Homework #0

This homework is easy and is meant to get you started with network analysis. Although the homework will not be graded, it will still be taken into account for course participation. All questions and comments regarding the homework should be directed to Piazza.

Submission details

This homework is due on **February 24th** at 2:00pm, while late days expire on **February 28th** at 1:00pm. The homework must be submitted as a hard-copy in the submission box in front of R 2.49 and also as an electronic version to eUcilnica. It can be prepared in either English or Slovene and either written by hand or typed on a computer. The hard-copy should include (1) this cover sheet with filled out time of the submission and signed honor code, (2) short answers to the questions, which can also demand proofs, tables, plots, diagrams and other, and (3) a printout of all the code required to complete the exercises. The electronic submission should include only (1) answers to the questions in a single file and (2) all the code in a format of the specific programming language. Note that hard-copies will be graded, while electronic submissions will be used for plagiarism detection. The homework is considered submitted only when both versions have been submitted. Failing to include this honor code in the submission will result in **10% deduction**. Failing to submit all the developed code to eUcilnica will result in **50% deduction**.

Honor code

The students are strongly encouraged to discuss the homework with other classmates and form study groups. Yet, each student must then solve the homework by herself or himself without the help of others and should be able to redo the homework at a later time. In other words, the students are encouraged to collaborate, but should not copy from one another. Referring to any solutions obtained from classmates, course books, previous years, found online or other, is considered an honor code violation. Also, stating any part of the solutions in class or on Piazza is considered an honor code violation. Finally, failing to name the correct study group members, or filling out the wrong date or time of the submission, is also considered an honor code violation. Honor code violation will not be tolerated. Any student violating the honor code will be reported to **faculty disciplinary committee** and vice dean for education.

Name & SID: _____

Study group: _____

Date & time: _____

I acknowledge and accept the honor code.

Signature: _____

1. Homework

Jernej Vivod
Introduction to Network Analysis

February 23, 2020

Question 1

Using your programming library, load the Zachary karate club network and print out the number of nodes and links in the network. Using your software tool, try to neatly visualize the network and export or print-screen the figure.

Since the Python programming language allows for fast implementation of potential solutions, I chose the SNAP (Stanford Network Analysis Platform) library (with the Python interface). The Zachary karate club has 34 nodes and 78 edges. The visualization of the network is show in figure 1.

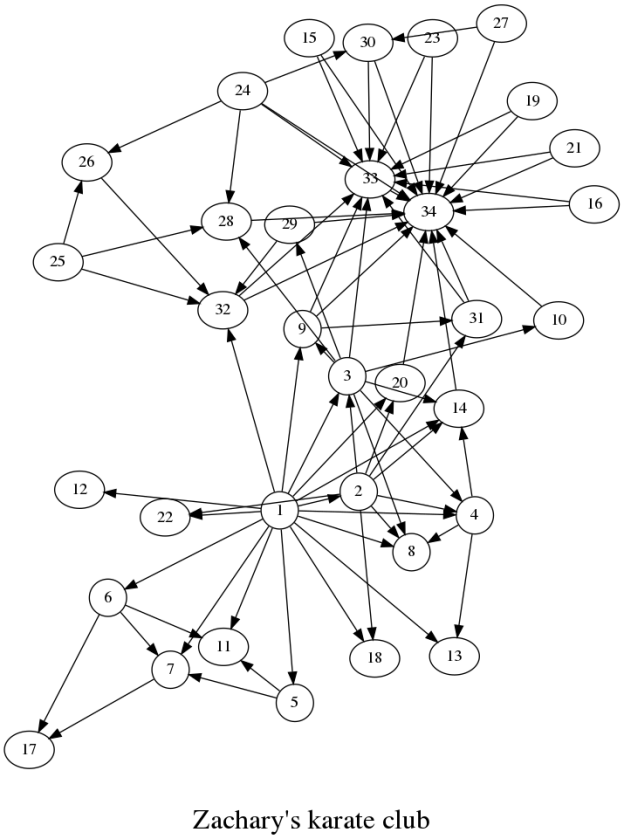

Figure 1: Visualization of the Zachary's karate club network

The code used to get the results for this question is shown below.

```
import snap

# Load network from text file.
Network = snap.LoadEdgeList(snap.PNEANet, "karate.txt", 0, 1)

# Print number of nodes and edges in network.
print("Number of nodes in the network: {}".format(Network.GetNodes()))
print("Number of edges in the network: {}".format(Network.GetEdges()))

# Save network visualization.
snap.DrawGViz(Network, snap.gvlNeato, "graph.png", "Zachary's karate club", True)
```

Question 2

Map out a network on your own. Try to be creative and collect a network that you have not seen in lectures or labs. The network can be anything, but since you will be analyzing it below, try to make it interesting. You might also use it for your course project. Compute the number of nodes and links, and the average degree in your network using either your network library or software tool

The longest used word in the German language is Rindfleischetikettierungsüberwachungsaufgabenübertragungsgesetz. We can build a network where the nodes represent letters and the edges connect the nodes for which the corresponding letters are neighbors in the word. Optionally, the edges can be weighted by the frequency of the adjacency between two letters. This network has 19 nodes and 47 edges. The average degree of a node is 4.9474.

The code used to get the results for this question is shown below.

```
import pickle

# Initialize word and decompose into list of constituent letters.
WORD = "Rindfleischetikettierungs berwachungsaufgaben bertragungsgesetz"
word_decomposed = list(WORD.lower())

# Map contained letters to their IDs.
letters = sorted(set(WORD.lower()))
letter_to_id = dict(zip(letters, range(len(letters))))

# Save inverted dictionary to file for ID decoding.
id_to_letter = {letter_id : letter for (letter, letter_id) in letter_to_id.items()}
with open('id_to_letter.pkl', 'wb') as f:
        pickle.dump(id_to_letter, f, pickle.HIGHEST_PROTOCOL)

# Initialize mapping for adjacencies.
adj_mapping = dict().fromkeys(letters)
for key in adj_mapping.keys():
    adj_mapping[key] = dict().fromkeys(letters, 0)

# Find letter adjacencies.
for idx in range(len(word_decomposed)-1):
    adj_mapping[word_decomposed[idx]][word_decomposed[idx+1]] += 1

# Construct network and save results to file.
SAVE_FILE_PATH = './longest.txt'
with open(SAVE_FILE_PATH, 'w') as f:
```

```
    for key1 in adj_mapping.keys():
        for key2 in adj_mapping[key1].keys():
            if adj_mapping[key1][key2] > 0:
                f.write("{0} {1}\n".format(letter_to_id[key1], letter_to_id[key2]))
```

```
import snap

# Load network from text file.
Network = snap.LoadEdgeList(snap.PNEANet, "longest.txt", 0, 1)

# Print number of nodes and edges in network.
print("Number of nodes in the network: {}".format(Network.GetNodes()))
print("Number of edges in the network: {}".format(Network.GetEdges()))

# Print the average degree in network.
node_iterator = Network.BegNI()
degree_aggr = 0
for idx in range(Network.GetNodes()):
   degree_aggr += node_iterator.GetDeg()
   node_iterator.Next()

# Print average node degree.
print("Average node degree in network: {}".format(degree_aggr/Network.GetNodes()))
```

Question 3

Create a random graph with the same number of nodes as in your network and set any other required parameter thus to best fit your network. Compute the number of nodes and links, and the average degree in a generated random graph.

The Erdős–Rényi model with the same number of nodes and edges also has 19 nodes and 47 edges. The average degree of a node is 4.9474.

The code used to get the results for this question is shown below.

```
import snap

# Generate a Erdos-Renyi random graph with 19 Nodes and 47 edges.
NUM_NODES = 19
NUM_EDGES = 47
Network = snap.GenRndGnm(snap.PNEANet, NUM_NODES, NUM_EDGES)

# Print number of nodes and edges in the Erdos-Renyi random graph.
print("Number of nodes in the network: {}".format(Network.GetNodes()))
print("Number of edges in the network: {}".format(Network.GetEdges()))

# Print the average degree in the random graph.
node_iterator = Network.BegNI()
degree_aggr = 0
for idx in range(NUM_NODES):
   degree_aggr += node_iterator.GetDeg()
   node_iterator.Next()

# Print average node degree.
print("Average node degree in network: {}".format(degree_aggr/NUM_NODES))

# Save constructed random graph for use with subsequent tasks.
snap.SaveEdgeList(Network, 'random.txt')
```

Question 4

Using your library compute the PageRank score of all the nodes in your network and print out the labels and scores of ten nodes with the highest PageRank score. Which nodes are the most important according to PageRank and how would you interpret these results? Is the difference between the first and the eighth node substantial or are all first eight nodes equally important?

Table 2 show the labels and PageRank scores of 8 top-rated nodes.

e	0.15779
g	0.07864
n	0.07356
s	0.06414
u	0.06148
f	0.06016
a	0.05884
b	0.05580

Table 1: First 8 top-rated nodes in the 'Longest German Word' Network.

The node corresponding to letter e is the most important according to the PageRank algorithm. This is due to it being adjacent to a more varied set of letters than the other letters. After the disproportionaly large PageRank score for the letter e, we can see the scores starting to decrease somewhat. There is a significant difference in the PageRank scores between the 8 top-rated nodes.

The code used to obtained the results is included in the answer to the next question.

Question 5

Repeat the analysis for a generated Erdős-Rényi random graph with the same parameters as before and print out the scores of eight most important nodes according to PageRank. How do the PageRank scores compare to those obtained for your network? Is the difference between the first and the eighth node substantial or are all first eight nodes somewhat equally important?

Table 2 show the labels and PageRank scores of 8 top-rated nodes. We can see that the the distance in the score between the top-rated node and worst rated node in the 8 top-rated nodes is substantial and greater than the one found in the 'Longest German Word' Network.

The code used to get the results for this and the previous questions is shown below.

```
import snap
import pickle

# Load networks from text files.
Network_random = snap.LoadEdgeList(snap.PNEANet, "random.txt", 0, 1)
Network_longest = snap.LoadEdgeList(snap.PNEANet, "longest.txt", 0, 1)
```

```
7
8  # Load dictionary mapping node IDs to labels for 'longest' network.
9  def load_dict(name):
10     with open(name, 'rb') as f:
11         return pickle.load(f)
12 id_to_letter = load_dict('id_to_letter.pkl')
13
14 # Compute PageRank scores for both networks.
15 prank_res_random = snap.TIntFltH()
16 prank_res_longest = snap.TIntFltH()
17 snap.GetPageRank(Network_random, prank_res_random)
18 snap.GetPageRank(Network_longest, prank_res_longest)
19
20 # Create list of tuples containing node IDs and their scores.
21 res_random = sorted([(node_id, prank_res_random[node_id]) for node_id in prank_res_random],
       key=lambda x: x[1], reverse=True)
22 res_longest = sorted([(node_id, prank_res_longest[node_id]) for node_id in prank_res_longest
       ], key=lambda x: x[1], reverse=True)
23
24 # Compare PageRank scores for 8 top rated nodes for Erdos-Renyi random graph.
25 for el in res_random[:8]:
26     print(el[1])
27
28 # Compare PageRank scores for 8 top rated nodes in graph of adjacencies in longest German
       word.
29 print("Label | Score")
30 for el in res_longest[:8]:
31     print("{0}  {1}".format(id_to_letter[el[0]], el[1]))
```

0.09832
0.09238
0.07987
0.07833
0.07711
0.07426
0.06147
0.05697

Table 2: First 8 top-rated nodes in the generated Erdos-Renyi random graph.