Problem Set 3

Due date: Feb 23, 2018 at 9pm

1. (6 points) (Exercise 7.49 from Kleinberg & Tardos) Consider an assignment problem where we have a set of n stations that can provide service, and there is a set of k requests for service. Say, for example, that the stations are cell towers and the requests are cell phones. Each request can be served by a given set of stations. The problem so far can be represented by a bipartite graph G: one side is the stations, the other the customers, and there is an edge (x, y) between customer x and station‘y if customer x can be served from station y. Assume that each station can serve at most one customer. Using a maximum matching computation, we can decide whether or not all customers can be served, or can get an assignment of a subset of customers to stations maximizing the number of served customers.

   Here we consider a version of the problem with an additional complication: Each customer offers a different amount of money for the service. Let U be the set of customers, and assume that customer $x \in U$ is willing to pay $v_x \geq 0$ for being served. Now the goal is to find a subset $X \subset U$ maximizing $\sum_{x \in X} v_x$ such that there is an assignment of the customers in X to stations.

   Consider the following greedy approach. We process customers in order of decreasing value (breaking ties arbitrarily). When considering customer x the algorithm will either "promise" service to $x$or reject x in the following greedy fashion. Let X be the set of customers that so far have been promised service. We add x to the set Xif and only if there is a way to assign $X \cup \{x\}$ to servers, and we reject x otherwise. Note that rejected customers will not be considered later. (This is viewed as an advantage: If we need to reject a high-paying customer, at least we can tell him/her early.) However, we do not assign accepted customers to servers in a greedy fashion: we only fix the assignment after the set of accepted customers. Does this greedy approach produce an optimal set of customers? Prove that it does, or provide a counterexample.

2. (Exercise 7.27 from Kleinberg & Tardos) Some of your friends with jobs out West decide they really need some extra time each day to sit in front of their laptops, and the morning commute from Woodside to Palo Alto seems like the only option. So they decide to carpool to work.

   Unfortunately, they all hate to drive, so they want to make sure that any carpool arrangement they agree upon is fair and doesn't overload any individual with too much driving. Some sort of simple round-robin scheme is out, because none of them goes to work every day, and so the subset of them in the car varies from day to day.

   Here's one way to define *fairness*. Let the people be labeled $S = \{p_1, \ldots, p_k\}$. We say that the *total driving obligation* of p_j would have driven, had a driver been chosen uniformly at random from among the people going to work each day. More concretely, suppose the carpool plan lasts for d days, and on the i-th day a subset $S_i \subseteq S$ of the people go to work. Then the above definition of the total driving obligation Δ_j for p_j can be written as $\Delta_j = \sum_{i:p_i \in S_i} \frac{1}{|S_i|}$.

Ideally, we'd like to require that p_j drives at most Δ_j times; unfortunately, Δ_j may not be an integer.

So let's say that a *driving schedule* is a choice of a driver for each day — that is, sequence $p_{i_1}, p_{i_2}, \ldots, p_{i_d}$ with $p_{i_t} \in S_t$ — and that a *fair driving schedule* is one in which each p_j is chosen as the driver on at most $\lceil \Delta_j \rceil$ days. ($\lceil x \rceil$ denotes the smallest integer that is greater than or equal to x.)

(a) (5 points) Prove that for any sequence of sets $S_1, \cdots, S_d$, there exists a fair driving schedule.

(b) (2 points) Give an alogirithm to compute a fair driving schedule with running time polynomial in k and d.

(Hint: Most likely you should have already argued the correctness of your algorithm in part (a) of the problem. In that case you may refer to that in part (b), without repeating the argument.)

3. (Exercise 7.34 from Kleinberg & Tardos) *Ad hoc networks*, made up of low-powered wireless devices, have been proposed for situations like natural disasters in which the coordinators of a rescue effort might want to monitor conditions in a hard-to-reach area. The idea is that a large collection of these wireless devices could be dropped into such an area from an airplane and then configured into a functioning network.

Note that we're talking about (a) relatively inexpensive devices that are (b) being dropped from an airplane into (c) dangerous territory; and for the combination of reasons (a), (b), and (c), it becomes necssary to include provisions for dealing with the failure of a reasonable number of the nodes.

We'd like it to be the case that if one of the devices v detects that it is in danger of failing, it should transmit a representation of its current state to some other device in the network. Each device has a limited transmitting range — say it can communicate with other devices that lie within d meters of it. Moreover, since we don't want it to try transmitting its state to a device that has already failed, we should include some redundancy: A device v should have a set of k other devices that it can potentially contact, each within d meters of it. We'll call this a *back-up set* for device v.

(a) (4 points) Suppose you're given a set of n wireless devices, with positions represented by an (x, y) coordinate pair for each. Design an algorithm that determines whether it is possible to choose a back-up set for each device (i.e., k other devices, each within d meters), with the further property that, for some parameter b, no device appears in the back-up set of more than b other devices. The algorithm should output the back-up sets themselves, provided they can be found.

(b) (4 points) The idea that, for each pair of devices v and w, there's a strict dichotomy between being "in range" or "out of range" is a simplified abstraction. More accurately, there's a power decay function $f(\cdot)$ that specifies, for a pair of devices at distance δ, the signal strength $f(\delta)$ that they'll be able to achieve on their wireless connection. (We'll assume that $f(\delta)$ decreases with increasing δ.)

We might want to build this into our notion of back-up sets as follows: among the k devices in the back-up set of v, there should be at least one that can be reached with

very high signal strength at least one other that can be reached with moderately high signal strength, and so forth. More concretely, we have values $p_1 \geq p_2 \geq \ldots \geq p_k$, so that if the back-up set of v consists of devices at distances $d_1 \leq d_2 \leq \ldots \leq d_k$, then we should have $f(d_j) \geq p_j$ for each j.

Give an algorithm that determines whether it is possible to choose a back-up set for each device subject to this more detialed condition, still requiring that no device should appear in the back-up set of more than b other devices. Again, the algorithm should output the back-up sets themselves, provided they can be found.

4. (Exercise 7.46 from Kleinberg & Tardos) Let G be an undirected graph where the nodes represent people and there is an edge between two nodes if the two people are friends with each other. The *cohesiveness* of a group S of people is defined as the ratio $e(S)/|S|$, where $e(S)$ denotes the number of edges in S, i.e., the number of edges that have both ends in S.

   (a) (5 points) Give a polynomial-time algorithm that takes a graph G and a rational number $\alpha > 0$ and determines whether there exists a set S whose cohesiveness is at least α.

   (b) (3 points) Give a polynomial-time algorithm to find a set S of nodes with maximum cohesiveness.