# 15-451 — Algorithms — Spring 2006

Sleator, Golovin, Kissner

**Homework #7**

**Due: May 4, 2006, start of class.**

**Some Reminders:**

- You may discuss these problems with others, in small groups. However we strongly recommend that you think for a while about them yourself before starting such discussions.
- The work that you turn in must be your own, written by you in your own words. We are allowing handwritten solutions, although typeset ones are preferred. If you handwrite, WRITE CLEARLY, or we will revert to the old system of requiring you to typeset solutions.
- The cover page of your submission must clearly display the assignment number, your name, your recitation section and your Andrew ID.

**Problems:**

Note: When attempting to show a problem is NP-complete by reducing a known NP-complete problem to your problem, you may use only those NP-complete problems covered in lecture or in the textbook.

## 1 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 $k$ courses from subset $S$". 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.

## 2 Graduation, Part 2

Cranberry-Melon University has switched to a less draconian policy for graduation requirements than that used in the previous problem. As before, 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, unlike the case before, a student *may* use the same course to fulfill several requirements. For example, if one requirement stated that a student must take at least one course from $\{A, B, C\}$, 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 that he (or she) needs to take in order to graduate.

(a) Prove that the problem faced by this freshman is NP-hard, even if each $k_i$ is equal to 1. Specifically, consider the following decision problem: given $n$ items labeled $1, 2, \ldots, n$, 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 $|S \bigcap S_i| \geq 1$ for all $S_i$. Prove that this problem is NP-complete (don't forget to prove it is in NP).

(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 grade between 0.00 and 1.00, and that requirement $r_i$ now states "the sum of your grades in courses taken from set $S_i$ 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 the minimum sum of all grades needed to satisfy all the requirements.

Show how this problem can be solved using *linear programming*. Be sure to specify what the variables are, what the constraints are, and what you are trying to minimize or maximize.

# 3 Building the Bomb

It is 1944, and you are a physicist working on the Manhattan project, attempting to build an atomic bomb. In the process, you will be enriching a lot of uranium. The thing about this enriched uranium is, if too much of it gets close together, it will explode, making you very unhappy. Naturally, you will take precautions to make sure you don't get blown up, though ideally you'd like to do it as inexpensively as possible...

(a) The first approach is to take your storage facilities as fixed, and place the uranium in locations that are sufficiently spaced out. To make this concrete, we will model the storage facility as an undirected graph $G = (V, E)$, where $V$ is the set of locations at which we can store uranium. The edges have weights $w(e)$, which induce a distance metric on the graph, in which $d(u, v)$ is the cost of the minimum weight path from $u$ to $v$. You are also given a parameter $D$. Putting uranium at two locations $u$ and $v$ such that $d(u, v) < D$ will cause an explosion. However you'd like to place as many units of uranium as possible down, subject to the constraint of not being blown up. Derive a polynomial time algorithm to maximize the number of units of uranium you can place (the output is the set of locations where you'd put it), or show that this problem is NP-hard.

(b) Another way to control the nuclear chain reaction is with control rods that stop the cascading neutrons. So you can imagine another undirected graph $G = (V, E)$ where you plan to place uranium at every location in $V$. This is safe if and only if you place control rods down at a set of locations $X \subseteq V$ such that each $v \in V$ is either in $X$ or has a neighbor in $X$. Your goal is the make the site safe while minimizing the size of $X$. Derive a polynomial time algorithm to minimize $|X|$, or show that this problem is NP-hard.

(c) A third way to deal with the uranium problem is to reinforce the walls with material that stops neutrons. Here you have an undirected graph $G = (V, E)$ with edge weights $w$ and with a set $Y \subseteq V$ of locations where you plan to put the uranium. Each edge $e$ represents a barrier to neutrons that can be reinforced any fractional amount $x(e)$ at a cost of $w(e)x(e)$. You must reinforce the edges of the graph such that for any two vertices $u, v \in Y$, the shortest path from $u$ to $v$ under lengths $x(e)$ (i.e. edge $e$ has length $x(e)$). is at least one. Your goal is to minimize the total cost you pay. Derive a polynomial time algorithm to minimize the cost, or show that this problem is NP-hard.