Code explanation

To begin with, after reading data from the local cloud in Google Colaboratory, we had three functions defined to simplify the computation process. The first of three serves the purpose of computing centrality measures using predefined functions in NetworkX; the other two are for plotting and visualizing for better analysis; the first provides a heat map for each centrality measure, and the second one plots a histogram for each centrality measure.

Code Listing 1: Functions for future computations

```
from networkx.classes.function import is_directed
def centrality_measures(g):
    d_cent = nx.degree_centrality(g) # this is normalized by default
    c_cent = nx.closeness_centrality(g) # this is normalized by default
    b_cent = nx.betweenness_centrality(g)
    e_cent = nx.eigenvector_centrality(g)
    h_cent = nx.harmonic_centrality(g)
    pr = nx.pagerank(g) # alpha is 0.85 by default, we can change it.
    clus_coeff = nx.clustering(g)
    # these are all dictionaries from nodes to their centrality value (or
        ↪ other variables).
    if nx.is_directed(g):
      in_cent = nx.in_degree_centrality(g)
      out_cent = nx.out_degree_centrality(g)
    else:
      in_cent = 0
      out_cent = 0

    return d_cent, c_cent, b_cent, e_cent, h_cent, pr, clus_coeff, in_cent
        ↪ , out_cent

def draw(G, pos, measures, measure_name):

    nodes = nx.draw_networkx_nodes(G, pos, node_size=2, cmap=plt.cm.plasma
        ↪ ,
                                   node_color=list(measures.values()),
                                   nodelist=measures.keys())
    nodes.set_norm(mcolors.SymLogNorm(linthresh=0.01, linscale=1, base=10)
        ↪ )
    edges = nx.draw_networkx_edges(G, pos)

    plt.title(measure_name)
    plt.colorbar(nodes)
    plt.axis('off')
    plt.savefig(measure_name+'.png', format='png')
    plt.show()


def plot_hist(measure, measure_name):
    plt.hist(list(measure.values()))
    plt.title(measure_name)
    # plt.axis('off')
    plt.savefig(measure_name+'hist.png', format='png')
    plt.show()
```

Dataset 1: General Relativity and Quantum Cosmology collaboration network

The first dataset was General Relativity and Quantum Cosmology collaboration network dataset or as it is referred to in the code: GrQc.

After trying to compute the radius, center, diameter, and eccentricity and getting errors giving infinity values, we discovered that our network was disconnected.

Degree Centrality Then, the defined functions were used to compute centralities. The first centrality is Degree Centrality. As shown in figure 1 histogram, most degree centralities are close to zero, and only a few are nearly 0.01. Since we are using a

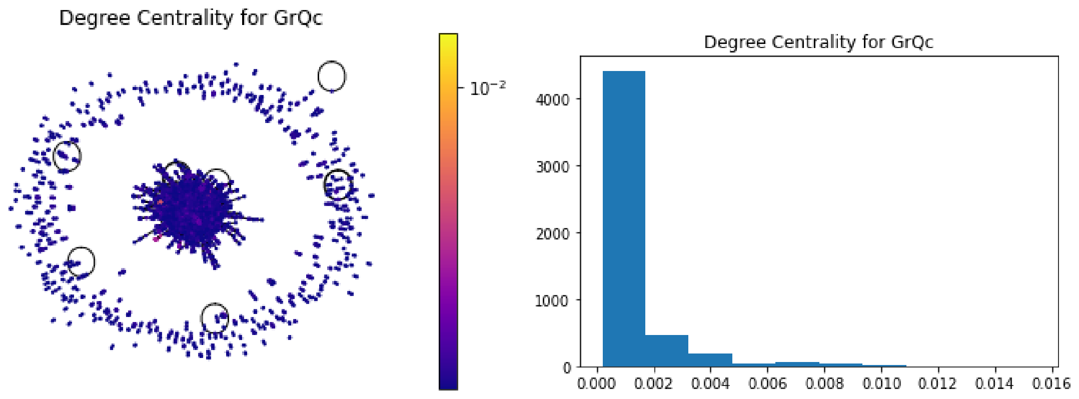

Figure 1: Heatmap for degree centrality and an approximate visualization of the network (on the left). Histogram of degree centrality of the nodes (on the right).

normalized measure, the highest possible centrality, meaning if we had a node connected to every other node, is 1. So we can conclude that our network is far from a complete graph, and we can say it's sparse.

Closeness Centrality The next measured centrality, closeness centrality, scores each node based on its sum of shortest paths. As seen in the figure 2, the distribution of closeness centrality is much more different from degree centrality. The histogram consists of some nodes whose closeness centrality is near zero and some with normal-like distribution with a mean near 0.14.
By looking at the heatmap, we can see what this means, some nodes in the middle are almost similarly close to each other, and there is a ring around these nodes with low closeness centrality meaning they're much further from each other than the ones in the middle.

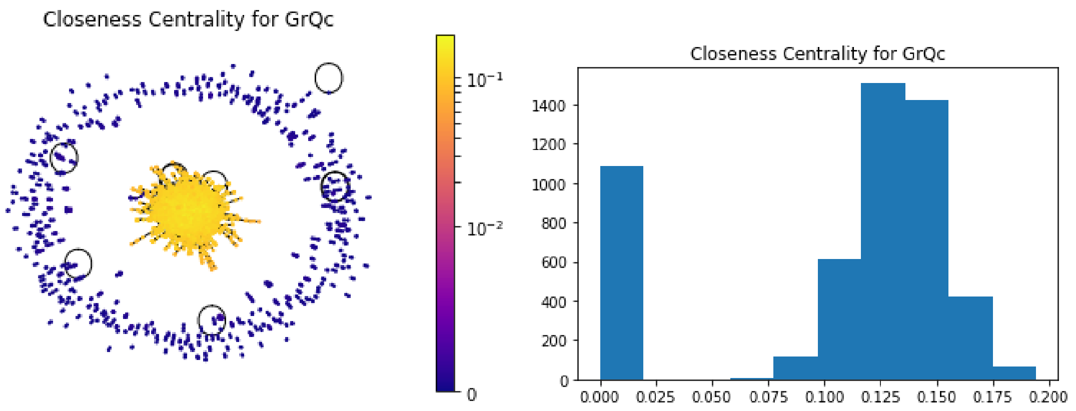

Figure 2: Heatmap for closeness centrality and an approximate network visualization (on the left). Histogram of closeness centrality of the nodes (on the right).

Betweenness Centrality counts the number of times a node lies on the shortest paths between other nodes. We can see in the figure 3 that the same thing happened to degree centrality is happening here as well. Most nodes have near zero betweenness centrality, and few are higher than 0.015. Combining the results from the last two measures, we

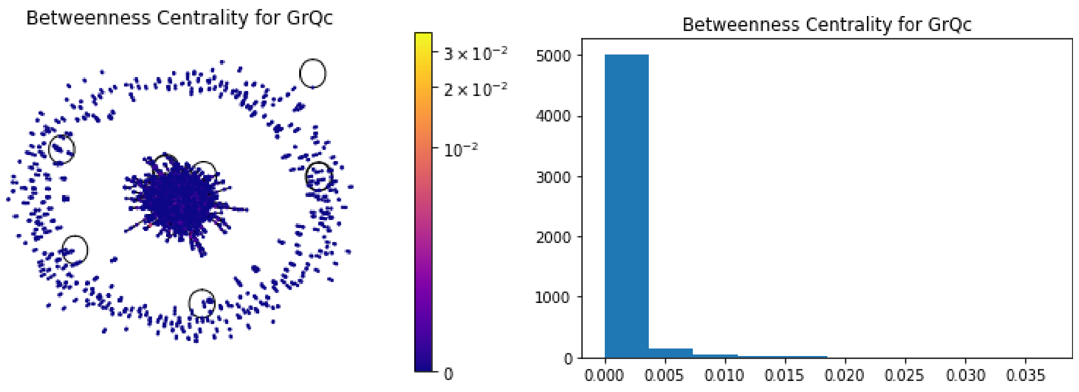

Figure 3: Heatmap for betweenness centrality and an approximate network visualization (on the left). Histogram of betweenness centrality of the nodes (on the right).

have some nodes that are relatively close to each other, but those same nodes are not on many shortest paths and are not important in this way.

Eigenvector Centrality can identify nodes with influence over the whole network, not just those directly connected to it. With this explanation, we could expect the output of the code, and seeing figure 4 confirms our expectations.

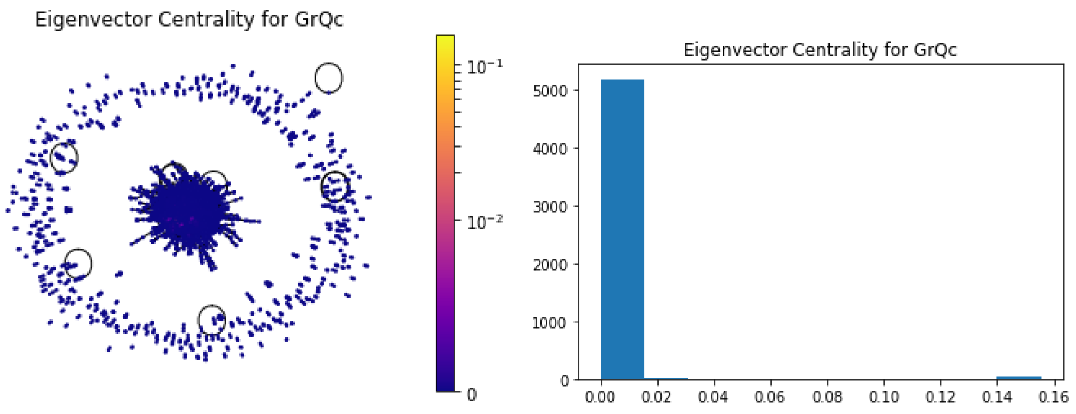

Figure 4: Heatmap for eigenvector centrality and an approximate network visualization (on the left). Histogram of eigenvector centrality of the nodes (on the right).

PageRank uncovers nodes whose influence extends beyond their direct connections into the wider network. This measure is useful in directed networks, and since our network is not directed, we may not be able to see this measure's full potential here. On

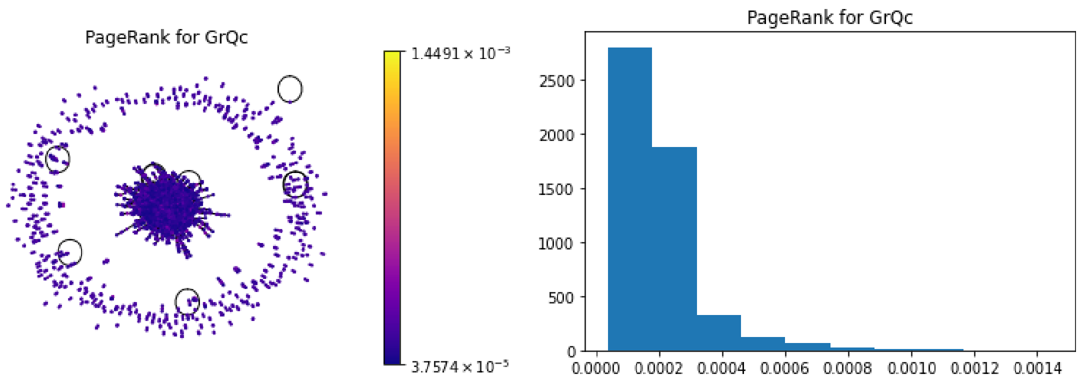

Figure 5: Heatmap for PageRank and an approximate network visualization (on the left). Histogram of PageRank of the nodes (on the right).

the histogram of PageRank, non-similar to betweenness and eigenvector centralities, its curve is not as steep as it was on former measures.

Clustering Coefficient And lastly, a clustering coefficient measures the degree to which nodes in a graph tend to cluster together. Looking at the heatmap in figure 6, we can see that nodes with clustering coefficient in the middle part are more than nodes with high cluster coefficient in the ring. Also, since there are not many connections on the ring, we can say that there are high clustering coefficients because some packs of fully connected nodes, such as two or three nodes connected, make their clustering coefficient

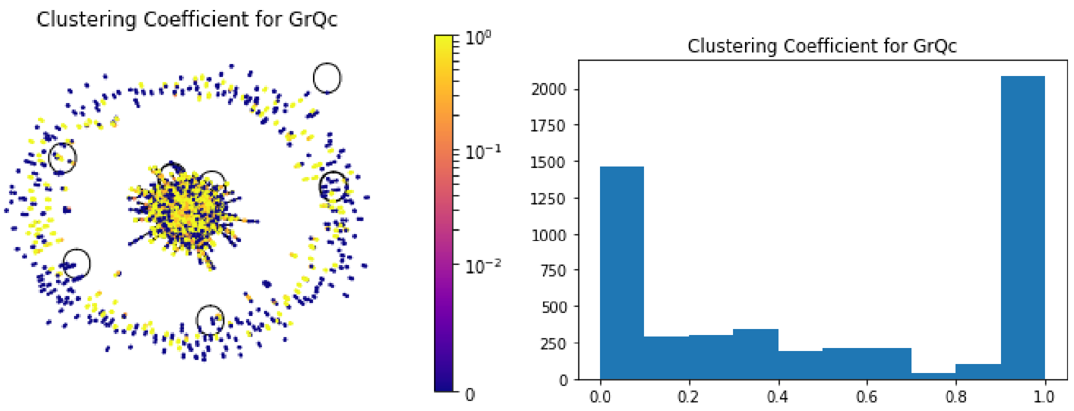

Figure 6: Heatmap for clustering coefficient and an approximate network visualization (on the left). Histogram of clustering coefficient of the nodes (on the right).

equal to one.

Further measurements There were some other measures computed, which are visible in the code, that are worth mentioning. We had 48,260 triangles, and with 5242 nodes, the total possible triangles were 23,993,367,880, meaning very few triangles were formed in the network. Moreover, there were bridges in the network (obviously), and the largest connected component was visualized in the middle of every figure up until now.
We had 4158 nodes in the largest connected component, which is close to %80 of the nodes, and we had 13428 edges in the largest connected component, which is around %92 of the total edges of the network. This means the nodes outside the largest connected component are barely connected, as we mentioned before.

NOTE From the next section on, we will only talk about more general measurements of networks in hand; however, we will still include visualizations in a folder named "Figures," and we will briefly talk about them.

Dataset 2: Wikipedia vote network

This section's network is a directed network where a directed edge from node i to node j represents that user i voted on user j.

Since visualization of the graph for this part takes a long time, I decided to skip that step and only use the metric and measurements plus histograms without knowing how the network looks. This situation can happen for many real-life networks since our datasets are getting bigger and bigger, and we will have to understand the structure and behavior of networks with minimum computation cost.

Analysis using metrics We know we have a directed graph and by trying to compute normal diameter we know our network is disconnected since we get the error "Found infinite path length because the digraph is not strongly connected." We used a new way of computing diameter which ignored infinity values.

$$diameter: \quad 10 \tag{1}$$

Moreover, we computed both largest weakly and strongly components, and computed the fraction of nodes and edges in thes components:

$$\begin{aligned} nodes\ in\ the\ largest\ weakly\ connected\ component: &\quad 0.9931131412508785 \\ edges\ in\ the\ largest\ weakly\ connected\ component: &\quad 0.9997492501615408 \\ nodes\ in\ the\ largest\ strongly\ connected\ component: &\quad 0.18271257905832747 \\ edges\ in\ the\ largest\ strongly\ connected\ component: &\quad 0.3805225240864508 \end{aligned} \tag{2}$$

These numbers tell us that almost every node is in the largest weakly connected component which consists almost every edge in our network. Therefore there are few nodes that are connected to each other outside this component that makes this graph disconnected. On the other hand, we have a relatively small largest strongly connected component which means most of the nodes in our weakly sonnected component are on the edge of our network.

Analysis using histograms We can also get some insights from seeing the distribution of centralities and histograms. Looking at histograms, we see some similarities with out first dataset. In the histograms we can see similar shapes in closeness and harmonic centralities, as expected, and also we see very low betweenness centrality for all nodes which means all of our nodes are similarly not on many shortest paths. There are some high clustering coefficients which are for mentioned disconnected nodes, and lastly, we have low PageRank for all the nodes which means we don't have nodes with many incoming links and we can confirm this by looking at in-degree centrality histogram.

Dataset 3: Gnutella peer-to-peer network, August 8 2002

For this part we got both histograms and shape of our network plus heatmap, same as first dataset, but we mostly talk about general metrics and briefly mention what we can get from histograms and heatmap, such as similarities and differences.

Analysis using metrics Same as last section, we have a directed network, and we computed same measures as last section.

$$\begin{aligned} nodes\ in\ the\ largest\ weakly\ connected\ component: &\quad 0.999682590065069 \\ edges\ in\ the\ largest\ weakly\ connected\ component: &\quad 0.9999518698560909 \\ nodes\ in\ the\ largest\ strongly\ connected\ component: &\quad 0.3282018727186161 \\ edges\ in\ the\ largest\ strongly\ connected\ component: &\quad 0.4482360302257304 \end{aligned} \tag{3}$$

Looking at these fractions and comparing them to WikiVote dataset, we can say this network has both larger strongly and weakly connected components. They both contain higher percentage of nodes and edges. But still, very few nodes are making our network disconnected, and the difference in strongly and weakly connected components tell us that many nodes are still on the edge of graph but graph is more connected in comparison with WikiVote network.

Some other measures were calculated for strongly connected component which are worth mentioning:

$$\begin{aligned} radius\ of\ largest\ strongly\ connected\ component\ is: &\quad 12 \\ diameter\ of\ largest\ strongly\ connected\ component\ is: &\quad 19 \end{aligned} \tag{4}$$

These results don't agree with the information given on the SNAP website which might be because of how the diameter and radius are calculated within directed graphs.

Analysis using histograms The most interesting histogram for this section is the closeness centrality histogram. For this network, most of the nodes have non-zero closeness centrality which means the average shortest path from a node to every other node follows an almost normal distribution.

Dataset 4: Social circles: Facebook

Analysis using metrics and figures For this part, we have a connected, undirected graph with 1,612,010 triangles. Since we have 4039 nodes in the network, most possible number of triangles are 10,973,563,139 which means we have a reasonable amount of triangles. Also, we can see that graph has bridges and we have extracted them in a dictionary since these bridges can be important. It's worth mentioning that we had bridges in previous networks too, but since they were representing nodes which were connected only to each other, they weren't as important, or in the same way important, as bridges for this part are.

We also computed diameter and radius of networks:

$$\begin{aligned} radius &: \; 4 \\ diameter &: \; 8 \end{aligned} \tag{5}$$

Which means the largest shortest path in our graph is 8 and we can get to any node at most in 8 steps. Moreover, using figure 7 we can see that our networks has communities. Here we don't see much difference in the histograms but we do have communities and clusters in the network. The reason this is not showing up in the clustering coefficient histogram is that these communities have almost similar clustering coefficients which the will shape a similar histogram to other datasets. Also, we tried to find and visualise

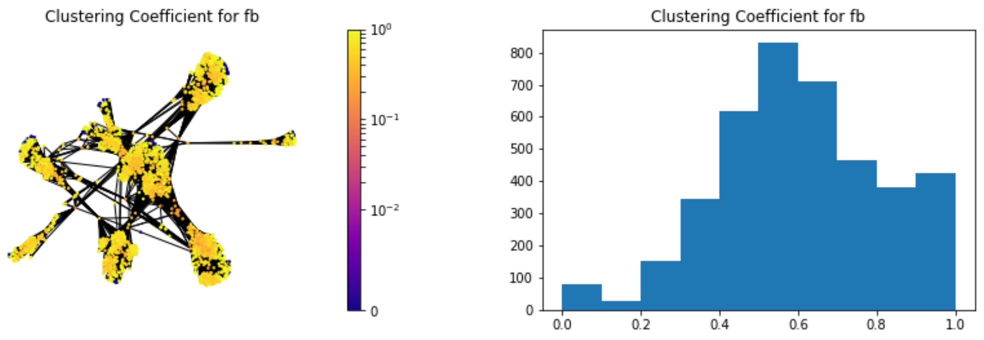

Figure 7: Heatmap for clustering coefficient and an approximate network visualization (on the left). Histogram of clustering coefficient of the nodes (on the right).

communities in the graph using Girvan-Newman algorithm but the computational cost of this algorithm is massive! Therefore, we decided to skip this step.

References

[1] Easley, David; Kleinberg Jon. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Cambridge University Press, 2010. Print.

[2] “Stanford Network Analysis Project”. SNAP, Stanford. Online. `http://snap.stanford.edu/index.html`

[3] “Network Analysis in Python”. NetworkX. Online. `https://networkx.org/documentation/stable/reference/introduction.html`