Social Computing [CS60017] 2020A

Assignment 1: *Measuring Network Properties using SNAP library*

Deadline for submission: 26 September 2020, 23:59 IST.

General Instructions

1. This assignment is meant to use SNAP (http://snap.stanford.edu/) which is a library for handling very large graphs efficiently, and provides a large number of functions for analyzing graphs. It has two versions, one for C++ and another for Python. You are free to use either C++ or Python language. API Reference manual for both languages in available on their website.

2. At the beginning of the source code files, set the seed of the random number generator of SNAP to **42**. For example, in Python:

```
import snap
Rnd = snap.TRnd(42)
Rnd.Randomize()
```

3. 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 of up to 20 marks** if the specified output format is not followed.

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 Preparation [10 points]

Download the edge list of graphs (the "combined.txt.gz" files) whose download links are given in Table 1. For each of the graphs we want to *generate the sub graphs* using the rules given in the same Table, to make them smaller for us to work with. These subgraphs will be used throughout this assignment. Generate the edge list files (with ".elist" extensions) of the subgraphs and save them in a folder called **"subgraphs"**. Do **NOT** upload the original dataset files in your submission.
NOTE: *For this assignment, treat all graphs as undirected.*

Subgraph name	Rule to extract the subgraph	Link to download page
facebook.elist	Remove all nodes with IDs divisible by 5	https://snap.stanford.edu/data/ego-Facebook.html
twitter.elist	Keep only the nodes whose IDs are divisible by 4	https://snap.stanford.edu/data/ego-Twitter.html

Table 1: Rules for extracting subgraph for each of the networks and the submission filenames

Computations [40 points]

Write a program **gen_structure.py** (or **.cpp**) to generate following structural metrics given below. Your code should take the name of the subgraph (specified in Table 1) as a command line argument. All the output needs to be printed on standard output, and not to any file. All plots generated should be kept in a folder called **"plots"**. We will run your code from a terminal with a the following command in Python and cpp (and similarly for the other subgraphs):
Python: `python gen_structure.py facebook.elist`
CPP: `./gen_structure facebook.elist`
NOTE: *For all fractions, round up to 4 decimal places while printing.*

1. **Size of the network** [4 points]

   (a) Number of nodes
   Your code should print the following line to stdout:
   Number of nodes: <value>

   (b) Number of edges
   Your code should print the following line to stdout:
   Number of edges: <value>

2. **Degree of nodes in the network** [6 points]

   (a) Number of nodes which have degree = 7
   Your code should print the following line to stdout:
   Number of nodes with degree=7: <value>

(b) Node id(s) for the node with the highest degree. Note that there might be multiple nodes with highest degree
Your code should print the following line to stdout:
Node id(s) with highest degree: <comma separated id(s) of nodes>

(c) Plot of the Degree distribution
Your code should create the plotted image in the **"plots"** directory in a file named: **"deg_dist_<subgraph name>.png"**

3. **Paths in the network** [12 points]

(a) Approximate full diameter (maximum shortest path length) computed starting from 10, 100, 1000 random test nodes. Also calculate the average and variance across these 3 estimates of the diameter.
Your code should print lines in stdout:
Approximate full diameter by sampling <#test nodes> nodes: <diameter value>
and in the next line:
Approximate full diameter (mean and variance): <mean value>, <variance value>

(b) Approximate effective diameter computed starting from 10, 100, 1000 random test nodes. Also calculate the average and variance across these 3 estimates of the diameter.
Your code should print lines in stdout:
Approximate effective diameter by sampling <#test nodes> nodes: <diameter>
and in the next line:
Approximate effective diameter (mean and variance): <mean value>, <variance value>

(c) Plot of the distribution of the shortest path lengths in the network.
Your code should create the plotted image in the **"plots"** directory in a file named: **"shortest_path_<subgraph name>.png"**

4. **Components of the network** [8 points]

(a) Fraction of nodes in the largest connected component
Your code should print the following line to stdout:
Fraction of nodes in largest connected component: <value>

(b) Number of edge bridges: An edge is a bridge if, when removed, increases the number of connected components.
Your code should print the following line to stdout:
Number of edge bridges: <value>

(c) Number of articulation points: A node is a articulation point if, when removed, increases the number of connected components.
Your code should print the following line to stdout:
Number of articulation points: <value>

(d) Plot of the distribution of sizes of connected components
Your code should create the plotted image in the **"plots"** directory in a file named: **"connected_comp_<subgraph name>.png"**

5. **Connectivity and clustering in the network** [10 points]

(a) Average clustering coefficient of the network (briefly explained here https://en.wikipedia.org/wiki/Clustering_coefficient#Network_average_clustering_coefficient).
Your code should print the following line to stdout:
Average clustering coefficient: <value>

(b) Number of triads
Your code should print the following line to stdout:
Number of triads: <value>

(c) Clustering coefficient of a randomly selected node. Also report the selected node id.
Your code should print the following line to stdout:
Clustering coefficient of random node <node id>: <value>

(d) Number of triads a randomly selected node participates in. Also report the selected node id.
Your code should print the following line to stdout:
Number of triads random node <node id> participates: <value>

(e) Number of edges that participate in at least one triad
Your code should print the following line to stdout:
Number of edges that participate in at least one triad: <value>

(f) Plot of the distribution of clustering coefficient
Your code should create the plotted image in the **"plots"** directory in a file named: **"clustering_coeff_<subgraph name>.png"**

Sample Output

The output of your code should be exactly in the format shown below (the numbers below are random).

```
Number of nodes: 42
Number of edges: 42
Number of nodes with degree=7: 42
Node id(s) with highest degree: 1,3,4
Approximate full diameter by sampling 10 nodes: 32
Approximate full diameter by sampling 100 nodes: 42
Approximate full diameter by sampling 1000 nodes: 52
Approximate full diameter (mean and variance): 42.0,10.0
Approximate effective diameter by sampling 10 nodes: 30
Approximate effective diameter by sampling 100 nodes: 40
Approximate effective diameter by sampling 1000 nodes: 50
Approximate effective diameter (mean and variance): 40.0,10.0
Fraction of nodes in largest connected component: 0.7
Number of edge bridges: 42
Number of articulation points: 42
Average clustering coefficient: 0.5
Number of triads: 42
Clustering coefficient of random node 3: 0.5
Number of triads random node 3 participates: 42
Number of edges that participate in at least one triad: 42
```

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