15-451 Algorithms, Spring 2009

Homework # 5 **due: Thursday April 2, 2009**

Please hand in each problem on a separate sheet and put your **name** and **recitation** (time or letter) at the top of each sheet. You will be handing each problem into a separate box, and we will then give homeworks back in recitation.

Remember: written homeworks are to be done **individually**. Group work is only for the oral-presentation assignments.

Problems:

(25 pts) 1. [Fair carpooling] The n employees of the Turing Machine Repair Company sometimes carpool to work together. Say there are m days, and S_i is the set of people that carpool together on day i. For each set, one of the people in the set must be chosen to be the driver that day. Driving is not desirable, so the people want the work of driving to be divided as fairly as possible. Your task in this problem is to give an algorithm to do this efficiently.

The fairness criterion is the following: A person p is in some of the sets. Say the sizes of the sets that p is in are $a_1, a_2, \ldots, a_k$. Person p should really have to drive $\frac{1}{a_1} + \frac{1}{a_2} + \cdots + \frac{1}{a_k}$ times, because this is the amount of resource that this person effectively uses. Of course this number may not be an integer, so let's round it up to an integer. The fairness criterion is simply that she should drive no more than this many times.

For example, say that on day 1, Alice and Bob carpool together, and on day 2, Alice, Carl, and Dilbert carpool together. Alice's fair cost would be $\lceil 1/2 + 1/3 \rceil = 1$. So Alice driving both days would not be fair. Any solution except that one is fair.

(a) Prove that there always exists a fair solution.

(b) Give a polynomial-time algorithm for computing a fair solution from the sets S_i.

Hint: Try to model the problem using network flow in such a way that part (a) falls out directly from the integrality theorem for network flow, and part (b) just follows from the fact that we can solve max flow in polynomial time. So, it all boils down to coming up with the right flow graph to model the problem.

(25 pts) 2. [Strongly connected components] A directed graph $G = (V, E)$ is semiconnected if for all pairs of vertices u and v in V, there is path from u to v, or a path from v to u (or both). Your task is to come up with an efficient algorithm for determining if a given graph is semiconnected. For each part below, give an algorithm, proof of correctness and a runtime analysis.

a) Suppose G is a directed tree (i.e. with all edges pointed down from the root). Not all directed trees are semiconnected. What does the tree have to look like? Can you test if G looks like this efficiently?

b) Now, suppose G is a directed acyclic graph (DAG). Again, not all DAGs are semiconnected? What does the DAG have to look like? Can you test if G looks like this efficiently?

c) Finally, solve the semiconnected problem for general directed graphs. Hint: It might be helpful to use your answers to part a and b.

(25 pts) 3. [Graduation] Cranberry-Melon University has n courses. In order to graduate, a student must satisfy several requirements. Each requirement is of the form "you must take at least k_i courses from subset S_i". The problem is to determine whether or not a given student can graduate. The tricky part is that any given course cannot be used towards satisfying multiple requirements. For example if one requirement states that you must take at least two courses from $\{A, B, C\}$, and a second requirement states that you must take at least two courses from $\{C, D, E\}$, then a student who had taken just $\{B, C, D\}$ would not yet be able to graduate.

Your job is to give a polynomial-time algorithm for the following problem. Given a list of requirements $r_1, r_2, \ldots, r_m$ (where each requirement r_i is of the form: "you must take at least k_i courses from set S_i"), and given a list L of courses taken by some student, determine if that student can graduate. In particular, show how you can solve this using network flow.

(25 pts) 4. [Realizing degree sequences] A social networking site called FaceBurgh claims that, for privacy reasons, for every account the only information available to the public consists of two numbers: how many people the person have ever sent messages to and how many people have ever sent messages to this person.

(a) Suppose there are 5 users that send messages only between each other, and the numbers of people they received messages from and send messages to look like the following:

$$d_{1,in} = 0, d_{1,out} = 2$$
$$d_{2,in} = 1, d_{2,out} = 2$$
$$d_{3,in} = 1, d_{3,out} = 1$$
$$d_{4,in} = 0, d_{4,out} = 1$$
$$d_{5,in} = 4, d_{5,out} = 0$$

Denoting the fact that a message was ever sent from user i to user j as a directed edge (i, j), is there a directed graph, with no multi-edges or self loops, that meets this specification? If so, what is it and is it unique?

(b) This type of specification, in which the in-degrees and out-degrees of each node are given, is called a *degree sequence*. The question above is asking whether a given degree sequence is *realizable* — that is, whether there exists a directed graph having those degrees.

Find an efficient algorithm that, given a degree sequence, will determine whether this sequence is realizable, and if so will produce a directed graph with those degrees. The graph should not have any self-loops, and should not have any multi-edges (i.e., for each directed pair (i, j) there can be at most one edge from i

to j, though it is fine if there is also an edge from j to i). Hint: as if you couldn't have guessed - think network flow!

(c) Suppose that the sequence is not realizable. Interpret the output of your algorithm, in this case.

(d) Suppose a messages (for the whole set of user accounts) were accidentally not counted in when computing the publicly available user data $d_{i,in}$, $d_{i,out}$. Extend your algorithm to find possible pairs of senders/receivers including the pairs that correspond to up to a missing messages.

(10 pts) 5. [Extra credit] Consider the following linear program for solving the max flow problem on a directed graph $G = (V, E)$.

maximize: $\sum_{e \in E} f(e)$

subject to:

$$0 \leq f(e) \leq c(e), \quad \forall e \in E$$
$$\sum_u f(u, v) = \sum_w f(v, w), \quad \forall v \in V \setminus \{s, t\}.$$

Here $f(e)$ is the flow on the edge e, $c(e)$ is the capacity of e, and s, t are the source and the sink vertices respectively. Show that the linear program will find the maximum flow in the graph or construct a counter example.