Kovacs, Nicholas

Problem 1

The adjacency matrix will look like this:

adj_list = [['a','b'], ['a','d'], ['b','a'], ['b','c'], ['b','d'], ['c','b'], ['d','a'], ['d','b']]

Show that it takes O(n(n+m)) time to compute the diameter of the network.

The pseudocode would look something like this:

```
for node in graph:
        for all_other_nodes in graph:
                if edge_between_node1_node2 exists in adj_list:
                        traverse_path.append()
print(max(traverse_path))
```

The first 2 loops equal n^2 since they visit every node twice. Visiting 1 more edge only adds 1 more round to the complexity since we are only concerned with actual edges, not all possible edges; therefore, the complexity only increases by the number of edges. Complexity is there for O(n(n^2 + m)).

Show that it takes O(E[k]) time, on the average, to list the neighbors of a node, where E[k] is the expected value of the node degree.

The degree of the adj_list is 2, since there are 8 edges and 4 nodes. For each node, we only expect to visit 2 other nodes, so it's only going to take O(E[k]) time to list neighbors of a node because there is only on average <k> edges for node.

Show that it takes time O(E[k2]) to list the second-hop neighbors (the neighbors of the neighbors).

To visit neighbors of a neighbor, we would do exactly as we do above, but we would do it twice, therefore, we expect complexity of O([k^2])

The reciprocity of a directed network is defined as the fraction of edges that run in both directions (i.e., I connects to j and j connects to i). For a directed network in which in- and out-degrees are uncorrelated, show that it takes time O(m2/n) to calculate the reciprocity of the network. Why is the restriction to uncorrelated degrees necessary?

Considering the adj_list from above, we look at all nodes that have and edge going to them, and we expect this to be equal to the number of edges divided by the number of nodes; m/n. We then have to look at edges going out of nodes, this is again of complexity m/n. Therefore, the complexity is O((m^2)/n)