Algorithmic Experiments of Real-World Phenomena
Project 3 Phase 2 Report

Introduction

In this report, we analyze various graph algorithms applied to models of real-world networks. Specifically, we test the **Diameter**, **Clustering Coefficient**, and **Degree Distribution** algorithms on two types of random graphs namely, Erdos Renyi random graph and Barabasi Albert random graphs. The goal of this study is to explore how these algorithms behave on different random graph types, especially focusing on the growth patterns of graph properties as the number of vertices in the graph increases.

Erdos-Renyi random graph : G(n,p), with p =2(ln n)/n
Barabasi-Albert random graph : Generated with the parameter d=5 as the number of neighbours each new vertex chooses.

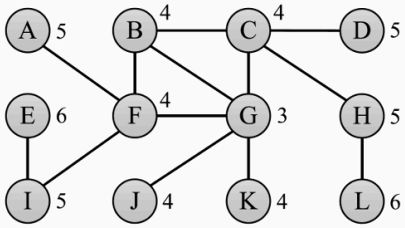
Algorithms

Diameter Algorithm : The diameter of a graph is defined as the longest shortest path between any two vertices in the graph.
For this, I made use of bfs and a heuristic to be able to find the diameter faster than having to check every path in the graph.
Here is the heuristic used :

Heuristic Idea 2

1. Let r be a random vertex and set $D_{max} = 0$.
2. Perform a BFS from r.
3. Select the farthest node, w, in this BFS.

- If the distance from r to w is larger than D_{max}, set D_{max} to this distance, let r = w, and repeat the above two steps.



Clustering Coefficient Algorithm : The clustering coefficient of a vertex measures how close its neighbors are to being a complete subgraph.
Formula used : C = (3×number of triangles)/number of 2 edge paths

I made use of a degeneracy list to count the number of triangles, using the following algorithm:

1. Initialize an output list, L, to be empty.
2. Compute a number, d_v, for each vertex v in G, which is the number of neighbors of v that are not already in L. Initially, d_v *is* just the degrees of v.
3. Initialize an array D such that $D[i]$ contains a list of the vertices v that are not already in L for which $d_v = i$.
4. Let N_v be a list of the neighbors of v that come before v in L. Initially, N_v is empty for every vertex v.
5. Initialize k to 0.
6. Repeat n times:
   - Let i be the smallest index such that $D[i]$ is nonempty.
   - Set k to $\max(k, i)$.
   - Select a vertex v from $D[i]$. Add v to the beginning of L and remove it from $D[i]$. Mark v as being in L (e.g., using a hash table, H_L).
   - For each neighbor w of v not already in L (you can check this using H_L):
     - Subtract one from d_w
     - Move w to the cell of D corresponding to the new value of d_w, i.e., $D[d_w]$
     - Add w to N_v

Pseudocode for the denominator (number of 2 edge paths) :

- For each vertex v, let deg(v) denote its degree.
- The number of paths of length 2 with v in the middle is deg(v) choose 2 = deg(v)(deg(v)-1)/2.
- So, to get the denominator for C, sum up deg(v)(deg(v)-1)/2 for all vertices, v, in G.

Degree Distribution Algorithm : The degree distribution of a graph describes the number of vertices with each possible degree. We calculate the degree of each vertex and plot the distribution. The degree distribution can be used to identify patterns such as power-law behavior.

Erdos-Renyi Graph generation Algorithm : I used G(n,p), with p =2(ln n)/n to generate the Erdos-Renyi graph, following the pseudocode below.

Input: number of vertices n, edge probability $0<p<1$
Output: $G=(\{0,\ldots,n-1\},E)\in\mathcal{G}(n,p)$
$E\leftarrow\emptyset$
$v\leftarrow 1;\ w\leftarrow -1$
while $v<n$ **do**
  draw $r\in[0,1)$ uniformly at random
  $w\leftarrow w+1+\lfloor\log(1-r)/\log(1-p)\rfloor$
  while $w\geqslant v$ **and** $v<n$ **do**
    $w\leftarrow w-v;\ v\leftarrow v+1$
  if $v<n$ **then** $E\leftarrow E\cup\{v,w\}$

Barabasi-Albert Graph generation Algorithm : Using degree = 5, I generated the graph following the pseudocode below.

Input: number of vertices n
  minimum degree $d\geqslant 1$
Output: scale-free multigraph
$G=(\{0,\ldots,n-1\},E)$
M: array of length $2nd$ // M is an array of edges chosen so far.
for $v=0,\ldots,n-1$ **do**
  for $i=0,\ldots,d-1$ **do**
    $M[2(vd+i)]\leftarrow v$ // Each vertex v appears d_v times in M.
    draw $r\in\{0,\ldots,2(vd+i)\}$ uniformly at random
    $M[2(vd+i)+1]\leftarrow M[r]$
$E\leftarrow\emptyset$
for $i=0,\ldots,nd-1$ **do**
  $E\leftarrow E\cup\{M[2i],M[2i+1]\}$

Experimental Setup

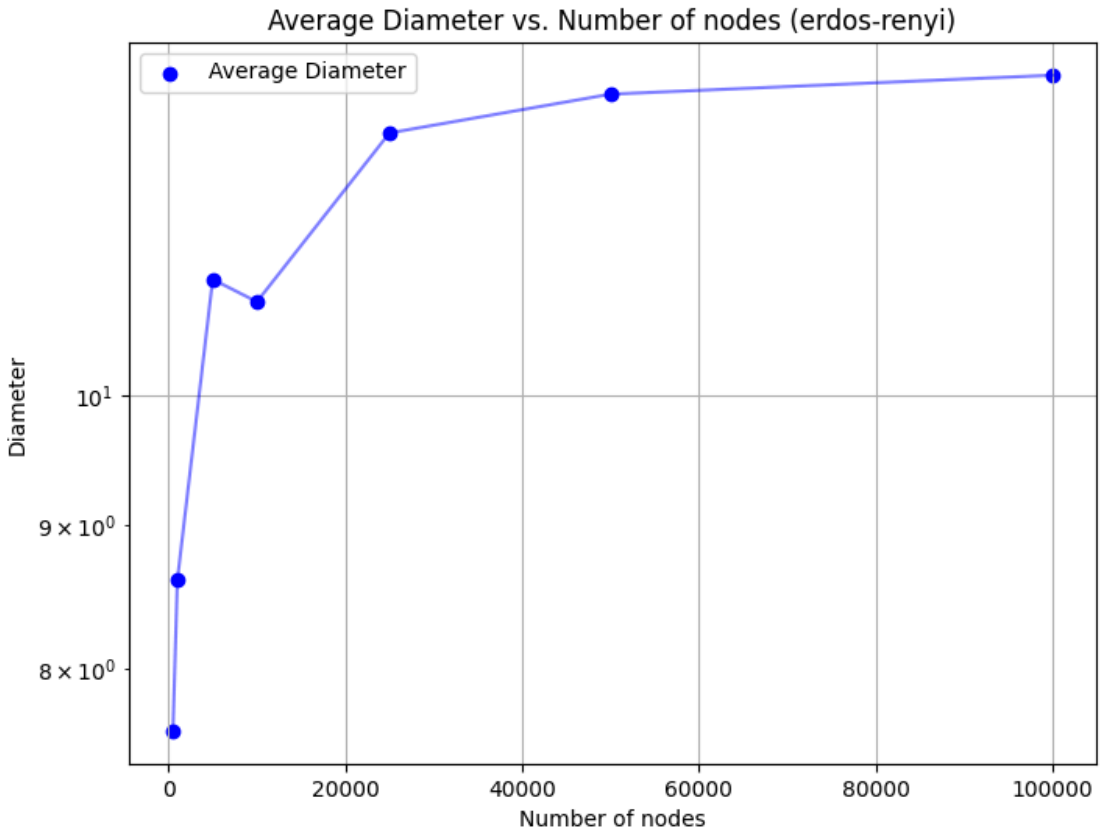
We tested the algorithms on random graphs of various sizes, as follows:

- For the **Diameter** and **Clustering Coefficient** algorithms, we used graphs with sizes n=500, 1000, 5000, 10000, 25000, 50000, 100000. These graphs were generated multiple times to obtain an average value for each of the properties, ensuring that the results were statistically meaningful.
- For the **Degree Distribution** algorithm, we used graphs with sizes n=1000, 10000, 100000 to analyze how the degree distribution behaves at different graph scales.

Results

Diameters

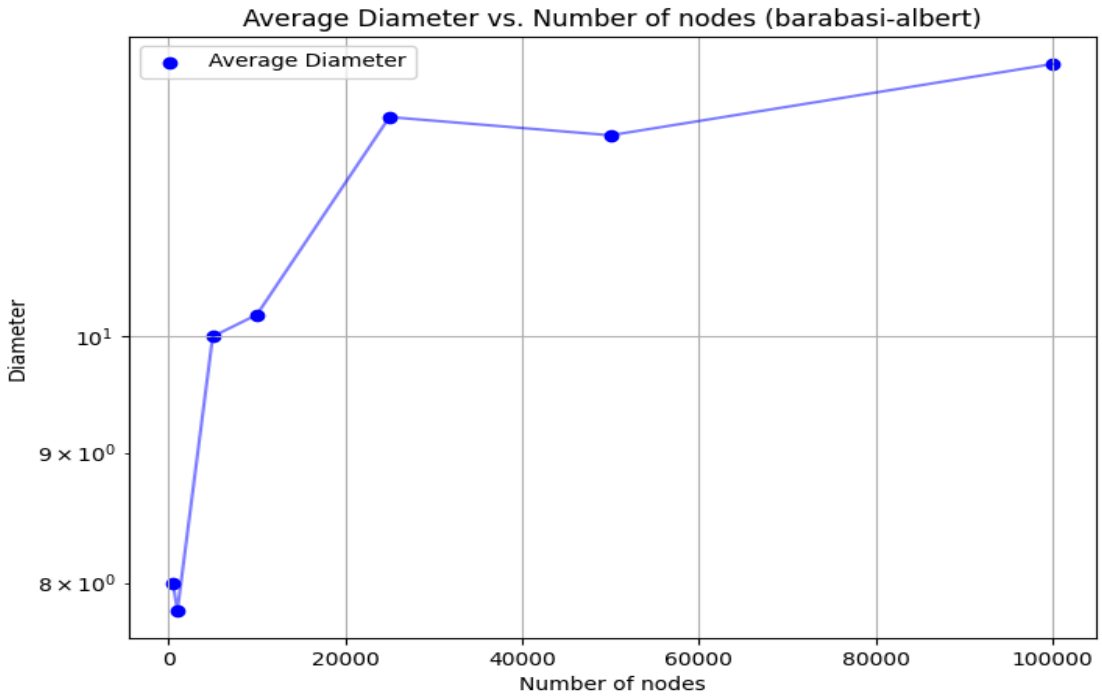
Erdos-Renyi



From the graph, we can see that the values grow as a function of n.

With regards to the function log n, between n=0 and n=100000, we can see that the graph is proportional to the function log n, in between it does look like it grows a little faster but it comes down to become proportional.

Barabasi-Albert

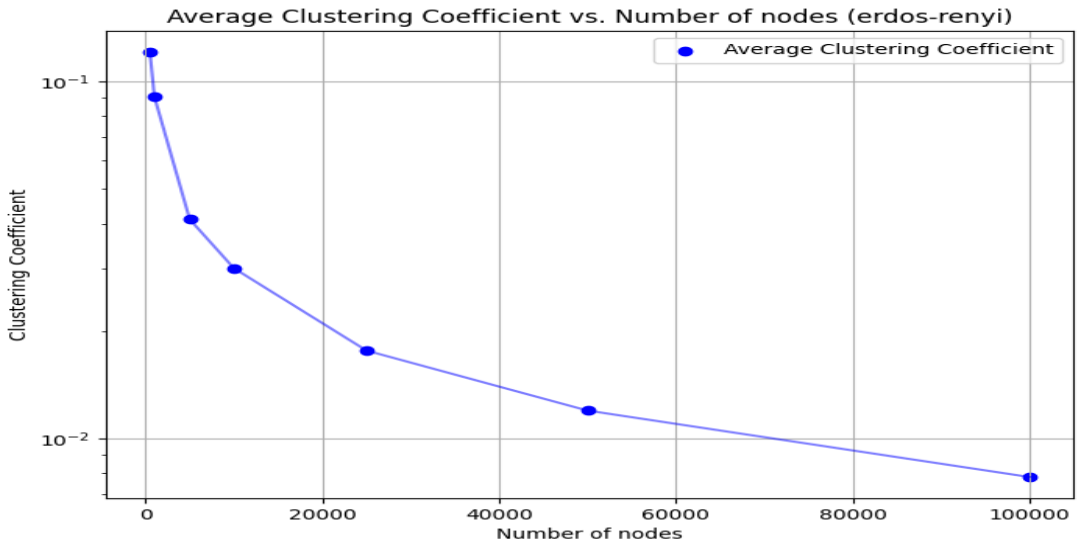


From the graph, we can see that the values grow as a function of n.

With regards to the function log n, between n=0 and n=100000, we can see that the graph is proportional to the function log n, in between it does look like it grows a little faster but it comes down to become proportional.

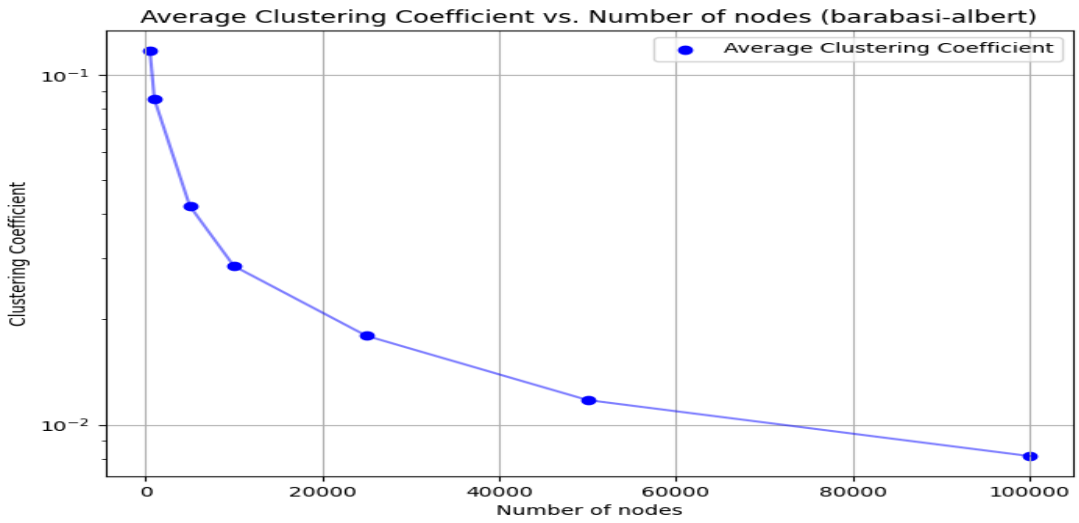
Clustering Coefficients

Erdos-Renyi



From the graph, we can see that the values **decrease** as a function of n.

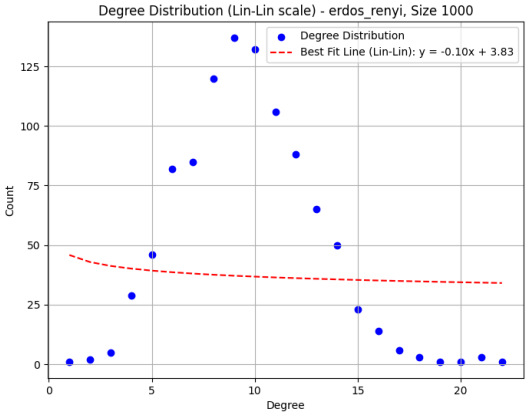
Barabasi-Albert

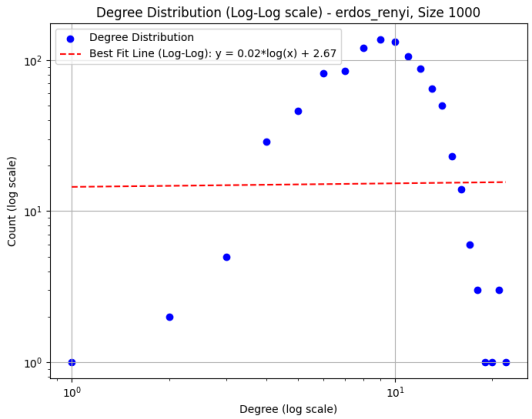


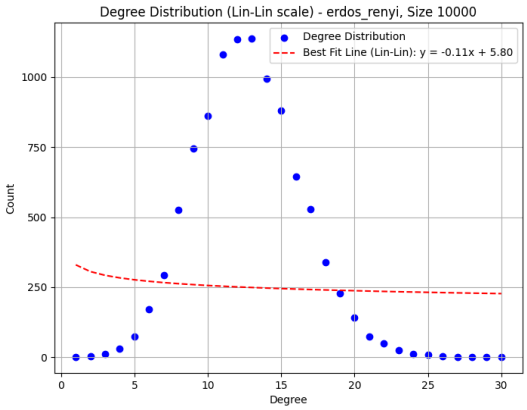
From the graph, we can see that the values **decrease** as a function of n.

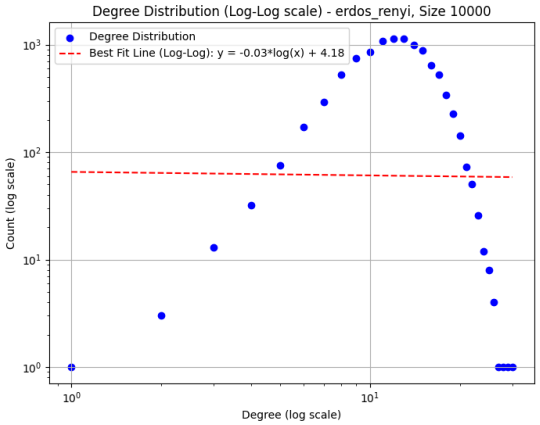
Degree Distributions

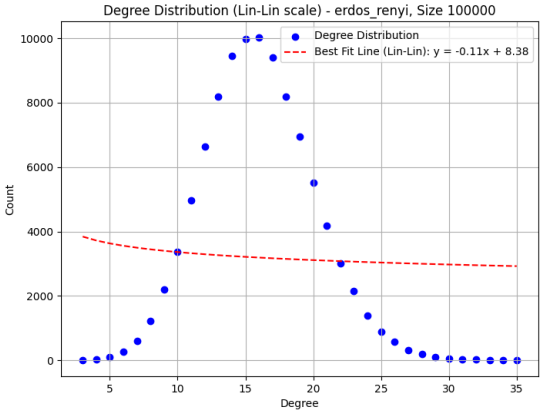
Erdos-Renyi

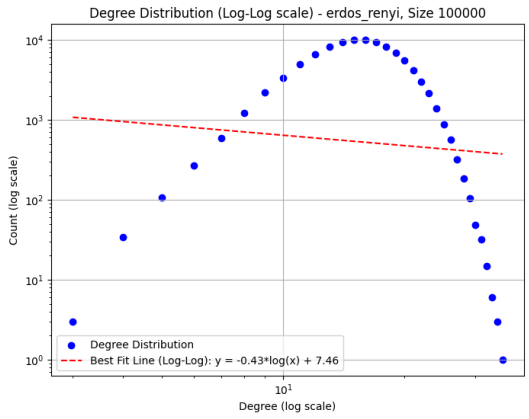






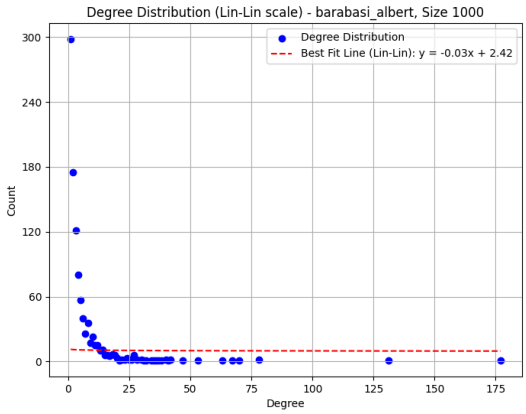


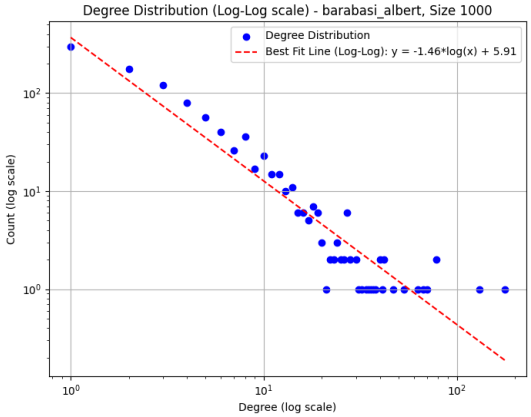


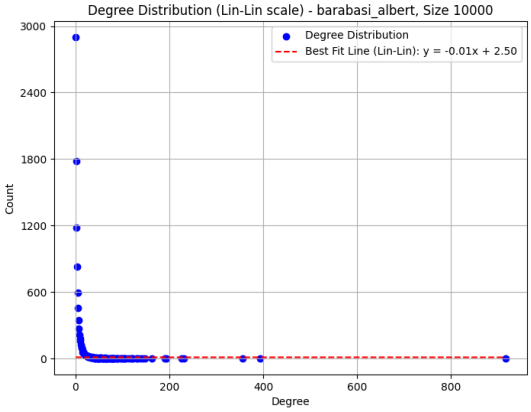


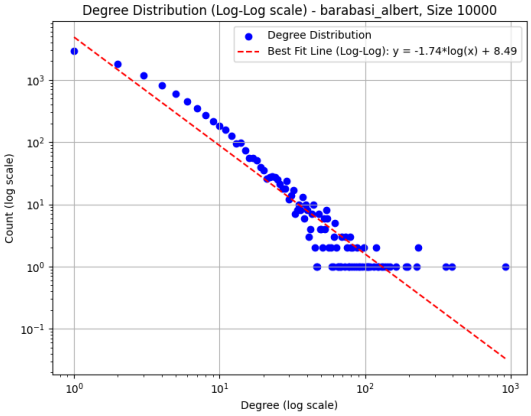
From the graphs above, we can see that Erdos-Renyi random graphs **do not follow a power law** as they do not have a pattern similar to the curve of a power law.

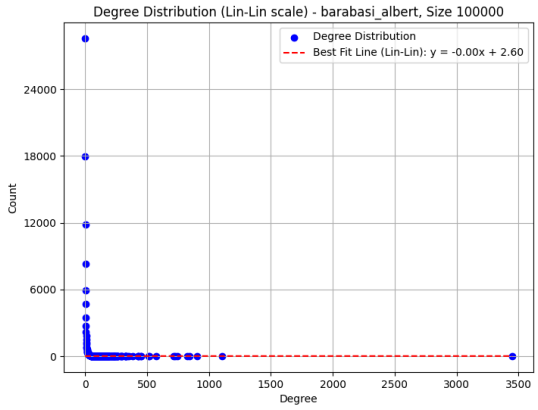
Barabasi-Albert

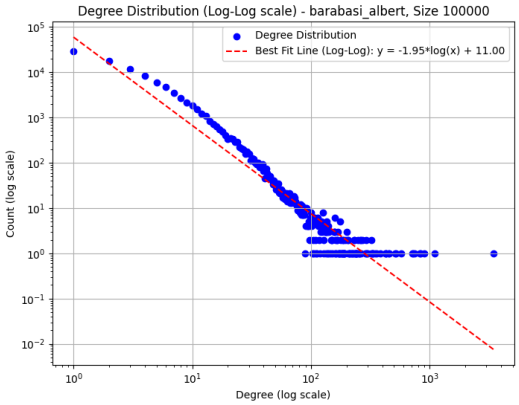












From the graphs above, we can see that Barabasi-Albert random graphs **do follow a power law** as they have a pattern **similar to the curve of a power law**. Observing the slope of the best-fit line, in the log-log graphs above, we can say that the exponent of that power law is close to -2.