1. Dataset Description
   The dataset chosen for this project is a comprehensive business graph database compiled by the startup Relato during 2015-2016. It includes 373,663 links between companies, characterized by various relationship types such as "partnership," "customer," "competitor," "investment," and "supplier."

2. Research Problem and Questions
   The primary focus of this project will be to analyze the degree distribution of the vertices in the business graph. The specific questions I aim to address are:
   1. What is the overall degree distribution of the company vertices in the graph?
   2. How does the degree distribution differ among various types of relationships (e.g., partnerships vs. competitor links)?
   3. Does the degree distribution follow a power-law, indicating scale-free properties typically observed in social networks?

3. How the code works:
   3.1. links.csv contains many records of company business connections. It includes the names of the companies, their domains, the business relationships they have, and some other identifying information. There are about 370k different company connections and about 50k companies. Unfortunately, some of the connections do not have reciprocal connections. Here's what I mean:
      1. Company A might list Company B as a partner, but Company B might not list Company A as a partner. This is a problem.
      2. Each relationship has a reciprocal. Some are symmetric (Partner and Partner, Competitor and Competitor, None and None) and others are asymmetric (Customer and Supplier, Investor and Investee)

   3.2. The code works by reading a record of links.csv and creating a CompanyConnection instance. The instance contains some Company instance (A) as the tail node, another instance of Company (B) as the head node, a value from EdgeTypes, and some other identifying info (such as an ID number).

   3.3. Then the code calls a function called reciprocal_connection to make the reciprocal of the original CompanyConnection instance. In the reciprocal instances, B becomes the tail node and A becomes the head node. The reciprocal value of EdgeTypes is found, and "-RECI" is appended to the ID number.

   3.4. There is a HashMap which takes a Company instance as the key, and contains a vector of CompanyConnection instances as the value. The program then appends both of the CompanyConnection instances to the vector containing CompanyConnection instances and passes the tail node of CompanyConnection as the key.

   3.5. After completing this for each record in links.csv, the graph is created.

   3.6. After the graph is created, the program iterates through the graph and writes each key/value pair to a file called "graph_links.txt"

3.7. Then the program works on counting the number of degrees a node (Company) has at a particular distance. For this assignment, I calculated the degree at distances 1 and 2.

3.7.1. The program uses a modified BFS (Breadth-First Search) to count the number of degrees. In this context, 'degrees' refers to the levels of direct and indirect connections each company has within a network, based on specified relationship types (such as Customer, Supplier, etc.). The BFS starts from a specific company, considered the root of the search, and explores all its direct connections first (i.e., all companies directly connected to the root company).

3.7.2. Each connection is evaluated based on its type, and only those that match specified edge types (if any are provided) are considered in the count. The BFS utilizes a queue to manage the order of exploration, ensuring that all direct connections are explored before moving on to connections one degree further out. This process continues, with each successive layer of connections representing an additional degree of separation from the root.

3.7.3. To avoid counting the same connection multiple times and to prevent infinite loops in the presence of cyclic connections, the algorithm maintains a set of visited companies. Each company is added to this set when it is first encountered, and subsequent encounters are ignored. This method ensures that each company is considered only once. The BFS increments a counter each time a valid connection is found that hasn't been visited yet, and this counter represents the total number of connections (or degrees) up to the specified limit.

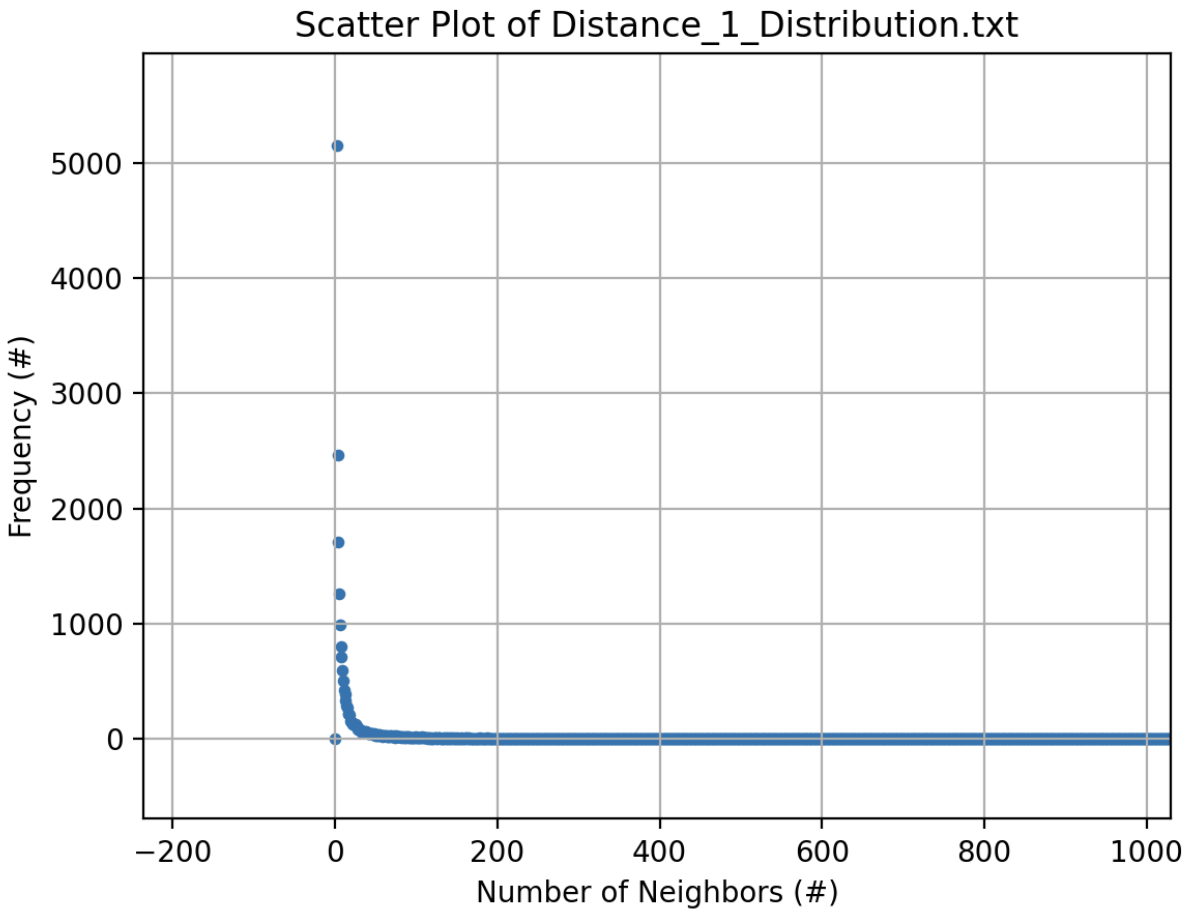
3.8. After counting the total number of degrees at a particular distance, the data is temporarily inputted into a .txt file. The program then uses a command line argument to execute a Python script that provides some useful information on the data.

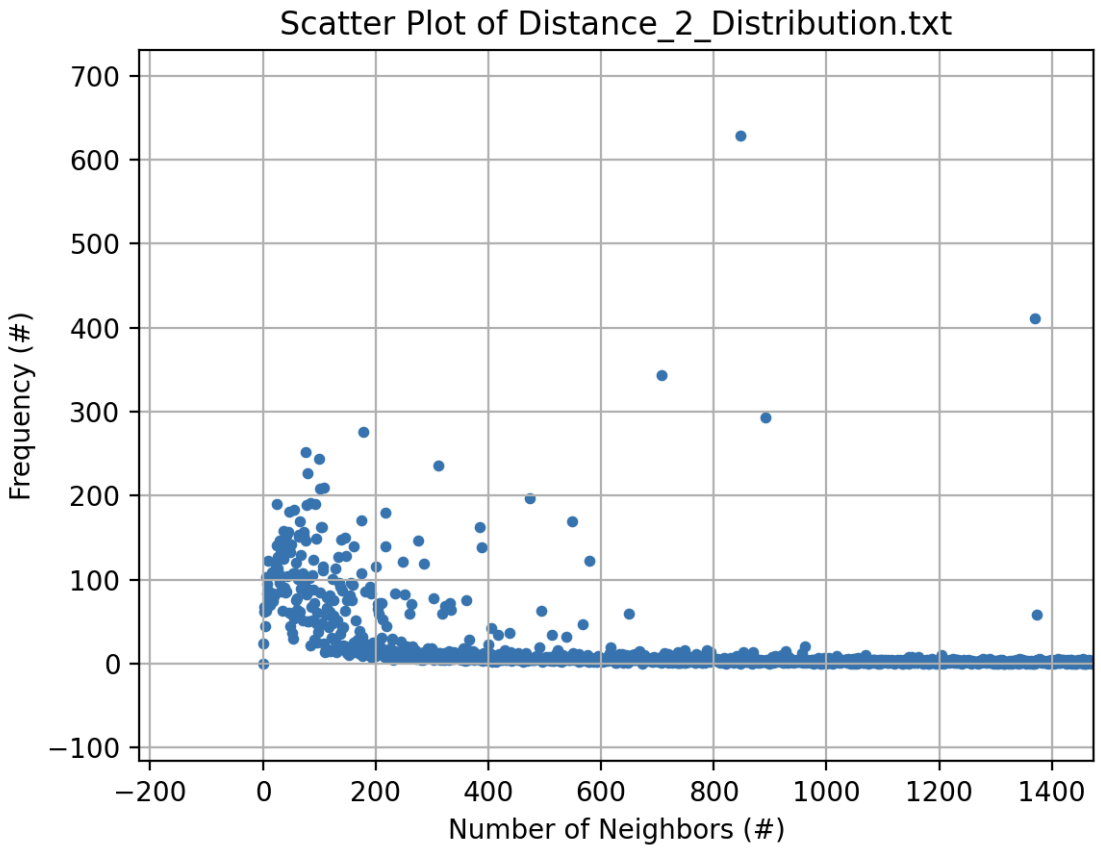
3.8.1. The script reads the .txt files and creates a NumPy array. Then the .txt files are deleted. The array is fitted to a powerlaw distribution and the Kolmogrov-Smirnov (KS) Test is calculated. From my understanding, the KS Test is a kind of goodness of fit test that basically tells you how well a data set fits a distribution. The lower the KS distance, the better the data fits to the distribution.

3.8.2. After the KS distances are calculated, some other useful metrics are calculated using NumPy. These include mean, median, standard deviation, minimum and maximum values.

3.8.3. The Python uses MatPlotLib to graph the data.

4. This is what the output looks like:




Scatter Plot of Distance_2_Distribution.txt
Frequency (#)
Number of Neighbors (#)
700
600
500
400
300
200
100
0
−100
−200
0
200
400
600
800
1000
1200
1400

```
Analysis for 1st Degree Distribution:
KS Distance: 0.0317841623562292 7
Mean: 4.219627956824619 6
Median: 0.0
Standard Deviation: 305.8853200965264
y-max: (1, 34375), x-max: (13062, 1)
y-min: (0, 0), x-min: (0, 0)


Analysis for 2nd Degree Distribution:
KS Distance: 0.041808748598560 65
Mean: 1.2377284771186061
Median: 0.0
Standard Deviation: 49.7709944419437
y-max: (13062, 9843), x-max: (44533, 1)
y-min: (0, 0), x-min: (0, 0)
```

It's immediately obvious that the degree distribution at distance = 1 (the first graph) more closely resembles a power law distribution when compared to the degree distribution at distance = 2. It's difficult to see from these images, but on the MatPlotLib window, it's even more obvious when you zoom into the data points.

Indeed, this is reflected in the command line output, which shows the KS statistic to be lower in the distribution at distance = 1. This means that the degree distribution at distance = 1 more closely resembles a power law distribution. This indicates that the network of companies resembles a scale-free network. Most nodes have very few connections (relatively speaking), but a few nodes have an extremely large number of connections.