**HW3 – Centrality measures**

**Submission Deadline:** 3.1.2021, 23:00

In this assignment, you will explore the python implementation of the various centrality measures and you will learn how to apply them over a friendship network.

**Submission guidelines:**

You are required to submit your solution as a zip file:

* Python file with your functions implemented, the file’s name should be ID.py where ID is the student’s ID.
* A PDF file with answers to open questions (these are marked in a blue font).
* The zip file’s name should be ID.zip where ID is the student’s ID. For example, the student Moshe Moshe with an ID of 1234567, should submit a zip file “1234567.zip”, containing his implemented solution “1234567.py” and a pdf file.
  + Do not zip the directory where your solution is stored, only zip the required files!
* You are required to implement a function called ‘get\_name’ that returns your full name in English.
* You are required to implement a function called ‘get\_id’ that returns your ID number.
* You have to follow **the exact API described in the HW** (exact function names, parameters and returned types). Please avoid typos.
* Your code **should not contain any part of loading data**. You may include a ‘main part’ code block (using the if \_\_name\_\_ == "\_\_main\_\_": syntax). Within this code block you can load data and test your implementation.

Objectives:

* Get familiar with a number centrality measures implemented in Python.

1. **Centrality measures**
2. Use the networkX package and implement centrality\_measures - a function that calculates basic centrality measures to a given network and a specific node:

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter name** | **Parameter type** | **Explanation** | **Default value** |
| ‘network’ | networkX object | The network to run the analysis over |  |
| ‘node’ | int | The node name (represented as integer) to retrieve the centrality measures for. Although in some networks, node names can take a string - we deal only with integer values. |  |
| ‘iterations’ | int | To be used by the page-rank and the authority score algorithms | 100 |

The function returns **a dictionary** with the following key/values:

|  |  |  |
| --- | --- | --- |
| **Key** | **Type** | **Explanation** |
| ‘dc’ | float | The degree centrality of the given node |
| ‘cs’ | float | The closeness centrality of the given node |
| ‘nbc’ | float | The normalized betweenness centrality of the given node |
| ‘pr’ | float | The normalized page-rank score of the given node |
| ‘auth’ | float | The normalized authority score of the given node |

* You can assume that ‘network’ has a single connected component.
* For calculating the page-rank score, use the default 0.85 dumping factor.

In the rest of this question, we will use a friendship network (called G1). The network can be found [here](https://moodle2.bgu.ac.il/moodle/mod/resource/view.php?id=1668051) (already saved as a networkX object). It represents social relations (i.e., friendship) at a university department.

1. Calculate the different centrality measures of nodes 1, 50 and 100 over the friendship network (G1). Calculate the measures using the function you wrote in section (i).
2. Suppose you are employed by an online shopping website and are tasked with selecting one user (taking from a social network) to send an online shopping voucher to. We expect that the user who receives the voucher will send it to their friends in the network. You want the voucher to reach as many nodes as possible. The voucher can be forwarded to multiple users at the same time, but the travel distance of the voucher is limited to **one step\***, which means if the voucher travels more than one step in this network, it is no longer valid. Implement a function called ‘single\_step\_voucher’ that selects the best candidate for the voucher. The function single parameter is:

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter name** | **Parameter type** | **Explanation** | **Default value** |
| ‘network’ | networkX object | The network to run the analysis over |  |

The function returns an integer - the best node name to send the voucher to.

\* We will use the term **‘step’** in the rest of this HW. A step means that the voucher moved from **a group of users to another group of users**. In the section above, it moves from a group of users (of size 1) to another group of users.

1. Run the ‘single\_step\_voucher’over G1 and record the results.
2. Now consider a new setting where there is no restriction on the number of times a voucher can be passed-on. Because the network is connected, regardless of who you pick, every node in the network will eventually receive the voucher. However, we now want to ensure that the voucher reaches all nodes in the lowest number of steps.

Implement a function called ‘multiple\_steps\_voucher’to calculate who is the best candidate in the network under this condition. The function parameters and output format are the same as in section (iii).

1. Run the ‘multiple\_steps\_voucher’over G1 and record the results.
2. Assume that the restriction on the voucher’s travel distance is still removed (same as in section (v)). However, the value of the voucher is diminished by 2.5% in every step the voucher makes and nullifies after 4 steps (it diminished by 2.5% from its **original starting value**) . We now want to maximize the total benefit of the network from the voucher. Re-implement the function from section (v), with the added information. The new function name is ‘multiple\_steps\_diminished\_voucher’. Input and output parameters of the original function remain the same.
3. Run the ‘multiple\_steps\_diminished\_voucher’ over G1 and record the results. Is there any difference compared with section (v)? Why?
4. Assume the restriction on the voucher’s travel distance is still removed (same as in section (v)), but now a competitor has developed a strategy to remove a person from the network in order to disrupt the distribution of your company’s voucher. Your competitor is specifically targeting people who often serve as bridges of the information flow between other pairs of people. Implement a function called ‘find\_most\_valuable’ which identifies the single node that is most valuable to your marketing strategy, thus most probable to be targeted by your competitor. The function parameters and output format are the same as in section (v).
5. Run the ‘find\_most\_valuable’ over G1 and record the results.

**BONUS QUESTION**

1. Revisit section (vii), but now, build a more generic function called ‘generic\_multiple\_steps\_diminished\_voucher’.

The function should get as input two new parameters:

1. The diminished rate ‘r’ (the one that was set to 2.5% in section (vii)) - a float value with default value of 2.5.
2. The maximum steps allowed to make with the voucher ‘max\_steps’ (the one that was set to 4 in section (viii)) - an integer with a default value of 4.

After implementing the function, check if results change once you play with the new two parameters ‘r’ and ‘max\_steps’.

**Notes:**

Please make sure you install the required packages using the latest version.

If you use any package that is not mentioned in the list below, please include a “requirements.txt” file, stating which packages you used for your solution.

List of packages:

* Pandas
* NetworkX
* Numpy
* Scipy
* tqdm
* Pickle