Social Networks and Online Markets 2022/2023
Homework 1

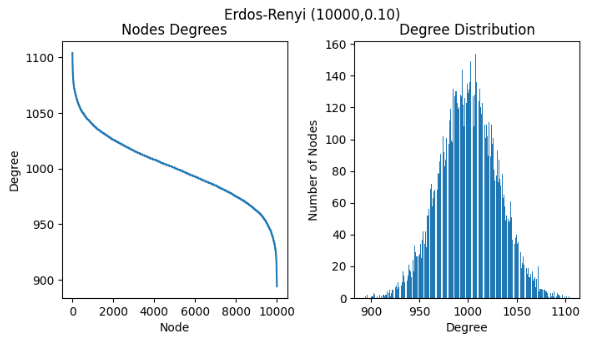
By Jacopo Masia, ID 1842884

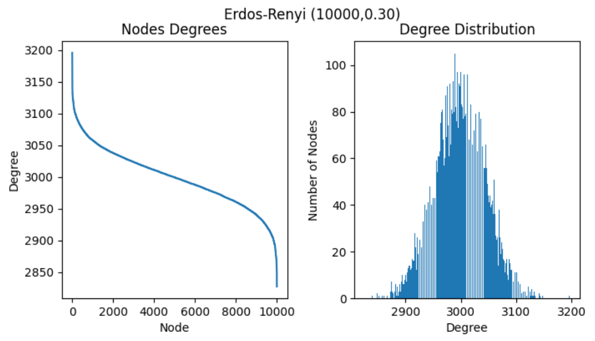
June 15, 2023

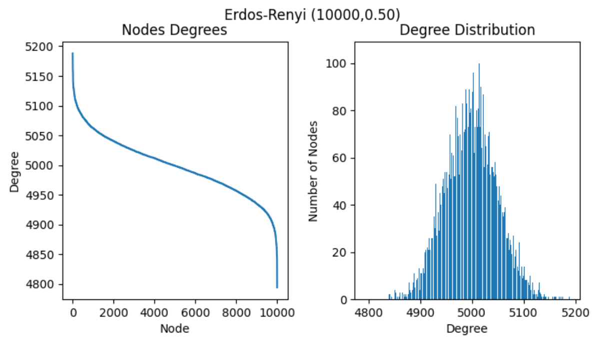
Exercise 1

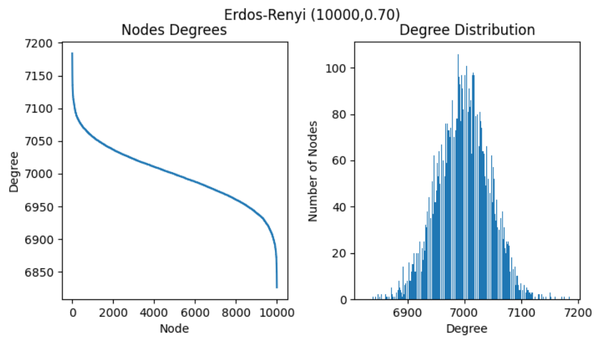
Fixing the nodes count to 10000 and testing different configurations of parameters for the three models, here are the results:

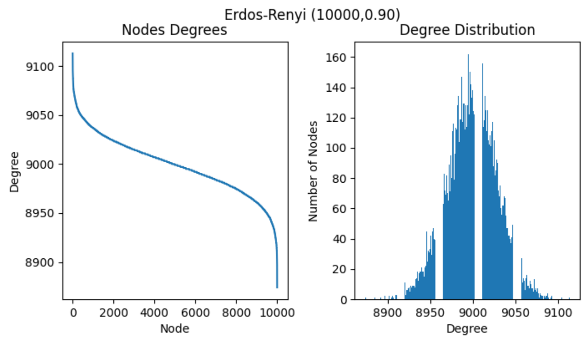
- For the **Erdos-Renyi** random graph the degree distributions and counts and the statistics are the following:







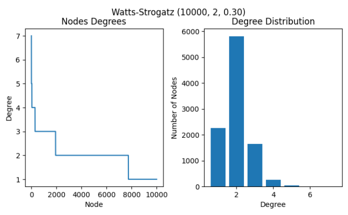


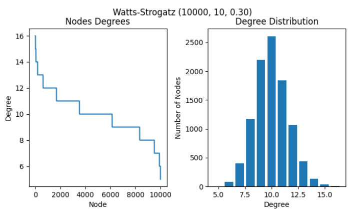


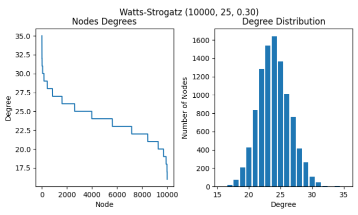
	p				
	0.1	0.3	0.5	0.7	0.9
Edges	$\sim 5m$	$\sim 15m$	$\sim 25m$	$\sim 35m$	$\sim 45m$
Diameter	2	2	2	2	2
CC	0.1	0.3	0.5	0.7	0.9

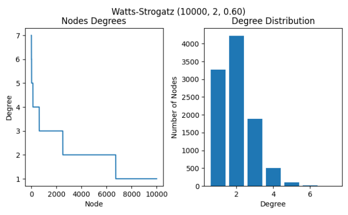
We notice that statistics, as expected, follows the binomial distribution, in particular the number of edges is $\frac{n^2 p}{2}$ and the node's degree and degree distribution are close to the CDF and PMF of a $B(n, p)$. The **diameter** is always 2 which is very small, because the number of edges is very high and the **average clustering coefficient** is approximately p, which is also expected.

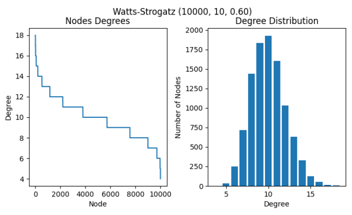
- For the **Watts-Strogatz**:

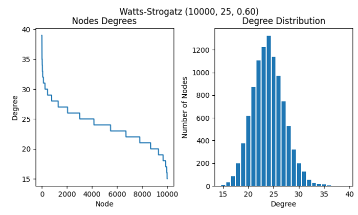


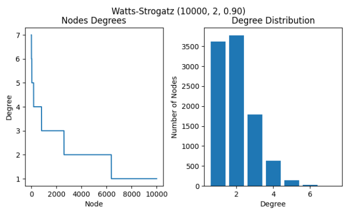


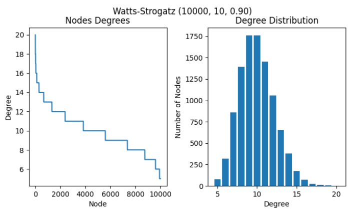


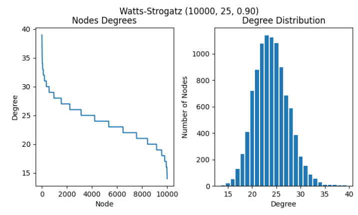








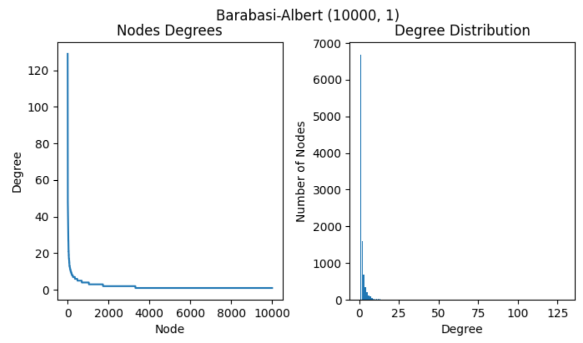


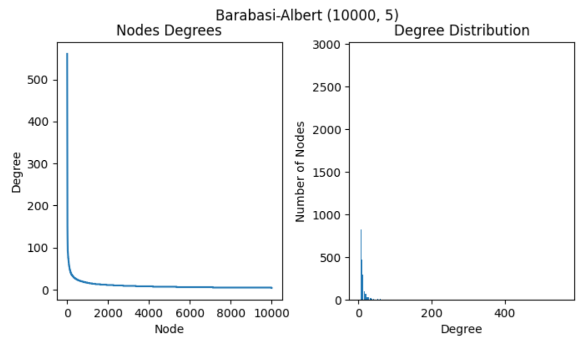


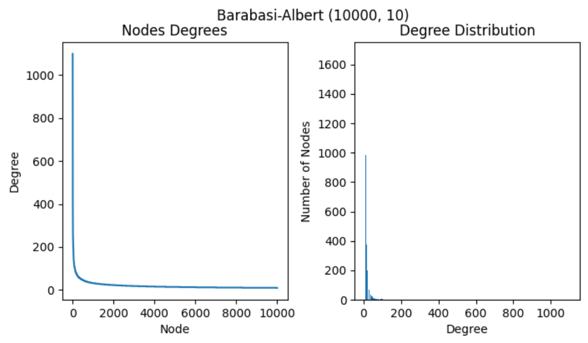
	k, β								
	$2, 0.3$	$10, 0.3$	$25, 0.3$	$2, 0.6$	$10, 0.6$	$25, 0.6$	$2, 0.9$	$10, 0.9$	$25, 0.9$
Edges	$\sim 10k$	$\sim 50k$	$\sim 120k$	$\sim 10k$	$\sim 50k$	$\sim 120k$	$\sim 10k$	$\sim 50k$	$\sim 120k$
Diameter	285*	7	5	342*	6	5	234	6	4
CC	0	0.23	0.25	0	0.043	0.051	0	0.0015	0.003

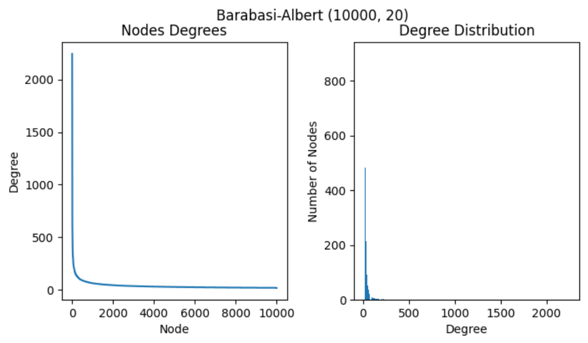
As we can see, the number of edges are obviously $\frac{nk}{2}$, since we start with that amount of edges and simply rewire them without adding or removing any edge.
The degree distribution for small values of k and β has very little values range and it's skewed to the left, the higher k and β get, the closer the distribution gets to a Gaussian.
The **diameter** is very high when k is very small, and small for higher values, that's because every node has only k edges, and to travel from a node to another far away in the initial configuration require some lucky rewires for small values of k. but in general the path between two arbitrary nodes will be inversely proportional to k. Also for $k = 2$ the graph tends to be disconnected, and the diameter values calculated above correspond to the highest diameters among all the connected components of each graph, thus if the graphs were connected they would have been even higher diameters.
The **average clustering coefficient** tends to 0 for very small values of k and grows as k grows, but the the higher β is, the less is the growth.

- And finally for the **Barabàsi-Albert** model we have









	ℓ			
	1	5	10	20
Edges	$\sim 10k$	$\sim 50k$	$\sim 100k$	$\sim 200k$
Diameter	24	6	5	4
CC	0	0.008	0.015	0.025

Here the degree distribution and count seems to follow some sort of **power law**, with the most nodes having ”small” degrees, and for all values of ℓ the distribution is the same, that’s because the model is **scale-free** which is a nice property to have for a social network.
The number of edges is $n\ell$ which is expected, and the **diameter** is relatively small for $\ell \geq 5$, but this is also expected since every new node comes with exactly ℓ edges.
The **average clustering coefficient** is also very small and grows with ℓ, that’s because the probability to link a new node is higher for nodes with already high degree, resulting in one giant and very connected component.

Exercise 2

1. If we want to calculate the probability that the shortest path between two nodes with distance k on the ring is ℓ, we first have to distinguish between the following cases:

   (a) $\ell \leq 0$ or $\ell > k$, in this case the probability is 0, because obviously a path of lenght 0 or less doesn't exist and the length of the shortest path is at most k since there is always the path on the ring between two nodes.

   (b) $\ell < k$, in this case we have to make $\ell - 1$ steps on the ring and skip the other edges using two shortcuts towards the center node, so we can write

   $$P(\ell, k) = \ell p^2 (1-p)^{\ell - 1}$$

   where p^2 represents the probability that we have the two shortcuts we need, $(1-p)^{\ell-1}$ represents the probability that shortcuts before the one we take towards the center or after the one we take from it don't exist, and ℓ are the possible paths of this type.

   (c) $\ell = k$, in this case we can either use the paths we defined above or the path on the ring, therefore we have to count the proabilities that no edges towards the center exist $((1-p)^(\ell+1))$ or that at most one exists $((\ell+1)(1-p)^{\ell})$, so we have

   $$P(\ell, k) = \ell p^2 (1-p)^{\ell-1} + (\ell+1)p(1-p)^{\ell} + (1-p)^{\ell+1}$$

   It's interesting to see that the probabilities are independent from k except for the case where $\ell = k$ or ℓ has an inadmissible value, that will be useful later.

2. To calculate the distribution $P(\ell)$ of the shortest paths in a graph with n nodes we can restrict our focus to a single node and calculate the shortest paths towards all the other nodes, since the probabilities are the same for all the nodes, and then multiply the result for the $n(n-1)$ paths between nodes. We have

$$P(\ell) = P(\ell, 1) + \cdots + P(\ell, n-1) = \sum_{i=1}^{\ell-1} 0 + \sum_{i=\ell}^{n-2} \ell p^2 (1-p)^{\ell-1} + \ell p^2 (1-p)^{\ell-1} + (\ell+1)p(1-p)^{\ell} + (1-p)^{\ell+1} =$$

$$= \sum_{i=\ell}^{n-1} \ell p^2 (1-p)^{\ell-1} + (\ell+1)p(1-p)^{\ell} + (1-p)^{\ell+1} = (n-\ell-1)\ell p^2 (1-p)^{\ell-1} + (\ell+1)p(1-p)^{\ell} + (1-p)^{\ell+1}$$

Plotting the distribution of the number of shortest paths of different lengths for different values of p we get the following result:

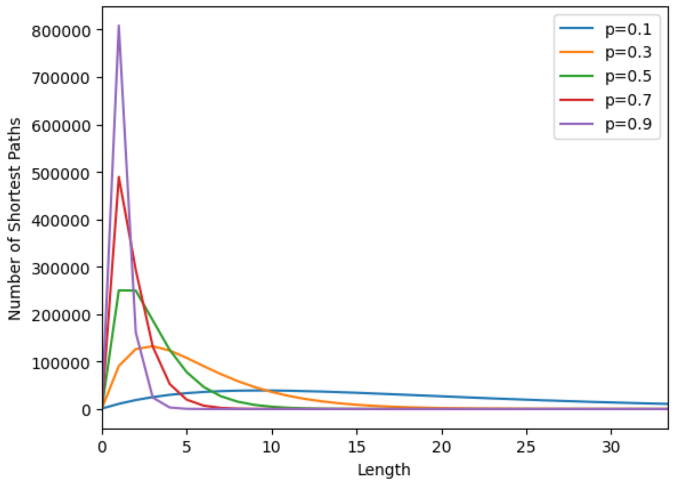


Figure 1: Distribution of the shortest paths with 1000 nodes for different values of p.

We notice that the higher p is the closer the distribution follows the typical **power law** often seen in social networks models with a lot of short shortest paths and very long tail of long shortest paths.

3. We can calculate the average shortest path $E[s]$ in a graph with n nodes as following:

$$E[s] = \sum_{i=1}^{n-1} iP(i)$$

computing the value for different sizes of n and values of p we obtain:

		p				
		0.1	0.3	0.5	0.7	0.9
n	10^1	1.55	3.72	2.87	1.85	1.22
	10^2	18.68	5.63	2.99	1.85	1.22
	10^3	18.71	5.63	2.99	1.85	1.22
	10^4	18.71	5.63	2.99	1.85	1.22

Table 1: Average shortest path between nodes.

We see that as the number of nodes grows, for each value of p we have different convergence values.

Exercise 3

1. The densest subgraph output by the greedy algorithm turns out to be the graph itself, with a density of 2.0, which is also the optimal solution (see code).
2. A minimum cut could be simply $C = (\{1\}, \{2,3,4,5,6,7,8,9,10\})$ with a size of 4.
3. First of all we calculate the normalized Laplacian $\mathcal{L} = \frac{D-A}{d}$ of the graph, where $d = 4$ since the graph is 4-regular. Then we compute the eigenvalues of $\mathcal{L}$ (see code) to find the second smallest eigenvalue which is
$$\lambda_2 = 0.25$$
Now we need to compute the conductance $\Phi(G)$ (see code) which is
$$\Phi(G) = 0.2$$
The Cheeger's inequality is then satisfied:
$$\frac{\lambda_2}{2} \leq \Phi(G) \leq \sqrt{2\lambda_2} \implies 0.125 \leq 0.2 \leq 0.7$$
4. The partition that satisfies the inequality is $P = (\{1,2,3,4,5\}, \{6,7,8,9,10\})$, infact in this case we have
$$\Phi(P) = \frac{E(P)}{d \cdot \min()} = \frac{4}{20} = 0.2$$

Exercise 4

The problem is essentially a variation of a maximum k-vertex cover problem: given a **directed** graph $G(V,E)$ where each node v_i has a **value** $s(v_i)$ for $i = 1,...,|V|$, we want to find set of nodes $S \subseteq V$ with $|S| = k$ such that the sum of the values of all the nodes covered by S is maximized.

A **greedy algorithm** that achieves a $1 - 1/e$ approximation is the following:

1. Initialize $S = \varnothing$ and $R = V$
2. Select the node $v \in R$ whose sum of the values of yet uncovered neighbors is maximized, remove v from R and mark all the neighbors of v as covered
3. Repeat step 2 until $|S| = k$

Important: when we say that a node u is a neighbor of v, we mean that exists an edge $e \in E$ from v to u and not the opposite.

Now we want to show that the proposed algorithm is a $1 - 1/e$ approximation for the problem.
First of all, we denote with x_i the total value of the **new** selected nodes after the i-th step of the algorithm, with S_i the total value of **all** the nodes selected after the i-th step (that is $S_i = \sum_{j=1}^{i} x_i$) and with $\Delta_i = OPT - S_i$ the difference between the total value of the optimal solution and the value we have reached after the i-th step.
We know that after the i-th step there is a difference of Δ_i between the final optimimum value and the value obtained by the algorithm until now, therefore there must exist a node that brings a value of at least $\Delta_i/(k-i)$ which has not been selected yet, and since the greedy algorithm always select the unselected node that brings the most value at each iteration, we have

$$x_{i+1} \geq \frac{\Delta_i}{k-i} \geq \frac{\Delta_i}{k}$$

now we can write

$$\Delta_{i+1} \leq \Delta_i - x_{i+1} \leq \Delta_i - \frac{\Delta_i}{k} = \Delta_i\left(1 - \frac{1}{k}\right) \implies \frac{\Delta_{i+1}}{\Delta_i} \leq \left(1 - \frac{1}{k}\right)$$

and then

$$\frac{\Delta_1}{\Delta_0} \cdot \frac{\Delta_2}{\Delta_1} \cdots \frac{\Delta_k}{\Delta_{k-1}} = \frac{OPT - S_k}{OPT} \leq \left(1 - \frac{1}{k}\right)^k$$

finally

$$OPT - S_k \leq \left(1 - \frac{1}{k}\right)^k OPT \leq \frac{1}{e} OPT \implies S_k \geq \left(1 - \frac{1}{e}\right) OPT$$

but S_k is exactly the solution found by the greedy algorithm, and that concludes the proof. □

Exercise 5

For the GNN implementation I choose the LastFM Asia dataset, which represents the friendships among the listeners of the LastFM radio.
I implemented three different models: a simple GNN, SAGE and GAT (details in the code), and experimented different parameters configurations. At the end of the training over 100 epochs and 100 simulations the average results were the following:

	Accuracy	Training Time
GNN	0.79	25s
SAGE	0.83	**20s**
GAT	**0.88**	60s

Clearly the worst model is the plain GNN, even though it's faster then GAT, but it's both slower and less accurate then SAGE. SAGE has the best accuracy/time ratio, but if we aim to the maximum accuracy GAT is the winner, which has also an impressive convergence speed.
I tried changing the various parameters but adding too many layers or hidden states didn't improve too much the results and simply slowed the training process.