Final exam for T-61.5060

Wed, Dec 16, 2015
13:00–16:00

This is an **open book** exam. It is allowed to use any textbook, printed material, or personal notes brought in the room. Using any electronic device is **not** allowed.

There are **three** problems. The number of points for each problem is 25, 30, and 45, respectively.

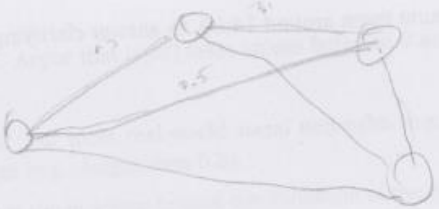
The teaching assistant will visit the exam room around 14:00, to answer clarifying questions.

Problem 1

Consider an undirected graph $G = (V, E)$. Assume that for each edge (u, v) of the graph we generate a *random* distance $d(u, v)$, drawn from the uniform distribution in the interval $(0, 1]$. For each pair of vertices u and v in the graph we consider the *shortest path distance* $d_{sp}(u, v)$ from u to v, where shortest path distance is computed over the edge distances d.

Question 1.1. Is the distance $d_{sp} : V \times V \to \mathbb{R}$ a metric?

Question 1.2. Consider now that the edge distances are drawn from the normal distribution (Gaussian), with 0 mean and standard deviation equal to 1. Is the distance $d_{sp} : V \times V \to \mathbb{R}$ is a metric?

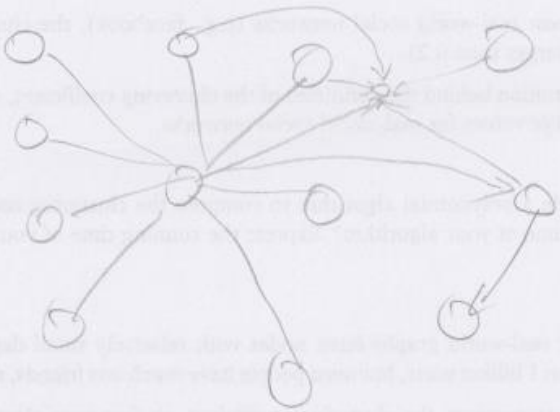


Problem 2

Consider a directed graph $G = (V, E)$, and one specific node $v^* \in V$. Our aim is to boost the PageRank score of v^* in the graph. We can do this by modifying the structure of the graph. In particular we can *add* new nodes and new edges in the graph. However, we want to do as *few* such modifications as possible.

Explain what would you do to accomplish this goal, i.e., boosting the PageRank score of v^* while peforming as few graph modifications as possible.

The answer need not be quantitative and no proofs are needed. You need however to provide a clearly-described method, and you need to explain the intuition.



Problem 3

We are given an undirected graph $G = (V, E)$, with $|V| = n$ nodes and $|E| = m$ edges. We define W to be the number of simple paths of length 2 in the graph, also known as "wedges". We define T to be the number of triangles in the graph. More formally

$$W = |\{(u, \{v, w\}) \text{ such that } u, v, w \in V \text{ and } (u, v), (u, w) \in E\}|,$$

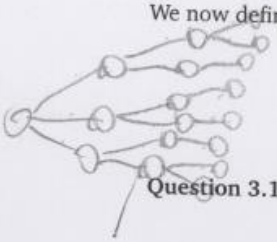
and

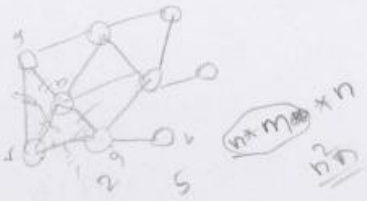
$$T = |\{\{u, v, w\} \text{ such that } u, v, w \in V \text{ and } (u, v), (u, w), (v, w) \in E\}|,$$

where the notation $(u, \{v, w\})$ is used to specify that u is in the middle of the path $v \sim u \sim w$.

We now define the *clustering coefficient* of the graph G to be

$$\text{cc}(G) = \frac{3T}{W}.$$





Question 3.1. Argue that $\text{cc}(G)$ takes values between 0 and 1.

Question 3.2. For most real-world social networks (e.g., facebook), the clustering coefficient is relatively large (e.g., larger than 0.2).

Explain what is the intuition behind the definition of the clustering coefficient, and why it is natural to expect relatively large values for real-world social networks.

Question 3.3. Provide a polynomial algorithm to compute the clustering coefficient of a graph. What is the running time of your algorithm? Express the running time of your algorithm in terms of n and/or m.

Question 3.4. Many real-world graphs have nodes with relatively small degree. (for instance, facebook has more than 1 billion users, but most people have much less friends, say, a few hundreds).

Provide an algorithm to compute the clustering coefficient, so that your algorithm is efficient for graphs in which most nodes have small degree.

What is the running time of your algorithm? Express the running time of your algorithm in terms of n and/or m and/or $\deg(v)$ (where $\deg(v)$ is the degree of a node $v \in V$).

(You may use the same algorithm that you proposed in 3.3., if you think that it works well for graphs with small-degree nodes; just express its running time in terms of the degree of the nodes).

Question 3.5. Provide an even *faster* algorithm that *approximates* the clustering coefficient (i.e., it does not need to return the *exact* answer).

What is the running time of your algorithm? Express the running time of your algorithm in terms of any variables that you think it is appropriate and convenient.

There is no need to analyze the approximation quality of your algorithm, only its running time.