Lampros Floudas
f3352126

Social Network Analysis

Snap.py

Part 1:

At first, the Graphs are generated. Since we need an undirected graph, the snap.TUNGraph is passed in the function for Erdos-Renyi Graph. For Barabasi-Albert, the function GenPrefAttach is used. Lastly, for the Watts-Strogatz model, GenSmallWorld function is used. In all of the above, the appropriate parameters are passed. As for the results for each graph, they are presented below.

Erdos-Renyi model:

```
Node 279 has the highest Degree with a value of 0.034034
Node 841 has the lowest Degree with a value of 0.008008
The average degree of all Nodes is 0.020020020020002
The variance of the degrees is 1.8897776655534415e-05
Node 279 has the highest PageRank with a value of 0.001608
```

Barabasi-Albert model:

```
Node 0 has the highest Degree with a value of 0.109109
Node 260 has the lowest Degree with a value of 0.005005
The average degree of all Nodes is 0.009979979979979 98
The variance of the degrees is 9.886473059646231e-05
Node 0 has the highest PageRank with a value of 0.009400
```

Watts-Strogatz model:

```
Node 296 has the highest Degree with a value of 0.017017
Node 311 has the lowest Degree with a value of 0.006006
The average degree of all Nodes is 0.009983983983983983
The variance of the degrees is 2.5063341619898175e-06
Node 296 has the highest PageRank with a value of 0.001586
```

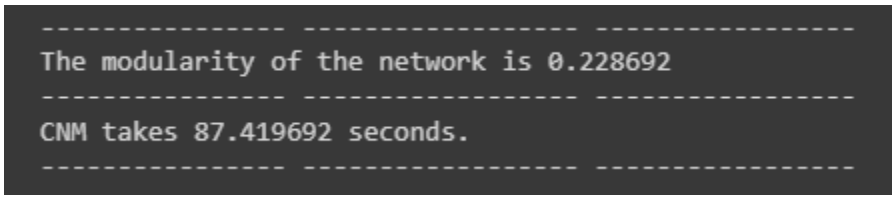
Since in case of a tie, any node would do, the loop is broken when the first node with a maximum value is found.

Part 2:

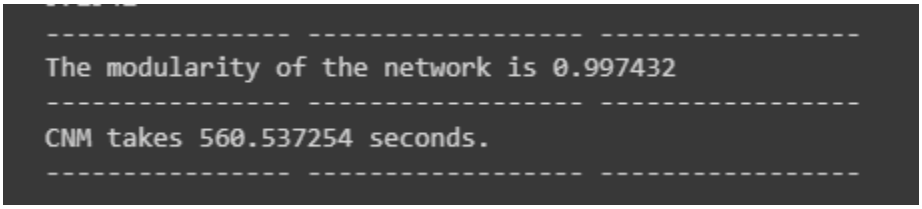
For the second part of the model, three Graphs using the Barabasi-Albert model were created. For the first, 50 nodes were passed, for the second 18000 and for the third one million since the Graph could not be created using 1 billion nodes. All graphs had the same degree of 7. Two community detection algorithms were used here. Firstly, the Clauset-Newman-Moore and secondly the Girvan-Newman. It was found that for all cases the Clauset-Newman-Moore algorithm was way faster. In fact, for the latter two cases, the Girvan-Moore did not yield any results as it could not run.

For the first question and the 50 nodes, both algorithms run in under 1 second. However, the difference between the CNM and GN methods was apparent, with the first running in 0.01 seconds while the latter running in 0.34 seconds. While each algorithm yields completely different results, one can use either of them.

Moving onto the second question, for 18000 nodes, The CNM algorithm run for 87 seconds when using 7 degrees graph. On the other hand, when using 50 degrees it took 277 seconds to run. The GN algorithm did not run at all. So in this case only CNM is viable to be used.



As for the third and final question, 1 million nodes were used as for 1 billion the Graph was not able to be made. For 5 degrees either algorithm was unable to run. The CNM run for 1 degree and it took 560 seconds to run.



*Some different results may appear in the code I sent since I was not able to use Rnd. I was getting an error GenRndGnm() got an unexpected keyword argument 'Rnd' when trying to.