600.363 Introduction to Algorithms / 600.463 Algorithms I **Fall 2014**
Homework #9 **Due:** November 18, 2014, 1:30pm

Please start each problem on a new page, and include your name on each problem. You can submit on blackboard, under student assessment.

Remember: you may work in groups of up to three people, but must write up your solution entirely on your own. Collaboration is limited to discussing the problems – you may not look at, compare, reuse, etc. any text from anyone else in the class. Please include your list of collaborators on the first page of your submission. You may use the internet to look up formulas, definitions, etc., but may not simply look up the answers online.

Please include proofs with all of your answers, unless stated otherwise.

1 Graduation Requirements Revisited (33 points)

John Hopskins has switched to a more lenient policy for graduation requirements than it had in Homework 8. As in the previous homework, there is 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". However, under the new policy a student *may* use the same course to fulfill multiple requirements. For example, if there was a requirement that a student must take at least one course from $\{A, B, C\}$, and another required at least one course from $\{C, D, E\}$, and a third required at least one course from $\{A, F, G\}$, then a student would only have to take A and C to graduate.

Now consider an incoming freshman interested in finding the *minimum* number of courses required to graduate.

(a) Prove that the problem faced by this freshman is NP-complete, even if each k_i is equal to 1. More formally, consider the following decision problem: given n items (say $a_1, \ldots a_n$), given m subsets of these items $S_1, S_2, \ldots, S_m$, and given an integer k, does there exist a set S of at most k items such that $|S \cap S_i| \geq 1$ for all $i \in \{1, \ldots, m\}$. Prove that this is NP-complete, i.e. that it is in NP and is NP-hard.

(b) Show how you could use a polynomial-time algorithm for the above decision problem to also solve the search version of the problem (i.e., actually find a minimum-sized set of courses to take).

(c) We could define a *fractional* version of the graduation problem by imagining that in each course taken, a student gets a score between 0 and 1. Then requirement r_i is changed to state "the sum of your scores in courses taken from S_i must be at least k_i" (courses not taken count as 0). The student now wants to know the least total work needed to graduate, defined as the minimum sum of all scored needed to satisfy all of the requirements.

Show how this problem can be solved in polynomial time by using linear programming. Be sure to specify what the variables are, what the constraints are, and what the objective function is.

2 Lecture Planning (33 points)

You have been asked to organize a freshman-level seminar that will meet once a week during the next seminar. The first ℓ weeks of the seminar is devoted to guest lectures: there are n options for guest speakers overall, and on week i (for $i \in \{1, 2, \ldots, \ell\}$) a subset L_i of the speakers are available to give a lecture.

In the weeks after the lectures have finished the students are required to complete individual projects. There are p projects that each student has to do, and each project requires background material from certain lectures. In particular, for each project j (for $j \in \{1, 2, \ldots, p\}$) there is a subset P_j of relevant guest speakers, and in order for students to complete project j there must have been *at least one* speaker from P_j among the ℓ guest speakers scheduled.

So we have the following problem: given these sets, can we select exactly one speaker for each of the ℓ weeks such that the speaker chosen for each week is available in that week, and for every project j at least one of the relevant speakers gave a talk? We call this the *Lecture Planning Problem*.

To make this clear, let's see an example. Suppose that $\ell = 2$, $p = 3$, and there are $n = 4$ speakers that we denote A, B, C, D. The availability of the speakers is given by the sets $L_1 = \{A, B, C\}$ and $L_2 = \{A, D\}$. The relevant speakers for each project are given by the sets $P_1 = \{B, C\}$, $P_2 = \{A, B, D\}$, and $P_3 = \{C, D\}$. Then the answer to this instance is yes, since we can choose speaker B in the first week and speaker D in the second week; this way, for each of the three projects, students will have seen at least one of the relevant speakers.

Prove that the Lecture Planning Problem is NP-complete. Hint: think of reducing from 3-SAT, and creating two speakers per variable.

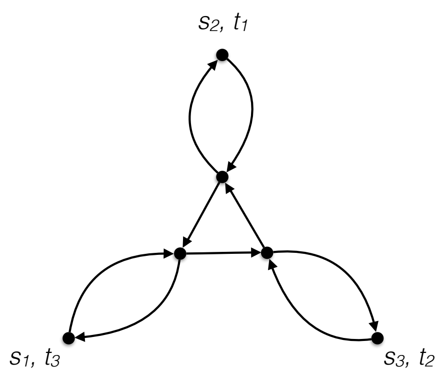
3 More Flows and Cuts (33 points)

We've spent a lot of time talking about $s-t$ flow. Here's a natural generalization: instead of a single source and single sink, we are given k source-sink pairs $(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$. These pairs are called the *commodities* – think, for example, of the graph as a road network and we need to ship oil from s_1 to t_1, packages from s_2 to t_2, nuclear waste from s_3 to t_3, etc. So the commodities all send flow along the same network, but are distinct flows. This is formalized by the *multicommodity flow* problem, in which we are given a directed graph $G = (V, E)$, edge capacities $c : E \to \mathbb{R}^+$, and a collection of k source-sink pairs (commodities). The objective is to ship as much total flow as possible, while 1) each commodity is itself a valid flow (obeys flow conservation everywhere but at the source and sink for that commodity), and 2) the total flow (summed over all commodities) across each edge is at most the capacity of the edge.

To say this another way, we are essentially asking for k flows, where the ith flow is from s_i to t_i and obeys flow conservation. And when we sum the flows together, edge capacities are still not violated. It is important to note that our objective is to ship as much flow as possible in total, not to be "fair" in any way between the commodities. It is also important to note that the commodities interfere: if we ship 2 units of $s_1 - t_1$ flow and 3 units of $s_2 - t_2$ flow across edge e, then the total flow across edge e is 5 (so the capacity of e needs to be at least 5).

(a) Show how to use linear programming to solve the multicommodity flow problem. Be sure to specify what the variables are, what the constraints are, and what the objective function is.

The natural dual to max-flow was min-cut, and we saw that the maximum flow always equaled the minimum cut. It turns out that the natural dual to multicommodity flow is a problem known as *Multicut*. A multicut is a set of edges E' such that in $G - E'$ there is no path from s_i to t_i for any $i \in \{1, 2, \ldots, k\}$, i.e., all source-sink pairs that we are given are disconnected when we remove E'. Then like with min-cut and $s - t$ flow, it is obvious that there is no multicommodity flow larger than the minimum multicut – any multicut is an upper bound on the possible multicommodity flow. But in this setting there is a gap – the max multicommodity flow is *not* always equal to the min multicut. Consider the following example, in which all edge capacities are 1:



(b) Prove that the maximum multicommodity flow has value at most 3/2.

(c) Prove that the minimum multicut has capacity 2.