

Task 1.2:

The entire code can be generated by running:

```
krithick@krithick-Lenovo-ideapad-320-15IKB:~/Desktop/SocialcompProj$ python gen_structure.py soc-Epinions1-subgraph.elist.txt
```

Text: python gen_structure.py soc-Epinions1-subgraph.elist.txt
Text: python gen_structure.py Cit-HepPh-subgraph.elist.txt
Text: python gen_structure.py Email-Enron-subgraph.elist.txt
Text: python gen_structure.py python gen_structure.py p2p-Gnutella04-subgraph.elist.txt

Upon running, you will have something that looks similar whats given below

```
Number of nodes in soc-Epinions1-subgraph: 27407
Number of edges in soc-Epinions1-subgraph: 102004
Number of nodes with degree=7 in soc-Epinions1-subgraph: 443
Node id (s) with highest degree in soc-Epinions1-subgraph: 645
Degree distribution of soc-Epinions1-subgraph is in: outDeg.soc-Epinions1-subgraph.png
Shortest path distribution of soc-Epinions1-subgraph is in: diam.soc-Epinions1-subgraph.png
Fraction of nodes in largest connected component in soc-Epinions1-subgraph: 0.921625862006
Number of edge bridges in soc-Epinions1-subgraph: 14555
Number of articulation points in soc-Epinions1-subgraph: 7105
Component size distribution of soc-Epinions1-subgraph is in: scc.soc-Epinions1-subgraph.png
Average clustering coefficient in soc-Epinions1-subgraph: 0.1063
Number of traids in soc-Epinions1-subgraph: 221969
Clustering coefficient of random node 37139 in soc-Epinions1-subgraph: 0.0
Number of traids of random node 47711 participates in soc-Epinions1-subgraph: 0
Number of edges that participate in at least one triad in soc-Epinions1-subgraph: 69085
Clustering coefficient distribution of soc-Epinions1-subgraph is in: ccf.soc-Epinions1-subgraph.png
```

The distribution plots shall be available as the following for soc-Epinions1 subgraph in the same working directory and should have a similar name for other subgraphs.

Parameter Distribution	Image Name
Degree Distribution	outDeg.soc-Epinions1-subgraph.png
Shortest paths	diam.soc-Epinions1-subgraph.png
Connected component	scc.soc-Epinions1-subgraph.png
Clustering Coefficient	ccf.soc-Epinions1-subgraph.png

Next page : Task 2.1

Problem 2:

Task 2.1:

NOTE: The Specified conditions of time limits are for Email-Enron subgraph because of its small size. The run time of other graphs may differ.

The entire code can be generated by running:

```
krithick@krithick-Lenovo-ideapad-320-15IKB:~/Desktop/SocialcompProj$ python gen_structure.py soc-Epinions1-subgraph.elist.txt
```

Text: python gen_centrality.py soc-Epinions1-subgraph.elist.txt
Text: python gen_centrality.py Cit-HepPh-subgraph.elist.txt
Text: python gen_centrality.py Email-Enron-subgraph.elist.txt
Text: python gen_centrality.py python gen_structure.py p2p-Gnutella04-subgraph.elist.txt

Upon running the output files shall be stored in the same directory and shall have the name in the following format <Subgraph name>_<centrality type>.txt

Upon running, you will have something that looks similar whats given below:

```
krithick@krithick-Lenovo-ideapad-320-15IKB:~/Desktop/SocialcompProj$ python gen_centrality.py Email-Enron-subgraph.elist.txt
File created for storing Email-Enron-subgraph with the name Email-Enron-subgraph_degree_centrality.txt
Execution for Closeness Centrality completed in  1.0  mins and  36.0 seconds
File created for storing Email-Enron-subgraph with the name Email-Enron-subgraph_closeness_centrality.txt
Execution for Betweeness Centrality completed in  3.0  mins and  1.0 seconds
File created for storing Email-Enron-subgraph with the name Email-Enron-subgraph_betweeness_centrality.txt
The top  10  nodes of  Closeness Centraliry  are:
1        node = 195       centrality = 0.00084011662551
2        node = 444       centrality = 0.000840064724822
3        node = 273       centrality = 0.000840053181429
4        node = 78        centrality = 0.000840040959358
5        node = 516       centrality = 0.000840030386584
6        node = 423       centrality = 0.000840029319621
7        node = 639       centrality = 0.000840022141944
8        node = 588       centrality = 0.000840017874195
9        node = 1077      centrality = 0.000840009338825
10       node = 378       centrality = 0.000840003519355
The top  10  nodes of  Betweeness Centrality  are:
1        node = 273       centrality = 1922913.0
2        node = 588       centrality = 1740845.0
3        node = 195       centrality = 1536185.0
4        node = 543       centrality = 935450.0
5        node = 516       centrality = 793453.0
6        node = 1824      centrality = 678834.0
7        node = 4746      centrality = 590984.0
8        node = 444       centrality = 554227.0
9        node = 213       centrality = 531713.0
10       node = 5022      centrality = 507969.0
krithick@krithick-Lenovo-ideapad-320-15IKB:~/Desktop/SocialcompProj$
```

The runtime for the program under Email-Enron is around 5 minutes and takes upto 15 mins for soc-Epinions1 subgraph.

The output is given so that we can compare it with the next task values

Next page : Task 2.2

Task 2.2:

NOTE: The Specified conditions of time limits are for Email-Enron subgraph because of its small size. The run time of other graphs may differ.

The entire code can be generated by running:

```
krithick@krithick-Lenovo-ideapad-320-15IKB:~/Desktop/SocialcompProj$ python analyze_centrality.py Email-Enron-subgraph.elist.txt
```

Text: python analyze_centrality.py soc-Epinions1-subgraph.elist.txt
Text: python analyze_centrality.py Cit-HepPh-subgraph.elist.txt
Text: python analyze_centrality.py Email-Enron-subgraph.elist.txt
Text: python analyze_centrality.py python gen_structure.py p2p-Gnutella04-subgraph.elist.txt

Upon running the program automatically compares the top ten and gives the output in the specified format.

Upon running, you will have something that looks similar whats given below:

```
krithick@krithick-Lenovo-ideapad-320-15IKB:~/Desktop/SocialcompProj$ python analyze_centrality.py Email-Enron-subgraph.elist.txt
Initialting Calculations of Closeness Centrality using inbuilt function
Execution for Closeness Centrality completed in  0.0  mins and  12.0 seconds
Initialting Calculations of Betweeness Centrality using inbuilt function
Execution for Betweeness Centrality completed in  0.0  mins and  30.0 seconds
The top  10  nodes of  Closeness Centraliry  are:
1        node = 195       centrality = 0.30288043335
2        node = 444       centrality = 0.293539277424
3        node = 273       centrality = 0.291539324715
4        node = 78        centrality = 0.289451219047
5        node = 516       centrality = 0.28766882354
6        node = 423       centrality = 0.287490167428
7        node = 639       centrality = 0.286294042813
8        node = 588       centrality = 0.285587540408
9        node = 1077      centrality = 0.284184945054
10       node = 378       centrality = 0.283236502696
The top  10  nodes of  Betweeness Centrality  are:
1        node = 273       centrality = 1713953.18957
2        node = 195       centrality = 1660229.38864
3        node = 588       centrality = 1423537.81512
4        node = 543       centrality = 793060.660136
5        node = 516       centrality = 761158.023022
6        node = 1824      centrality = 759165.862037
7        node = 444       centrality = 706038.568974
8        node = 213       centrality = 666525.061049
9        node = 93        centrality = 502865.22941
10       node = 78        centrality = 482811.605873
Number of overlaps for Closeness Centrality: 10
Number of overlaps for Betweeness Centrality: 8
```

The runtime for the program under Email-Enron is around 45 seconds and takes 6 minutes for soc-Epinions1 subgraph.

We note that the number of overlaps in Betweenness centrality is 8, this is because we have considered 0.8 as the node fraction while running inbuilt for faster results and the 2 matching nodes could have been eliminated in this process.