Social Computing [CS60017] 2020A

Assignment 2: *Calculating Centrality Measures*

Deadline for submission: 18 October 2020, 23:59 IST.

General Instructions

1. Read the instructions given in each section carefully.
2. Your codes should print out *exactly* what is asked, and in the specified format (sample given at the end). Do **NOT** output anything extra that isn't asked for. The results will be evaluated by an automated checker. There will be a **penalty** if the specified output format is not followed.
3. Try writing codes into different functions and add several comments. Make sure your code can be easily read. Illegible codes will be **penalised**!
4. **How and what to submit:** Solutions should be uploaded via the CSE Moodle website (see course website for details). Submit one .zip or .tar.gz file containing a compressed folder that should contain all source codes, all files to be submitted (as per the task descriptions given below) and an instructions file (see next point). Name the compressed file the same as your roll number. Example: name the compressed file "19CS60R00.zip" or "19CS60R00.tar.gz" if your roll number is 19CS60R00.
5. Along with the source codes and files asked in the tasks, also submit an additional text file called "**instructions.txt**" where you should state how to run your codes as well as any additional information you want to convey, such as the version of Python or C++ compiler. The instructions.txt file should also contain your name and roll number.
6. We should be able to run your submitted code in a computer with a reasonable configuration (for instance 2GB or more RAM) by following your submitted instructions. If any part of your code takes a long time to run (e.g., more than 10 minutes) report that in the instruction file with an estimate of time required.
7. The assignment should be done individually by each student. You should not copy any code from one another, or from any web source. Plagiarised codes will be awarded zero for the whole assignment.

Dataset

For this problem use the edge list of Facebook social circles from the SNAP website:
https://snap.stanford.edu/data/ego-Facebook.html

Task 1: Coding centrality functions [80 points]

Write a code to compute the following centrality metrics for a graph:

1. **Closeness centrality** for node i, given by $C_i = \frac{n-1}{\sum_j d_{ij}}$, where d_{ij} is the length of the shortest path from i to j, and n is the number of nodes in the graph. [20 points]
2. **Betweenness centrality** for node i, given by $B_i = \frac{2}{(n-1)(n-2)} \sum_{st} \frac{n_{st}^i}{g_{st}}$, where n_{st}^i is the number of shortest paths between nodes s and t which pass through i, and g_{st} is the total number of shortest paths between nodes s and t. [30 points]
3. **Biased PageRank** for a node i calculated using the standard PageRank power-iteration method (as discussed in class). Use a non-uniform preference vector biased towards nodes that have their node IDs divisible by 4. Use damping factor, $\alpha = 0.8$. [30 points]

Please note the following carefully:

1. Your code file should be named **gen_centrality.py** (or in cpp), and should **NOT** take any input arguments. Include the dataset in your submission.
2. It should output a text file each for the centrality measures, which contains a line for each of the node. Name the output files "**closeness.txt**", "**betweenness.txt**" and "**pagerank.txt**". Generate the files inside a folder called "**centralities**".
3. Each line in the output files has the format: *nodeID <white space> centrality value*. The centrality values can be up rounded to **6** decimal places. The nodes in each file should be sorted by the centrality value. (Sample at the end)
4. You can build your code on SNAP (using graph classes etc.).
5. There are already built-in functions in SNAP to calculate all these centralities (example: "GetBetweennessCentr"). However, you should **NOT** use these functions. You need to implement these centralities yourself, as extensions to SNAP's graph framework.
6. In fact you can implement more efficient ways of computing centralities in your code. There are multiple algorithms, like variation of Floyd Warshall algorithm, Johnson's algorithm and Brandes' algorithm. You can read more about them here: https://en.wikipedia.org/wiki/Betweenness_centrality#Algorithms .

Task 2: Using built-in functions [20 points]

Calculate the **Closeness centrality**, **Betweenness centrality**, and **standard PageRank** of all nodes using the in built functions available in SNAP library. While calculating **Betweenness centrality**, set *NodeFrac* parameter to 0.8 to calculate approximate values (but faster!). Use instructions same as the Task 1 where applicable.

Your code should be in a file: **analyze_centrality.py** (or in cpp). Use the output files generated in Task 1 and do the following for each of the centralities:

1. Take the top ranked 100 nodes generated by SNAP for each of the centralities.
2. Take the top ranked 100 nodes generated by your implementation of the corresponding centrality, in Task 1.
3. Calculate and print how many nodes among the top 100 overlap.

Your code should print the following lines in stdout:

```
#overlaps for Closeness Centrality: <value>
#overlaps for Betweenness Centrality: <value>
#overlaps for PageRank Centrality: <value>
```

*[**Note:** Replace* <value> *with the corresponding number of overlaps]*

Sample centrality file:

The centrality files should have the format shown below (the numbers below are random). There should be a line for each of the nodes in the network, sorted by the centrality value.

```
5   0.756412
1   0.5
3   0.4511
4   0.333333
```

For any queries regarding the assignment, contact TA **Soham Poddar** *(email id on course website).*