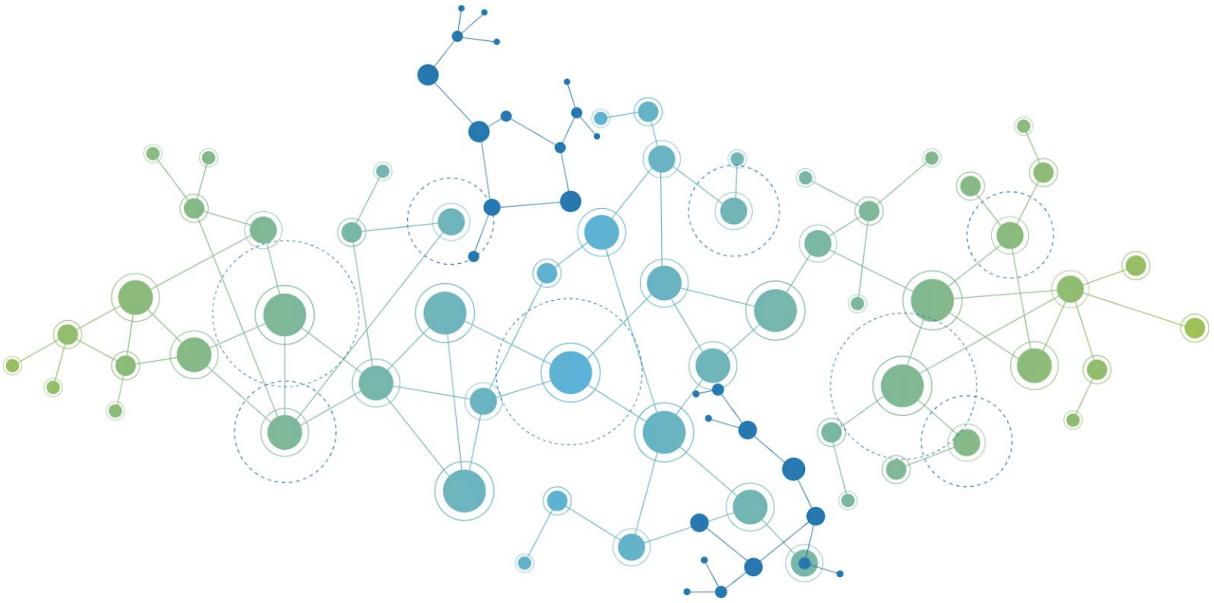
U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO



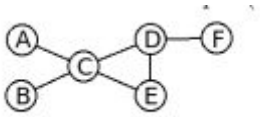
Homework #1

Network Science

Henrique Branquinho **201804341**

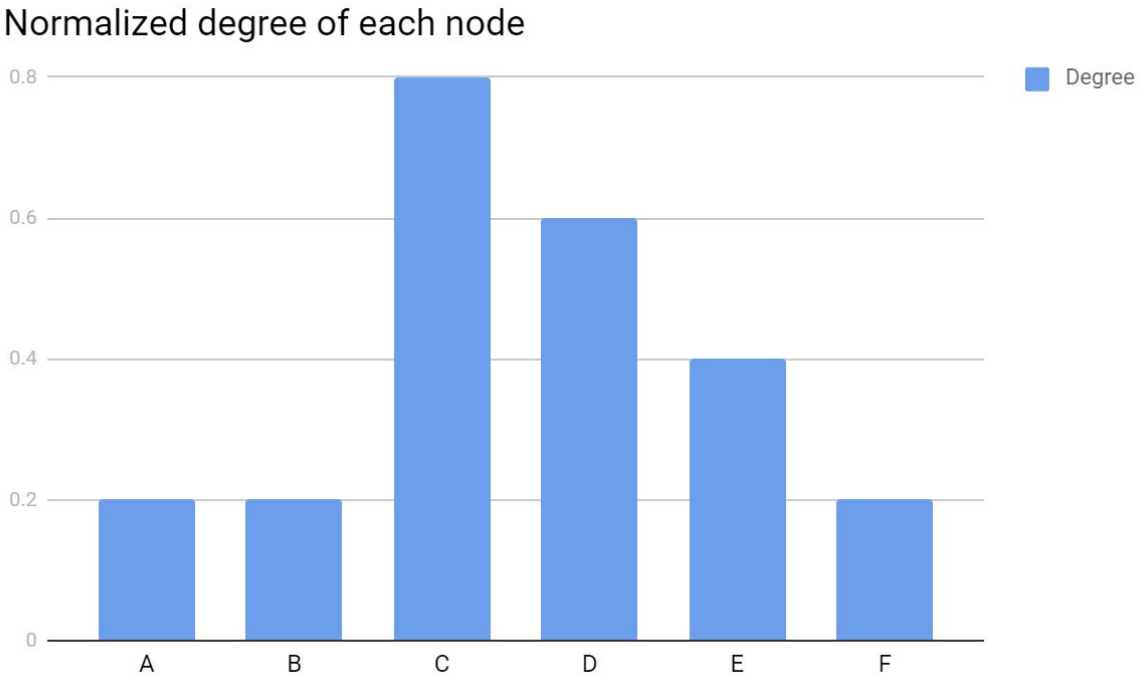
Março, 2020

1. Consider the simple (undirected) network represented by the following graph:



a) Show the degree of each node and make a plot of its (normalized) degree distribution.

Node	Degree	Normalized (/ (N-1))
A	1	0.2
B	1	0.2
C	4	0.8
D	3	0.6
E	2	0.4
F	1	0.2



b) Calculate the diameter and the average path length of the network

Length of shortest path between i and j (h_{ij}) :

A B - 2	B C - 1	C E - 1
A C - 1	B D - 2	C F - 2
A D - 2	B E - 2	D E - 1
A E - 2	B F - 3	D F - 1
A F - 3	C D - 1	E F - 2

Diameter = max (shortest path) = **3**

Average path length $\bar{h} = \frac{1}{2E_{\max}} \sum_{i,j \neq i} h_{ij}$ =

= 26 / (2 * 15) = 13 / 15 = **0.867**

c) Calculate the local clustering coefficient of each node and the average local clustering coefficient of the entire network.

$$C_i = \frac{2e_i}{k_i(k_i - 1)}$$

where e_i is the number of edges between the neighbors of node i

$C_A = 0$ ($k_A = 1$)
$C_B = 0$ ($k_B = 1$)
$C_C = \frac{1}{6}$ ($e_C = 1$; $k_C = 4$)
$C_D = \frac{1}{3}$ ($e_D = 1$; $k_D = 3$)
$C_E = 1$ ($e_E = 1$; $k_E = 2$)
$C_F = 0$ ($k_F = 1$)

Avg. Local Clustering Coefficient: $C = \frac{1}{N}\sum_{i}^{N} C_i$

= ⅙ * (0 + 0 + ⅙ + ⅓ + 1 + 0) = **0.25**

d) Calculate the (normalized) betweenness centrality and closeness centrality of each node.

Node	Paths	Normalized Betweenness centrality (/ [(n-1)(n-2)/2] = / 10)
A	-	**0**
B	-	**0**
C	(A,B) -> 1/1 (A,D) -> 1/1 (A,E) -> 1/1 (A,F) -> 1/1 (B,D) -> 1/1 (B,E) -> 1/1 (B,F) -> 1/1	**0.7**
D	(A,F) -> 1/1 (B,F) -> 1/1 (C,F) -> 1/1 (E,F) -> 1/1	**0.4**
E	-	**0**
F	-	**0**

Cc(A) = 5 /(2 + 1 + 2 + 2 + 3) = 5 / 10 = **0.5**
Cc(B) = 5 /(2 + 1 + 2 + 2 + 3) = 5 / 10 = **0.5**
Cc(C) = 5 /(1 + 1 + 1 + 1 + 2) = 5 / 6 = **0.83**
Cc(D) = 5 /(2 + 2 + 1 + 1 + 1) = 5 / 7 = **0.714**
Cc(E) = 5 /(2 + 2 + 1 + 1 + 2) = 5 / 8 = **0.625**
Cc(F) = 5 /(3 + 3 + 2 + 1 + 2) = 5 / 11 = **0.4545**

Closeness Centrality:

$$C_c(i) = \left[\sum_{j=1}^{N} d(i,j)\right]^{-1}$$

Normalized Closeness Centrality

$$C_C'(i) = (C_C(i))/(N-1)$$

2. Which network clustering metric (global or average cluster coefficient) gives more weight to high degree nodes? Why?

$$C_{global} = \frac{3 \times triangles}{number\ of\ triplets}$$

Both metrics use the notion of triangles and triplets to calculate the cluster coefficient. Defining a wedge as a length-2-path, the denominator of the local cluster coefficient k_i (k_i - 1) is the number of wedges centered at node i and the coefficient 2 in the numerator comes from the fact that each triangle at node i closes two wedges.

With this in mind, the local clustering coefficient is the fraction of closed wedges centered at node i.

The global clustering coefficient is the fraction of closed wedges in the entire network.

“Both the average and global clustering coefficient are weighted averages of local clustering coefficients. The weight in the global case is the number of wedges centered at each node, which is at the order of the degree squared, and thus places more weight on high degree nodes. The weight in the average case is uniform amongst all nodes and thus (implicitly) places more weight on low-degree nodes as they outnumber high-degree nodes in real-world networks with heavy-tailed degree distributions. “ [1].

Global clustering coefficient gives more weight to high degree nodes.

3. Using Gephi

a) What is the number of airports and flights in the network?

There are 3147 airports in the network and 66679 flights. (Number of nodes and edges in statistics).

b) On average, an airport has how many outgoing flights? And to how many different airports?

21,88 flights (Avg. degree) to 11,699 airports (Avg. degree in network imported with “Sum” merge strategy for edges) .

c) What is the diameter and average path length of the network?

Diameter - 13
Avg path length - 3.969

In Statistics, ran “Network Diameter”

d) What is the pair of airports with more flights between each other (and how many flights)?

The pair of airports is (3682 , 3830) with 39 flights . I imported the network with “Undirected” edges and “Sum” option for edge merging strategy. Then, just sorted by “Weight” column.

3682 - Hartsfield Jackson Atlanta International Airport
3830 - Chicago O'Hare International Airport

e) List the top-3 of the airports that have flights to the highest number of other airports

1 - Frankfurt am Main Airport (239)
2 - Charles de Gaulle International Airport (237)
3 - Amsterdam Airport Schiphol (232)

After importing network with “Sum” option for edge merging strategy, calculate Avg. Degree in Statistics and sort “Out-degree” column.

f) List the top-3 of the airports with highest normalized betweenness centrality

1 - Los Angeles International Airport (0.85)
2 - Charles de Gaulle International Airport (0.72)
3 - London Heathrow Airport (0.62)

After calculating “Avg. Path length” in Statistics and sorting the “Betweenness Centrality” column

g) Consider Ted Stevens Anchorage International Airport. What is its global ranking in terms of betweenness centrality and out-degree? Can you explain the discrepancy? Indicate another airport with the same kind of behavior (high betweenness centrality but relatively low out-degree)

Out-degree : 59 (272º)
Betweenness centrality 0.053 (8º)
This is probably because this airport connects important airports that otherwise wouldn’t be connected. Even though its degree is low, it works as a bridge between very important airports, that is, airports with many connections.

Faa'a International Airport
Out : 37 (394º)
Betweenness centrality 0.017 (40º)

h) List the top-3 of countries with the highest number of airports

1 - United States (16,68% of total airports)
2 - Canada (6,48% of total airports)
3 - China (5,5% of total airports)

Filtering with “Partition” by country

i) List the top-3 of airlines with the highest number of flights.

-> Appearance -> Edges -> Partition -> “airline”

1st - FR (Ryanair) (3.73%)
2nd - AA (American Airlines) (3.53%)
3rd - UA (United Airlines) (3.26%)

j) What is the number of domestic flights inside USA ?

10487 (Filtering airports with country “United States”, all edges left are between american airports / Inter Edges by country filter)

k) How many airports in China fly to at least 50 other airports?

21 (Filtering with INTERSECTION with Topology -> Out Degree Range (>= 50) and Attributes -> Equal -> Country = “China” , after importing the network with “Sum” merge strategy)

l) How many flights are there between Brazil and Portugal?

24
Intra Edges by country filter. Selected Brazil and Portugal

m) Consider a network formed only by Ryanair flights. What is the number of nodes and edges of its giant component? Considering only this giant component, what is the most important airport in terms of closeness centrality?

Code = “FR”
Nodes = 176
Edges = 5632
First, we filter by airline = “FR”. With this graph, we calculate Connected Components from Statistics. Then we use Topology -> Giant Component.
Then we calculate Avg. Path Length on the resulting graph to get Closeness Centrality.
The airport is London Stansted Airport with a closeness centrality of 0.792

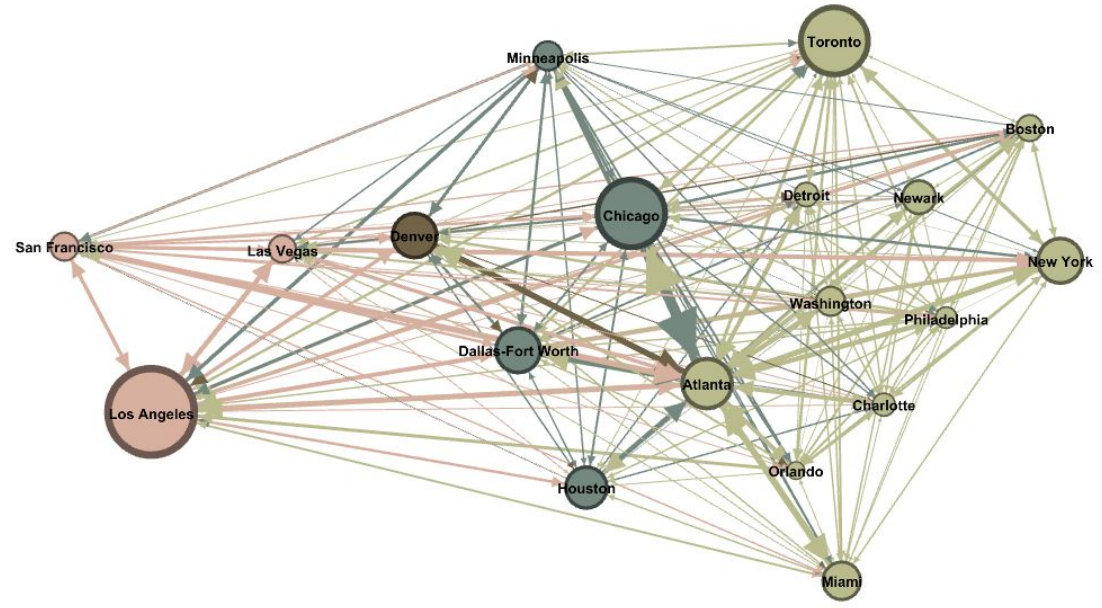
n) How many airport are reachable from Francisco de S´a Carneiro Porto Airport in 1 flight? And in at most 2 flights? And in at most 3 flights?

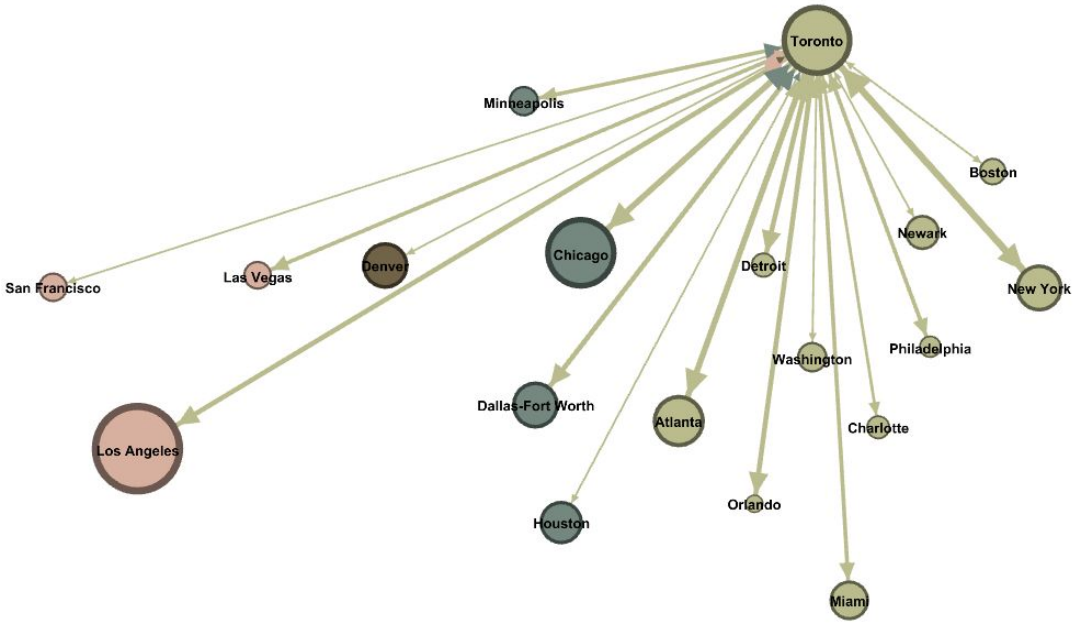
ID = 1636
1 flight - 61
2 flights - 755
3 flights - 2376
With Topology -> Ego Network

o) Create an image showing the flight network between american and canadian airports with more than 100 destinations in the global network. The size of the nodes should reflect the global betweenness centrality, and their colors should be different for each time zone. Nodes should be labeled with the city name. Try to make your image as comprehensible and aesthetically pleasing as possible.

Importing the network with “Sum” merging strategy .
Calculating betweenness centrality with “Avg. Path Length” in Statistics.
Calculating number of different destinations with out-degree by calculating “Avg. Degree” in Statistics.
Filtering with:

- Intersection
    - Union
        - Country = “United States”
        - Country = “Canada”
    - Attributes -> Range -> Out-Degree > 100
    - (Second picture) Intra-Edges -> Country


Toronto
Minneapolis
Boston
Detroit
Newark
Chicago
Denver
San Francisco
Las Vegas
New York
Washington
Philadelphia
Dallas-Fort Worth
Atlanta
Charlotte
Los Angeles
Orlando
Houston
Miami


Toronto
Minneapolis
Boston
Newark
Chicago
Denver
Detroit
San Francisco
Las Vegas
New York
Philadelphia
Washington
Dallas-Fort Worth
Atlanta
Charlotte
Los Angeles
Orlando
Houston
Miami

Erdos-Renyi Model

For this exercise, I divided the code in 3 separate files, each one corresponding to an exercise.

- **graphGen.py** - This is the file with the Graph class that is used to generate the graph according to the model. It should be used as "**python3 graphGen.py N p > outfile**" where **N** is the number of nodes in the graph, **p** is the probability for the creation of each edge and **outfile** is the name of the file where we want to save the network.

  The generation of the graph is made with the function Graph.**generateGraph**(p). Initially, the graph is created with N nodes and no edges. Then, we call Graph.generateGraph(**p**) and for every edge (I only go through each edge once because I assume the graph is undirected) I "toss a coin" and create an edge according to **p**.

- **giantComponent.py** - This is the file that calculates the giant component of a graph. It should be used as **"python3 giant-component.py filename**", where filename is the name of the file that contains the graph. It uses a BFS on each node to try and find the biggest component, marking each node passed as visited and iterating over all non visited nodes, performing a BFS on them. The algorithm is O(n) as it only passes through each node once.

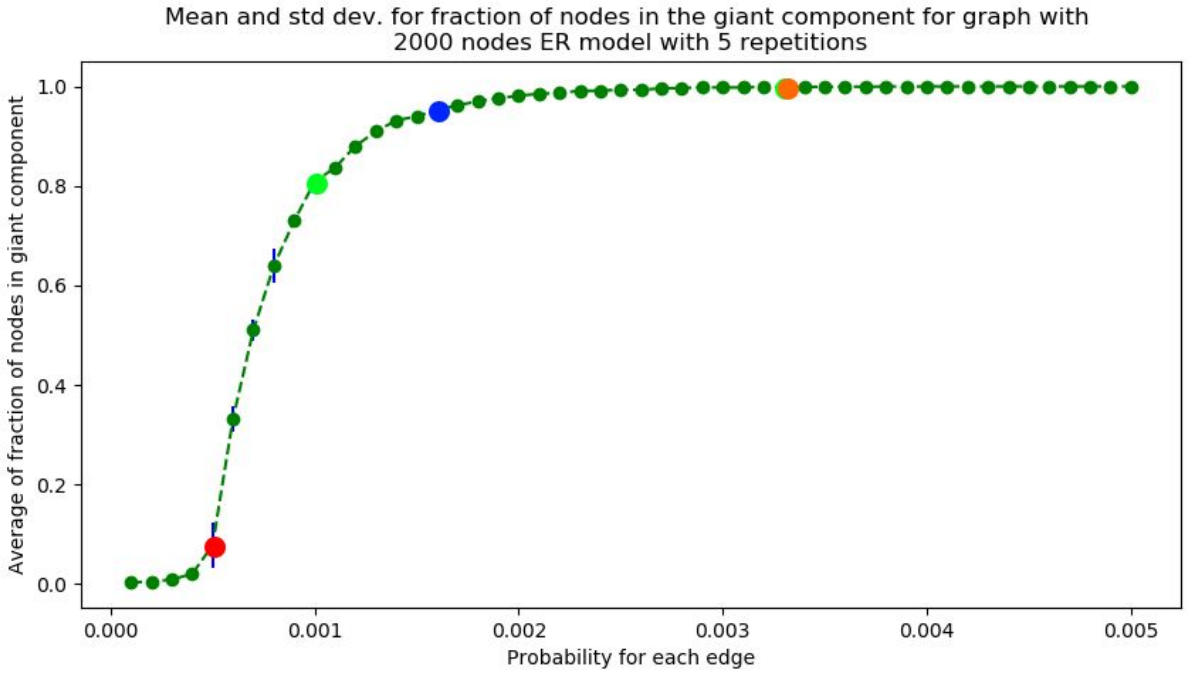
  Outputs the number of components and the size of the giant component.

- **plotter.py** - This is the file that creates several networks according to the specifications in exercise **6**. It should be used as "**python3 plotter.py N pmin pmax step repetitions**", where N is the number of the nodes for the graphs, **p** varies between [pmin,pmax] with **step** defined by user. **repetitions** is the number of graphs that should be created for each configuration $G_{N,p}$. Since they have a random nature, I decided to allow the repetition of graph creations. The creation of graphs and calculation of giant components is made in the function **generateGraphs(N,pmin,pmax,step,repetitions,dirname)**.

  I store the giant component size in each graph, and then I use the list of created graphs in the function **plotGC(N,repetitions,dirname,graphs)** to plot the fraction of nodes in the giant component according to the probability of creating an edge. Since I allow repetitions, the values shown are the average and I also plot the standard deviation.

dirname is a directory name created according to the current time and date. Every graph created is stored in this folder, along with the created plot in png format.

Example ("python3 plotter.py 2000 0.0001 0.005 0.0001 5"):



The plot is what I was expecting from the theoretical properties of giant components in these networks according to the average degree.

Initially, there is no giant component. When **p = 0.0005**, I noticed the emergence of a giant component, a small one nevertheless. With p = 0.0005 , the average degree is equal to 1.

From p = 0.0005 on forward, there is a sudden increase in the size of the giant component, while the average degree is still between 1 and 2. When the average degree is 2 (**p = 0.001**) almost all nodes are in the giant component (~ 80%). When **p = 0.0016** ≃ log(N)/(N-1) (average degree is 3.2) the percentage of nodes in the giant component is close to 1, meaning there are very few isolated nodes. From this point on, the increase in the size of the giant component is more subtle, and it grows gradually up until it reaches **p > 0.0033**, ≃ 2*log(N)/(N-1) where all nodes are in the giant component (average degree is 6.6).

Barabási-Albert Model

For this exercise, I divided the code in two files, one for each exercie.

- **graphGen.py -** This is similar to the previous file used in the ER model. It should be used as ”**python3 graphGen.py N[number of nodes] m0[Size of original network] m[Number of connections per node added] > outfile**”. I use an adjacency matrix to store the graph, along with a list that stores repeated nodes according to their degree (if node A has degree 3, it appears 3 times in this list).

  Initially the graph is created with no edges. I use a function **generateGraph(M0,M)** that adds edges between all first M0 nodes. Then, while the graph doesn't have N nodes, I choose randomly M nodes from the **degreeNodes** list mentioned above and add edges between them and the new node I'm creating.

- **degreePlotter.py** - This is the file used to plot the degree distribution of a graph. It should be used as “**python3 degreePlotter.py filename**” where filename is the name of the graph file.

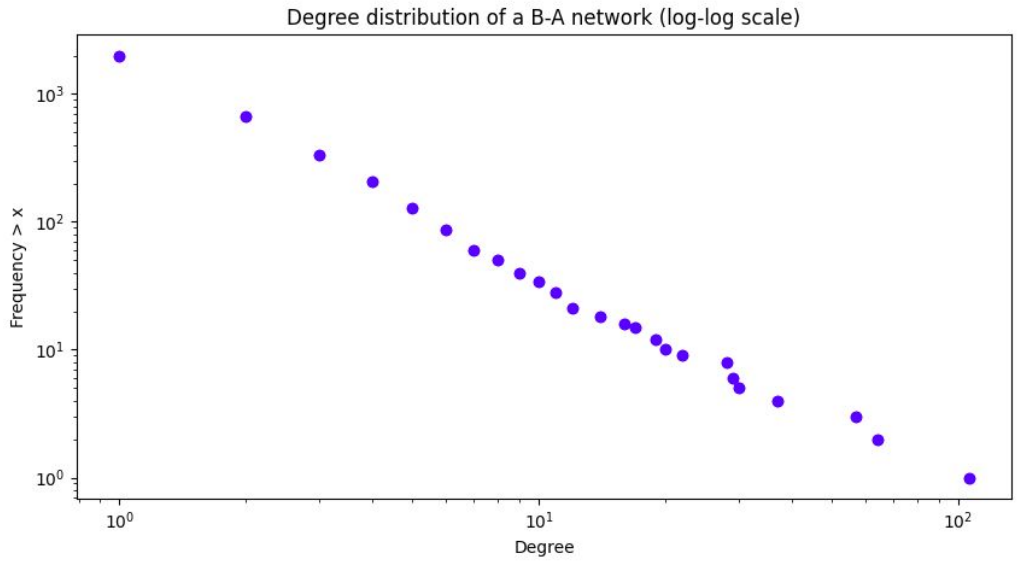
  It reads the graph and calls the function Graph.**getDegreeDistribution**(), which returns a dictionary where keys are degrees and values are the frequency of nodes with that degree.

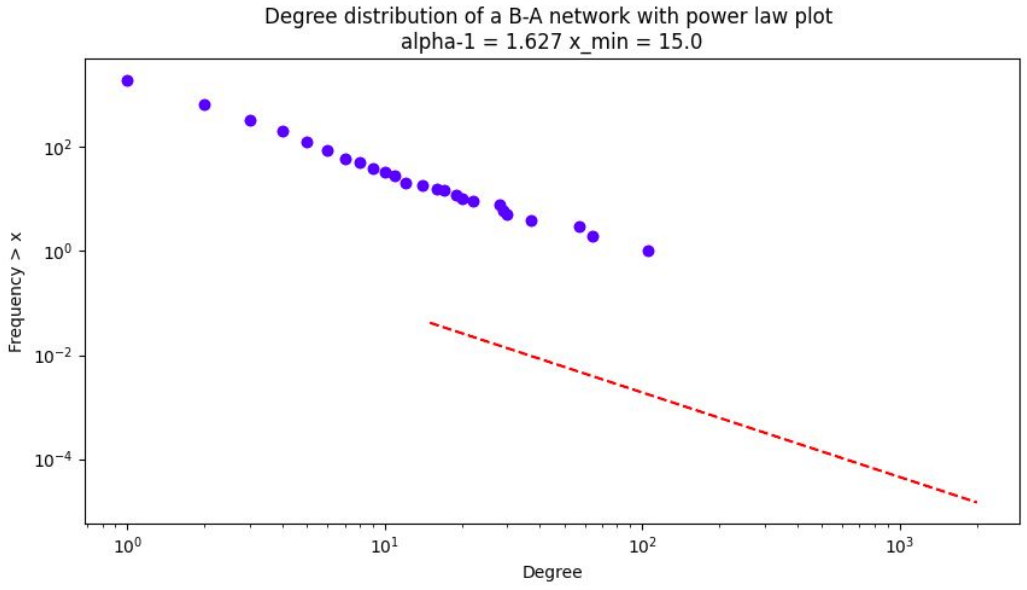
  Then, I perform cumulative binning with the function numpy.**cumsum**().

  I plot the values on a log-log scale and store the plot in a file called filename_plot.png. Then, I fit a power law to the values with the **powerlaw** module, print the alpha and the x_min calculated by the model (instead of explicitly defining a minimum degree value, I let the fit function choose the best one), and I try to plot the fitted model along with the degree distribution, saving it in a file called filename_plot2.png. However, I had some trouble with the plotting of the power law. In the final graph, the power law plot is dislocated from the points, even though the slope appears to match the one of the fitted line of the points. I believe it is because the constant to the power law function is being miscalculated.

Results:

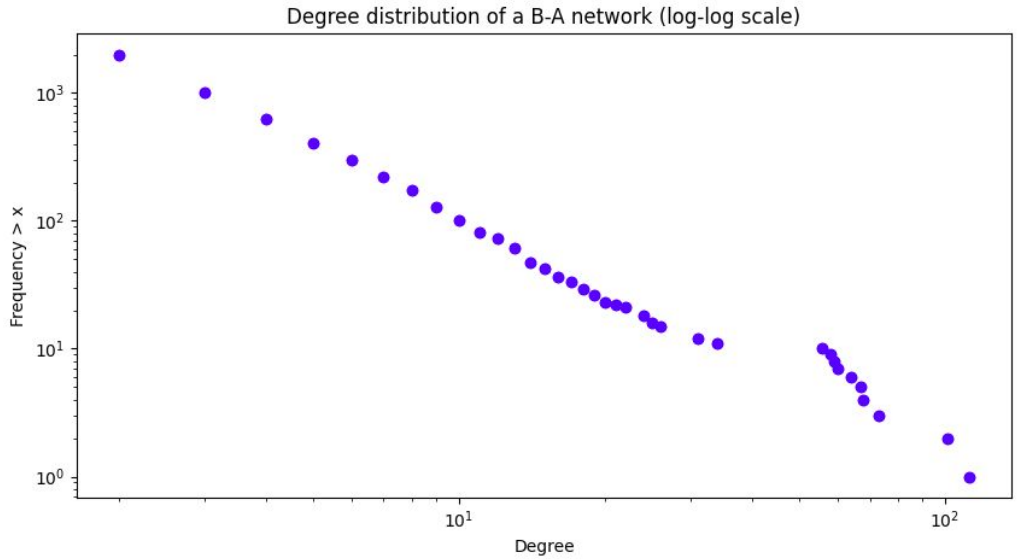
- ba1.txt

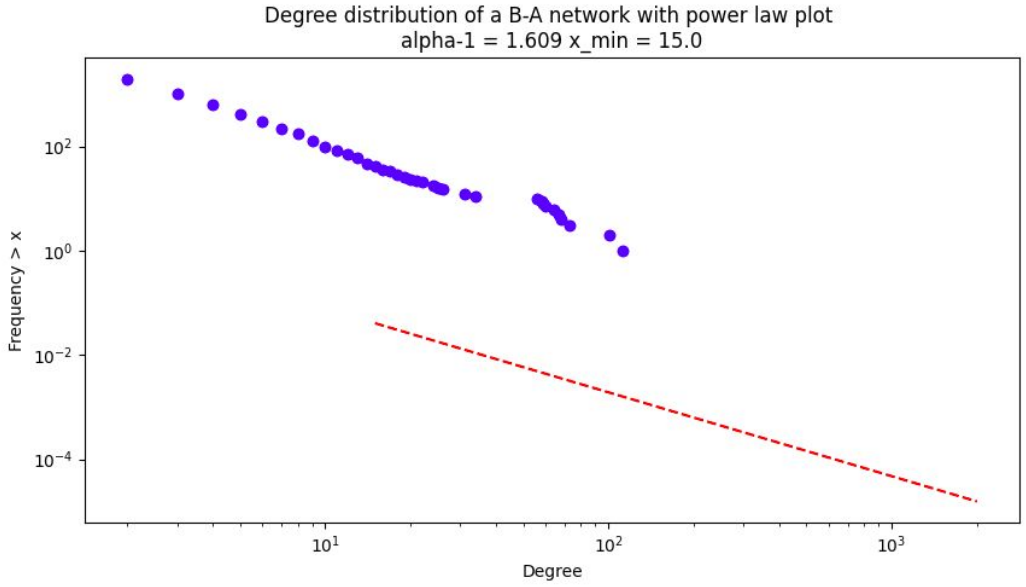




alpha = 1.627 + 1 = **2.627**

- ba2.txt





alpha = 1.609 + 1 = **2.609**

References

[1] -Yin H, Benson A, Leskovec J, *The Local Closure Coefficient: A New Perspective On Network Clustering*, 2019