Task1: Network Identification
Consider the network **N** represented in the file **net_x**, where **x** is your group number.
This network has been generated with one of the network models seen during the course.

You have to analyze the network **N** with the network mining tools (the ones shortlisted in the midterm project) and guess which model has been used for creating it. Your guess has to be supported by an appropriate set of experiments to confirm that networks generated with the proposed model have characteristics similar to **N** (note that you have to guess also the parameters of the model).

During the discussion of the project, you will be asked to motivate your guess. Motivations may be related to both theoretical properties of the models seen during the course (e.g., "I analyzed the provided network and I observed that its node degree distribution follows a power law. Hence, I conclude that it is not possible that the graph has been generated with a model **random(n, p)**."), and to experimental evidence (e.g., "I generated a lot of random graphs with **p = 1/3**, and none of them had similar properties as the provided network. Hence I conclude that it is improbable that the graph is **random(n, 1/3)**").

A bonus point will be assigned to all the components of the groups whose guess is closer to the model (and parameters) used to generate **N**.

Task2: Mechanisms on Weighted Social Networks with Unknown Weights

Your company is specialized in producing applications for social networks. You have been assigned the task of selling these applications with the goal of maximizing the revenue for the company. The contract between your company and the social network allows you to access to your preferred nodes over the network, and to spread whichever information you want from that node. However, the contract allows you to access to as many nodes as you want at each day. However, it does not allow for a node to receive multiple copies of the same information: whenever this happens, the social network marks the message as a spam, and it does not deliver the message to the corresponding node.

Then, your strategy is to select a subset of at most **B** nodes every day and spread from these nodes information about a network auction for selling **k** items (i.e., each day you start a different auction from a potentially different set of nodes). Clearly, the information spreading process is not deterministic: in fact, even if a node invites a neighbor into the auction, it is not for sure that the latter will participate in the auction. First, if the node receives more than one invitation, the social network will delete all of them. Moreover, even if a node receives only one invitation, then the probability that she accepts to participate depends on the strength of the relationship with the node inviting her. Unfortunately, while you know the topology of the network, you do not know the strength of these relationships.

Hence, you have to choose every day the nodes from which to start a social network auction in order to maximize the cumulative revenue obtained by all the auctions over the entire campaign. To this aim you are required at each day to both be able to reach agents with large valuation about the items you are selling (and to avoid to reaching this nodes from more than one starting nodes) and to selects a kind of auction in order to maximize the amount of money that you are able to collect from them. Notice that you can use different types of auctions in different days.

Specifically, you are given a graph **G = (V, E)**, a time horizon **T**, and a threshold **k** on the number of items to sell each day from each starting node. At each day:

- you choose a subset **S** of nodes in **V**;
- from each **s** in **S**, you send from **s** to each neighbor $\mathbf{v_s}$ to participating to the auction for selling the **k** items, by publicizing how the winner will be selected, and how the payments will be defined;
- each node **v** receiving invitation that will accept the invitation with an (unknown) probability $\mathbf{p_{sv}}$; if the invitation is accepted, the node **v** submits to the social network her personal bid $\mathbf{b_v}$ for the item, and a subset $\mathbf{S_v}$ of her neighbors that she would like to invite to the auction;
- **v** sends an invitation to all the nodes **w** in $\mathbf{S_v}$. Then the procedure is repeated as in the previous step with **w** in place of **v** and **v** in place of **s**;
- when there are no more invitations to send, the social network delivers to the seller **s** all the pairs $\mathbf{(b_v, S_v)}$ collected from nodes **v** such that **v** received only the information spread from **s**;
- each seller **s** will run the auction and declare the winners and the collected payments according to the publicized procedures.

In particular, you must implement a class **SocNetMec** that has the attributes **G**, **T**, and **k** as described above (further private attributes are allowed) and implements the following private methods:

- **__init(t)**: that takes in input the time step **t**, and returns a subset **S** of nodes of **G**, and a function **auction** (as defined in the midterm project). Note that at each time step the function **__init** may select both different nodes and different auction formats.

- **__invite(t, u, v, auction, prob, val)**, that takes in input the time step **t,** a pair of nodes **u** (the inviting node) and **v** (the invited node), an auction format (i.e., a function **auction** as defined in the midterm project), an edge probability oracle **prob** (i.e., a function that takes in input a pair of nodes **u**, **v**, and time step **t,** and returns **True** with probability $\mathbf{p_{uv}}$ and **False** with remaining probability if this is the first query about nodes **u** and **v** at time step **t**, and it returns the value previously returned for every successive query about **u** and **v** at time step t) and a valuation oracle **val** (i.e., a function that takes in input a time step **t**, and a node **v** and return the valuation of **v** for the item at the time step **t**). If **prob(u,v)** is **True**, the function outputs a bid $\mathbf{b_v}$ for node **v** and a subset of neighbor $\mathbf{S_v}$ that **v** suggests to invite. If the auction format guarantees that truthful bidding is a dominant strategy, then $\mathbf{b_v}$ is the truthful valuation (i.e., the one returned by **val(t,v)**), otherwise $\mathbf{b_v}$ is randomly chosen between the truthful valuation and **1/2 *** the truthful valuation; similarly, if the auction format guarantees that truthful reporting is a dominant strategy, then $\mathbf{S_v}$ contains all neighbors of **v**, otherwise each neighbor of **v** is inserted in $\mathbf{S_v}$ with probability 0.2. If **prob(u,v)** is **False**, then the function outputs **False**.

Moreover, it implements the following public method:

- **run(t, prob, val)**, that takes in input the time step **t**, the edge probability oracle **prob** and the valuation oracle **val**. The function invokes **init(t)** to choose the set **S** of seed nodes and the auction format. Finally, it invites nodes as described above by using the function **invite(t, u, v, auction, prob, val)** from each seed node. As soon as there is no further node to invite, the function computes for each seed node **s** the set $\mathbf{V_s}$ of nodes that have been reached only by the information sent by **s**. Then the function computes for each seed node **s**, the allocation and payments through the auction function returned by **init** when run only over the nodes in $\mathbf{V_s}$, and returns the total revenue for the seller (i.e., the sum of the payments received by each seed node).

All the implementations of the class **SocNetMec** will be run against a specific input. The one project that achieves the largest revenue, will receive a bonus point.

In particular,

- the graph **G** in input will be generated with the same network model as **net_x**;
- the diffusion probability for each edge **(u,v)** will be generated uniformly at random in the interval **[0, min(0.25, 10/max(deg(u), deg(v))]**, with **deg(x)** being the degree of node **x**;
- at each time step the valuation of node **u** for the item will be generated uniformly at random in the interval **[1, 50]**.
- the threshold **k** will be selected between **0.1%** and **0.5%** of nodes;
- the time horizon **T** will be selected between **20000** and **100000.**

Note that you do not know exactly the instance against which your implementation will be run, but you need to evaluate the performance of your implementation against multiple alternative settings generated as described above.