Question 1 (KT 7.7). *Consider a set of mobile computing clients in a certain town who each need to be connected to one of several possible* base stations. *We'll suppose there are n clients, with the position of each client specified by its (x, y) coordinates in the plane. There are also k base stations; the position of each of these is specified by (x, y) coordinates as well.*

For each client, we wish to connect it to exactly one of the base stations. Our choice of connections is constrained in the following ways.

- *There is a range parameter r — a client can only be connected to a base station that is within distance r.*
- *There is a load parameter L — no more than L clients can be connected to any single base station.*

Your goal is to design a polynomial time algorithm for the following problem. Given the positions of a set of clients and a set of base stations, as well as the range and load parameters, decide whether every client can be connected simultaneously to a base station, subject to the range and load conditions in the previous paragraph.

> **Note:** Question 1 helps you think about solving problems using the mathematical machinery of flow networks. Notice that the problem is similar to the maximum bipartite matching problem we discussed in class: you want to match clients to base stations subject to some constraints. Hence, you will solve it by casting it into a maximum flow problem. You have two opportunities to be creative. First, you can play with network *topology* — that is the arrangement and connectivity of nodes in the network. Second, you can fiddle with edge *capacities*.

Question 2 (KT 7.5). *Decide whether you think the following statement is true or false. If it is true, give a short explanation. If it is false, give a counterexample.*

> *Let G be an arbitrary flow network, with a source s, a sink t, and a positive integer capacity c_e on every edge e; and let (A, B) be a minimum s-t cut with respect to these capacities $\{c_e \mid e \in E\}$. Now suppose we add 1 to every capacity; then (A, B) is still a minimum s-t cut with respect to these new capacities $\{1 + c_e \mid e \in E\}$.*

Question 3 (KT 7.23). *Suppose you're looking at a flow network G with source s and sink t, and you want to be able to express something like the following intuitive notion: Some nodes are clearly on the "source side" of the main bottlenecks; some nodes are clearly on the "sink side" of the main bottlenecks; and some nodes are in the middle. However, G can have many minimum cuts, so we have to be careful in how we try making this idea precise. Here's one way to divide the nodes of G into three categories of this sort.*

- *We say a node v is* upstream *if, for all minimum s-t cuts (A, B), we have $v \in A$—that is, v lies on the source side of every minimum cut.*
- *We say a node v is* downstream *if, for all minimum s-t cuts (A, B), we have $v \in B$—that is, v lies on the sink side of every minimum cut.*
- *We say a node v is* central *if it is neither upstream nor downstream; there is at least one minimum s-t cut (A', B') for which $v \in A$, and at least one minimum s-t cut (A', B') for which $v \in B'$.*

Give an algorithm that takes a flow network G and classifies each of its nodes as being upstream, downstream, or central. The running time of your algorithm should be within a constant factor of the time required to compute a single *maximum flow.* Hint: think about running a BFS on the residual graph.

Question 4 (KT 7.24). *Let $G = (V, E)$ be a directed graph with source $s \in V$, sink $t \in V$, and non-negative edge capacities $\{c_e\}$. Give a polynomial-time algorithm to decide whether G has a* unique *minimum s-t cut (*i.e., *an s-t cut of capacity strictly less than that of all other s-t cuts).* Hint: Use your result from Question 3.

Question 5 (KT 7.51 (extra credit)). *Some friends of yours have grown tired of the game "Six Degrees of Kevin Bacon" (after all, they ask, isn't it just breadth-first search?) and decide to invent a game with a little more punch, algorithmically speaking. Here's how it works.*

You start with a set X of n actresses and a set Y of n actors, and two players P_0 and P_1. Player P_0 names an actress $x_1 \in X$, player P_1 names an actor y_1 who has appeared in a movie with x_1, player P_0 names an actress x_2 who has appeared in a movie with y_1, and so on. Thus, P_0 and P_1 collectively generate a sequence $x_1, y_1, x_2, y_2, \ldots$ such that each actor/actress in the sequence has costarred with the actress/actor immediately preceding. A player $P_i (i = 0, 1)$ loses when it is P_i's turn to move, and she cannot name a member of her set who hasn't been named before.

Suppose you are given a specific pair of such sets X and Y, with complete information on who has appeared in a movie with whom. A strategy *for P_i in our setting is an algorithm that takes a current sequence $x_1, y_1, x_2, y_2, \ldots$ and generates a legal next move for P_i (assuming it's P_i's turn to move). Give a polynomial time algorithm that, given some instance of the game, decides at the start of the the game which of the two players can force a win.* Hint: think about this problem as a maximum bipartite matching problem. What happens when there is a perfect matching? What if there is not a perfect matching?