Assignment 1 SNCAS

Van Nguyen s3726266

October 2023

1 Part I

Question 1.1

Combined degree is calculated with both in and out degree of a node.
Using notion of reverse neighborhood:

$N'(v) = \{u \in V : (u, v) \in E\}$

$deg(v) = deg^-(v) + deg^+(v)$

Where $deg^-(v)$ is in-degree and $deg^+(v)$ is out-degree.
In-degree being set of nodes N' (v) where (u,v) $\in E$ and out-degree (v,u) $\in E$.

Question 1.2

Given:

$$N_k(W) = N(N_{k-1}(W)) \cup N_{k-1}(W)$$

This formula is redundant since every node is connected to every other node. Therefore there is no need for iteration of k¿1 since every node is reachable in k=1. We can say that for each node in neighborhood will contain every other node. N(v) = V. To be precise we need to substract the node itself where we start.

$$N(v) = V - \{v\}$$

Where {v} is node we start at.

Question 1.3

Since k-neighborhood represent set of nodes with k-reach of a node (let say node {v}) and reverse neighborhood is set of nodes connected to node {v}. Combining them would give us a set of nodes that are k-reachable and connected to node {v}.
It measures how does the node {v} influence other nodes within k-reach.

Question 1.4
Reciprocity can be defined using reverse neighborhood as:

$$r(G, v) = \frac{\{u \in N'(v) : (v, u) \in E\}}{N'(v)}$$

{u $\in E$ N'(v) : (v, u) $\in E$ E} is set of all nodes (u) that are from reverse neighborhood (direct incoming connection/degree).

Question 1.5
KosarajuSCC(graph):
1. Initialize nodes and set status to "not visited"

2. Create an empty stack
3. Perform a DFS traversal put nodes into the stack
4. Reset status of all visited nodes to "not visited"
5. Create a reversed graph (reverse edges)
6. Initialize a list to store SCCs
7. While the stack is not empty:
8. Take a node from the stack
9. If the node is not visited:
10. DFS traversal from this node in the reversed graph
11. Add all visited nodes to the current SCC
12. Append the current SCC to the list of SCCs
13. Return the list of SCCs
Inspired by source [1].

Question 1.6
1. Initialize count to 0
2. For each node v in the network:
a. Calculate the degree of node v (degree_v)
b. Initialize local_count to 0
c. For each neighbor u of node v:
i. Calculate the degree of node u (degree_u)
ii. If degree_u >degree_v, increment local_count by 1
d. Add local_count to count
3. Calculate paradox_false as count - number_of_nodes

Time complexity: Iteration for all nodes in network = 0(V).
Iteration for each node to find out neighborhood = O(V).
Pairs in neighborhood can in be worst case = $O(V^2)$.
Therefore total time complexity can be $O(V^4)$ in worst case. Although apparently in practise it's more like $O(V^3)$ and can be reduced to $O(V^x)$ using Strassen's matrix [2] multiplication. Where x is 2.8074.

Question 1.7
1. For each node v in network:
a. Count degree of node
2. For each neighbor u of node v:
a. Count degree of node
b. if degree{u}>degree{v} then count += 1
3. Repeat for each node in network and each neighbor for said node.
4. paradox_false = count - number_of_nodes

paradox_false variable will hold number of counts where Friendship paradox does not hold true.

Time complexity:
Iteration for all nodes in network = 0(V).
Iteration for each neighbor is equivalent to amount of edges therefore 0(E)
Step 3. is just O(1)
Total complexity therefore is $O(E * V)$. Depending on amount of edges in network.

Question 1.8
1. Deleting an edge {v,w} $\in E$
a. both nodes will loose one edge therefore $(n' - 1, m' - 1)$
b. neighbors of either v or w will simply loose connection to the nodes therefore $(n', m' - 1)$
2. Adding an edge {v,w} $\in$ / E
a. Both nodes will gain one edge therefore $(n' + 1, m' + 1)$
b. neighbors of either {v} or {w} will simply gain connection to the nodes therefore $(n', m' + 1)$.
Unless the node is connected to affected node said node will not feel a change of state.

Question 1.9 Radius R(G) = minu ∈ V e(u) = minimal eccentricity
Basically calculating the longest direct route with the fewest edges from one node to another.

1. Calculate eccentricity of each node in V. (either BFS or Dijkstra)
2. The biggest value of shortest path (maximum shortest distance) is the value of radius (minimal eccentricity).

2 Part II.

Question 2.1
number of directed edges in mediun.tsv: 16329

number of directed edges in large.tsv: 149755

First I had to open the file and then I had to clean the data. First by striping white space characters including newline. Then I split the cleaned line into two parts separated by (tab). After this I get source and target representing source and target nodes of a degree. Then using .add_edge() I add source and target into to the graph. Using native function from networkx I calculate number of edges using function .number_of_edges.
Question 2.2
Users in medium social network: 5895

Users in large social network: 41767

Same principle when cleaning data. Instead of adding source and target to graph we simply add sources and target to number_of_nodes and then using function number_of_nodes.len we get number of users in social network.
Question 2.3

Using native function of networkx we can calculate indegree and outdegree by using functions called .in_degree and .out_degree. After that we are able to plot these values in histogram. Issue is that the frequency and amount of degrees is disproportional to the actual values so we had to choose bins=np.logspace (30 for medium.tsv and 40 for large.tsv) so the graphs can be showed more visibly and clearly. Figure 1 is for medium.tsv and Figure 2 is for large.tsv data set.

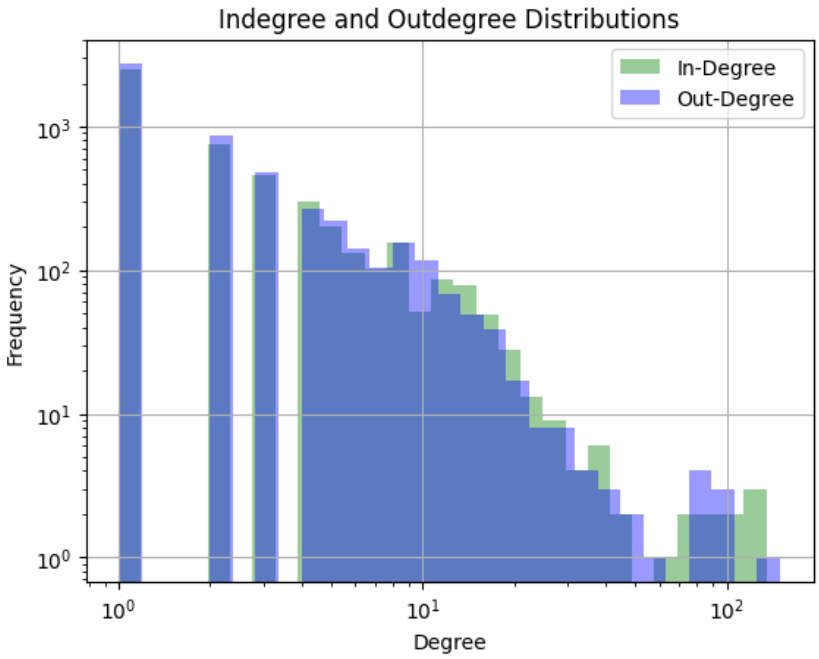

Figure 1: Indegree and Outdegree Distribution of medium.tsv

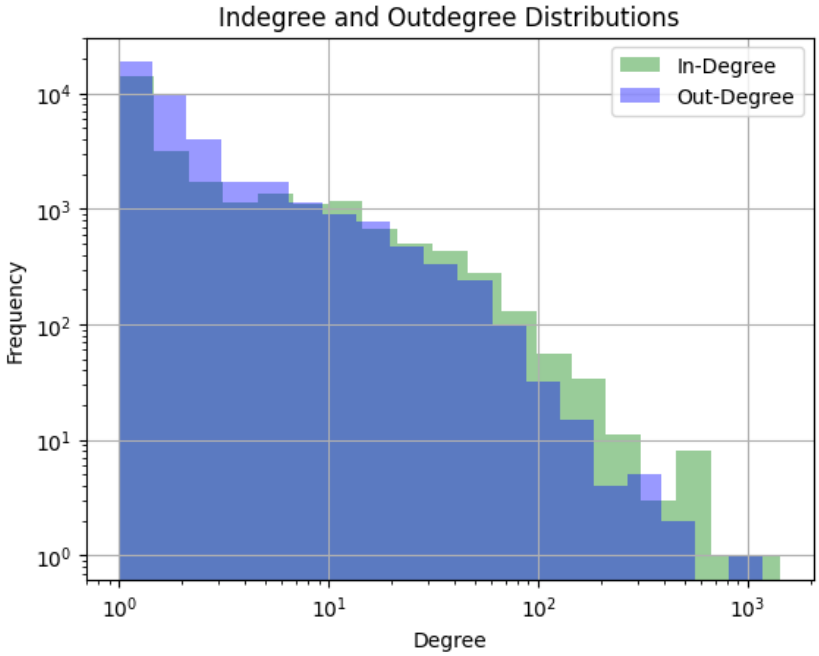

Figure 2: Indegree and Outdegree Distribution of large.tsv

Question 2.4

Strongly connected components and weakly connected components are calculated by same called functions in networkx. Number of WCC (weakly connected components) is calculated using same called function as well same foes for number of SCC. Largest WCC and SCC is calculated using max(WCC or SCC, key=len). Nodes was calculated using len(largest_SCC) and edge was calculated using subgraph.

medium.tsv:

Number of weakly connected components: 200

Number of strongly connected components: 1804

Number of nodes in the largest weakly connected component: 3677

Number of links in the largest weakly connected component: 13166

Number of nodes in the largest strongly connected component: 3677

Number of links in the largest strongly connected component: 13166

For large.tsv:

Number of weakly connected components: 647

Number of strongly connected components: 19250

Number of nodes in the largest weakly connected component: 21226

Number of links in the largest weakly connected component: 120614

Number of nodes in the largest strongly connected component: 21226

Number of links in the largest strongly connected component: 120614

Question 2.5

For average clustering coefficient I used native function called .average_clustering. This function does not take in consideration directionality.

average clustering coefficient for medium network is: 0.16390475180145161

average clustering coefficient for large network is: 0.253891847505671

Question 2.6

For this question I chose approximation method of calculating distance distribution. First calculation of largest WCC and its subgraph. Then I defined function that samples distances for node pairs. Takes graph as input and generates list of sampled shortest path distances between pair of nodes. Repeating it for given amount of times. Output will be a graph of approximate distance distribution.

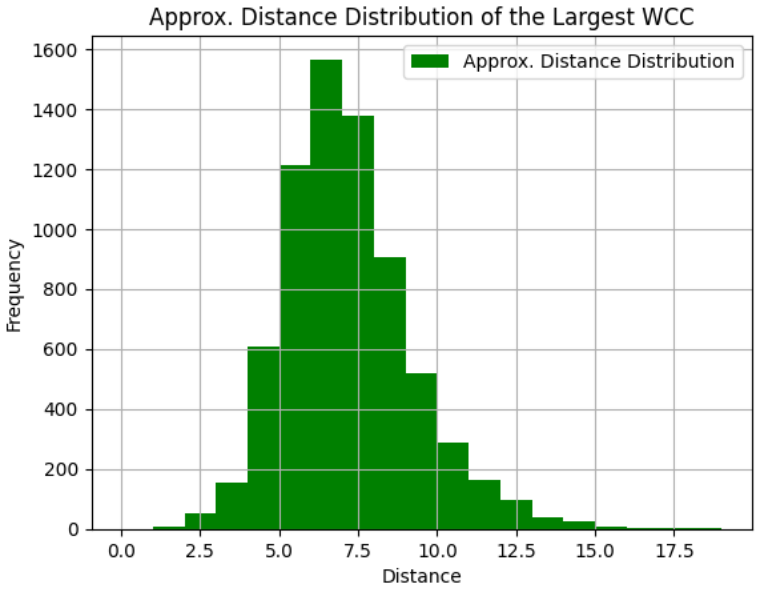

Figure 3: medium.tsv distance distribution

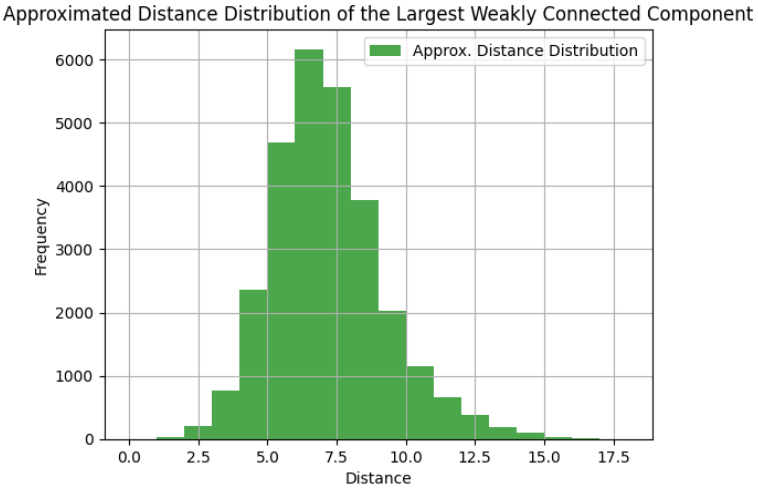

Figure 4: large.tsv distance distribution

Question 2.7 For this we just take the function from Question 2.6 and then we calculate average by dividing the sum of distances by its length (sum(distance)/len(distance).

Approximated Average Distance: 6.69201030927835

Approximated Average Distance: 6.42097040985113

Question 2.8

Figure 5 created in Gephi. Using Yifan's Hu algorithm with optimal distance set to 80. Nodes ranked by its in-degree value of sizes from 1 to 75. Opacity set to 70 and degrees set to be straight. I wanted to visualize amount of in-degree connections by stressing node sizes to see the most connected nodes.

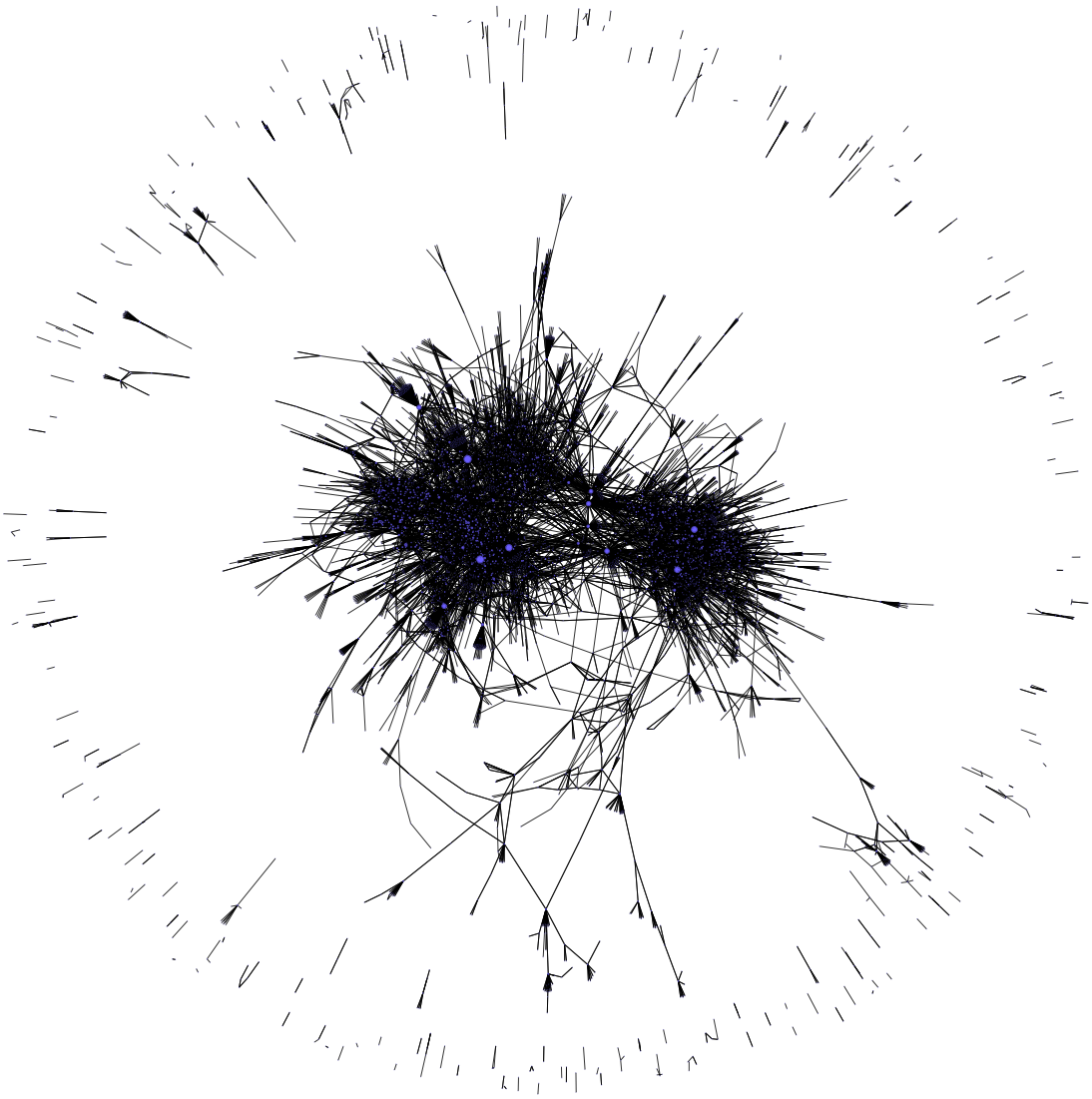

Figure 5: Gephi network visualization of medium.tsv

Sources:

[1] https://www.geeksforgeeks.org/number-of-triangles-in-a-undirected-graph/

[2] https://www.geeksforgeeks.org/connectivity-in-a-directed-graph/

[3]Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart, "Exploring network structure, dynamics, and function using NetworkX", in Proceedings of the 7th Python in Science Conference (SciPy2008), Gäel Varoquaux, Travis Vaught, and Jarrod Millman (Eds), (Pasadena, CA USA), pp. 11–15, Aug 2008