Exercise set #6 (17 pts)

- The deadline for handing in your solutions is November 4th 2019 23:55.
- Return your solutions (one `.pdf` file and one `.zip` file containing Python code) in MyCourses (Assignments tab). Additionally, submit your pdf file also to the Turnitin plagiarism checker in MyCourses.
- Check also the course practicalities page in MyCourses for more details on writing your report.

1. Centrality measures for undirected networks (7 pts)

In this exercise, we get familiar with some common centrality measures by applying them to undirected networks (although these measures can all be generalized also to directed networks). Below, we list and define the measures used in this exercise:

1. degree $k(i)$:
   Number of neighbors of node i
2. betweenness centrality $bc(i)$:
   Number of shortest paths between other nodes of the network that pass through node i. However, if there exist several shortest paths between a given pair of nodes, then the contribution of that node pair to the betweenness of i is given by the fraction of those paths that contain i. The betweenness scores are also normalized by $(N-1)(N-2)$, i.e. the number of all node-pairs of the network, excluding pairs that contain i (because paths starting or ending in node i do not contribute to the betweenness of i), which is the maximum possible score. Formally, if σ_{st} is the number of shortest paths from s to t and σ_{sit} the number of such paths that contain i, then
   $$bc(i) = \frac{1}{(N-1)(N-2)} \sum_{s \neq i} \sum_{t \neq i} \frac{\sigma_{sit}}{\sigma_{st}}.$$
3. closeness centrality $C(i)$:
   Inverse of the average shortest path distance to all other nodes than i:
   $$C(i) = \frac{N-1}{\sum_{v \neq i} d(i,v)}.$$
4. k-shell $k_s(i)$:
   Node i belongs to the k-shell, if it belongs to the k-core of the network but does not belong to the $k+1$-core. The k-core is the maximal subnetwork (i.e. the largest possible subset of the network's nodes, and the links between them) where all nodes have at least degree k. In other words, the 1-core is formed by removing nodes of degree 0 (isolated nodes) from the network, the 2-core is formed by removing nodes of degree 1 and iteratively removing the nodes that become degree 1 or 0 because of the removal, the 3-core is formed by removing nodes of degree less than 3 and those nodes that become of degree less than 3 because of the removals, and so on. The 1-shell is then the set of nodes that was removed from the 1-core to obtain the 2-core.

5. eigenvector centrality $e(i)$:
   Eigenvector centrality is a generalization of degree that takes into account the degrees of the node's neighbors, and recursively the degrees of the neighbors of neighbors, and so on. It is defined as the eigenvector of the adjacency matrix that corresponds to the largest eigenvalue.

a) (2 pts) Your first task is to **compute/reason without a computer the first 4 centrality measures** of the above list for the network shown in Fig. 1 (i.e. using pen-and-paper; note that you do not need to compute the eigenvector centrality, as one then would need to compute the eigenvalues of a 5×5 matrix which can be a bit painful). In your computations, use the definitions given above and show also intermediate steps where necessary.

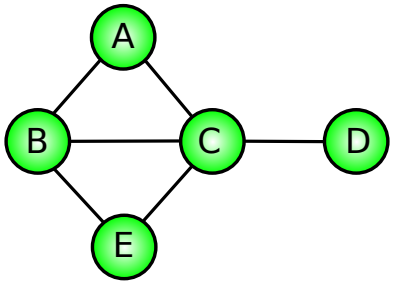

Figure 1: A small undirected network.

b) (2 pts) **Use NetworkX to compute all five centrality measures** for the networks shown in Fig 2 (`small_cayley_tree.edg, larger_lattice.edg, small_ring.edg`) as well as for the Karate club network (`karate_club_network_edge_file.edg`). Then, **visualize betweenness, closeness, k-shell, and eigenvector centrality as a function of degree in a scatter plot** for each of the networks. For easier visual comparison of the measures, you should normalize the k-shell values by dividing them by the maximal k-shell value. For this and the following tasks, you can use the Python template `centrality_measures_for_undirected_networks.py`.

Hint: For some of the networks, the power iteration algorithm used by NetworkX to calculate eigenvector centrality may not converge. In this case, increase the value of the tolerance (`tol`) parameter of `eigenvector_centrality()` until the iteration converges.

c) (1 pts) To highlight the differences between the centrality measures, **plot five visualizations** for all the networks studied in part b (the ring lattice, the 2D-lattice, the Cayley tree and the Karate club network), **each time using one of the centrality measures to define the colors** of the network nodes. To make the visualization easier, coordinates of the nodes are provided in `.pkl` files (`small_cayley_tree_coords.pkl, larger_lattice_coords.pkl, small_ring_coords.pkl, karate_club_coords.pkl`).

d) (2 pts) Based on the results of parts a and b, **how do these centralities differ** from each other? To answer the questions, you should pick some representative nodes and try to explain why different centrality measures rank these nodes differently regarding their centralities. In your answer, briefly **cover all the networks** visualized in part c.

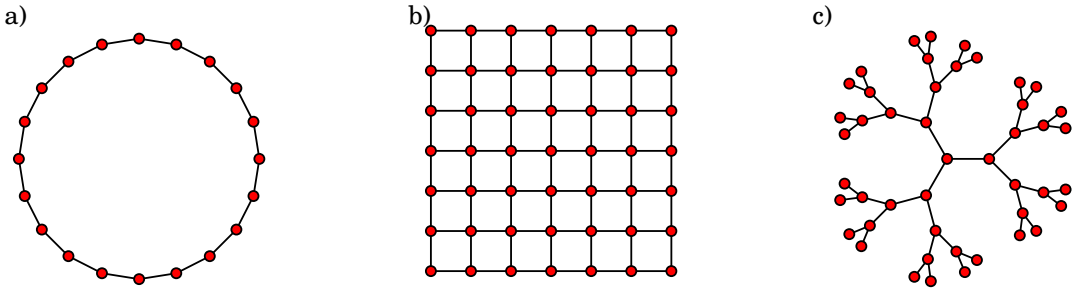

Figure 2: The model networks. **a)** A simple ring lattice with $N = 20$. **b)** A simple 2D-lattice with $k = 4$, $L = 7$, ($N = 49$). **c)** A Cayley tree with $k = 3$, and $l = 4$

2. Degree correlations and assortativity (7 pts)

In this problem, we consider degree correlations and assortativity of two real-world networks: the Zachary karate club network (`karate_club_network_edge_file.edg`) [1] and a snowball-sampled subgraph of a Facebook friendships network (`facebook-wosn-links_subgraph.edg`) [2, 3]. To get started, you can use the accompanying Python template (`degree_correlations_assortativity.py`).

For both networks, perform the following analyses:

a) (2 pts) **Create a scatter plot** of the degrees of pairs of connected nodes. That is, take each connected pair of nodes (i,j), take their degrees k_i and k_j, plot the point (k_i, k_j) on two axes with degrees as their units, and repeat for all pairs of connected nodes. Because the network is undirected, the plot should be symmetrical, containing points (k_i, k_j) and (k_j, k_i) for all connected pairs (i,j).

b) (2 pts) **Produce a heat map**[1] of the degrees of all connected nodes. The heat map uses the same information as you used in a), that is, the degrees of pairs of connected nodes. However, no points are plotted: rather, the two degree axes are *binned* and the number of degree pairs (k_i, k_j) in each bin is computed. Then, the bin is colored according to this number (*e.g.*, red = many connected pairs of nodes with degrees falling in the bin). **What extra information do you gain by using a heatmap instead of just a scatter plot (if any)?**

c) (1 pts) The assortativity coefficient is defined as the Pearson correlation coefficient of the degrees of pairs of connected nodes. **Calculate the assortativity coefficient** of the network using `scipy.stats.pearsonr` and compare your result with the output of NetworkX function `degree_assortativity_coefficient`. As mentioned in the lecture, social networks typically are assortative. **Does this hold for these two social networks? What could explain this result?**

d) (2 pts) For each node, **compute the average nearest neighbour degree** k_{nn} and **make a scatter plot** of k_{nn} as a function of k. In the same plot, **plot also the curve** of $\langle k_{nn} \rangle(k)$ as a function of k, *i.e.* the averaged k_{nn} for each k. **Comment the result** from the viewpoint of assortativity.

[1] `http://en.wikipedia.org/wiki/Heat_map`

3. Bipartite networks (3 pts, pen and paper)

Consider the bipartite network of actors and movies shown in Figure 3.

a) (1 pt) **Construct** the two unipartite projections of the network – the network of actors and the network of movies. In the former, actors are linked if they have acted together in at least one movie, and in the latter, movies are linked if there is at least one common actor.

b) (2 pts) Show that, in general, it is not possible to uniquely reconstruct a bipartite network from its two unipartite projections. **Prove this by providing a counterexample**: Take the same 4 actors and 4 movies, and design a different bipartite network that has exactly the same unipartite projections. That is, connect the actors and movies in some way that results in the same projections as in part a (draw your counterexample bipartite network in a layout similar to the the original network, so the nodes reperesening movies are located on one side and the nodes representing the actors on the other side).

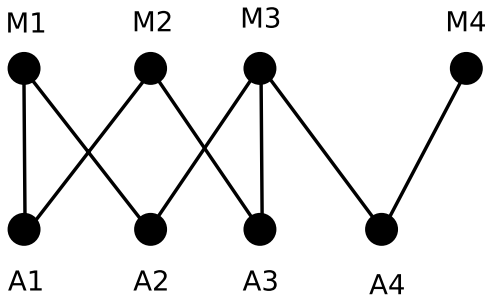

Figure 3: A bipartite network of actors (A) and movies (M).

Feedback (1 pt)

To earn one bonus point, give feedback on this exercise set and the corresponding lecture latest two day after the report's submission deadline.
Link to the feedback form: `https://forms.gle/G7HKXcsdnDEnD6599`.

References

[1] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of anthropological research*, vol. 33, no. 4, pp. 452–473, 1977.

[2] [Online]. Available: http://konect.uni-koblenz.de/networks/facebook-wosn-links

[3] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, "On the evolution of user interaction in facebook," in *Proceedings of the 2nd ACM workshop on Online social networks*. ACM, 2009, pp. 37–42.