1 Integer Linear Programming (33 points)

In class we talked about linear programming, and the fact that it can be solved in polynomial time. Slightly more formally, we defined the feasibility version of linear programming to be the following decision problem

- Input: $n$ variables $x_1, x_2, \ldots, x_n$, and $m$ linear inequalities over the variables.
- Output: YES if there is a way of assigning each variable a value in $\mathbb{R}$ so that all $m$ linear constraints are satisfied, NO otherwise.

Let Integer Linear Programming be the same problem, but where each variable is only allowed to take variables in $\mathbb{Z}$ rather than in $\mathbb{R}$. Prove that Integer Linear Programming is NP-complete, i.e. it is in NP and is also NP-hard.

2 Graduation Requirements Revisited (34 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. 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. Hint: consider a reduction from the Vertex Cover problem.

3 Bin Packing (33 points)

Suppose that we are given a set of $n$ objects, where the size $s_i$ of the $i$th object satisfies $0 < s_i < 1$. We wish to pack all the objects into the minimum number of unit-size bins. Each bin can hold any subset of the items whose total size does not exceed 1. Let $S = \sum_{i=1}^{n} s_i$. 


(a) Prove that the optimal number of bins required is at least $\lceil S \rceil$.

The *first-fit algorithm* considers each object in turn (from 1 to $n$) and places it in the first bin that can accommodate it. If there is no such bin, then we create a new bin for it and make it the last bin. Note that this defines an ordering over bins based on when we created them, so “first” and “last” make sense.

(b) Prove that the first-fit algorithm leaves at most 1 bin at most half full. In other words, all bins but 1 are more than half full.

(c) Prove that the first-fit algorithm is a 2-approximation to bin packing.