# TTIC 31010 / CMSC 37000 Algorithms, Winter Quarter 2019

**Homework # 3** **Due: February 14, 2019**

---

Homeworks due in class on the due date. Please put your name on each page of your handin just in case pages get separated.

Lateness policy: up to 24 hours late: 10 points off. 24-48 hours late: 20 points off. More than 48 hours late: 60 points off (at this point, solutions will be posted - and you may look at them if you wish, but your answers should be in your own words).

Written homeworks are to be done *individually*. Group work is only for the oral-presentation assignments. If you have questions, please contact the course staff.

Please do *not* post questions or solutions on websites such as coursehero etc.

---

**Problems:** (Hint: all involve network flow)

(30 pts) 1. **Graduation.** The University of Bureaucracy 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 hard 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 $m$ requirements, where requirement $i$ 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. Prove that your algorithm is correct; that is, it states that a student can graduate if and only if the student indeed can graduate.

(35 pts) 2. **Fair cooking.** $n$ students share a house together. Each evening $i$, some subset $S_i$ of the students gather for dinner (the remaining $n - |S_i|$ students are still in the lab), and one of the students in $S_i$ is selected as the cook for that evening. The students want to select the cook for each evening in such a way that nobody has to cook much more than their "fair share". Specifically, the fairness criterion is the following: Say that person $p$ is in some $k$ of the sets, which have sizes $n_1, n_2, \ldots, n_k$, respectively. Person $p$ should really have to cook $\frac{1}{n_1} + \frac{1}{n_2} + \cdots + \frac{1}{n_k}$ 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 person $p$ should have to cook no more than this many times.

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

(a) Prove that there always exists a fair solution. Specifically, given sets $S_1, S_2, \ldots, S_m$, there always exists a selection of cooks for each day that satisfies the above fairness criterion.

(b) Give a polynomial-time algorithm for computing a fair solution from the sets $S_1, \ldots, S_m$.

(35 pts) 3. **Degree sequences.** You are the chief engineer for Graphs-R-Us, a company that makes graphs to meet all sorts of specifications.

(a) A client comes in and says he needs a 4-node directed graph in which the nodes have the following in-degrees and out-degrees:

$$\begin{array}{llll} d_{1,in} = 0, & d_{2,in} = 1, & d_{3,in} = 2, & d_{4,in} = 3 \\ d_{1,out} = 2, & d_{2,out} = 2, & d_{3,out} = 1, & d_{4,out} = 1 \end{array}$$

Is there a directed graph, with no multi-edges or self loops, that meets this specification? If so, what is it?

(b) What about a 3-node graph (again with no multi-edges or self loops) with these in-degrees and out-degrees?

$$\begin{array}{lll} d_{1,in} = 2, & d_{2,in} = 2, & d_{3,in} = 1 \\ d_{1,out} = 2, & d_{2,out} = 2, & d_{3,out} = 1 \end{array}$$

(c) 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 stated at the top of the assignment, think network flow.