Fall 2017 Approximation Algorithms

# Midterm Exam

*Due October 10*

**Instructions:** To solve these problems, you are allowed to consult your class textbook (*The Design of Approximation Algorithms* by Williamson and Shmoys) and notes, but no other sources. *This exam is individual work; you should not talk about it with anyone except the course instructor.* Please write clearly and concisely.

**(1) [10 points]** You are a crafty spy in charge of a spy agency, and are choosing a team for a long-term mission. You are given a universe set $U$ of the spies that you can choose for your mission, with $|U| = n$. You are also given a set of $m$ *tasks* that your spies will have to perform on your mission. Each task $i$ can only be performed by either set $S_i \subseteq U$, or a set $T_i \subseteq U$. Therefore, for every $i = 1 \ldots m$ you must choose either $S_i$ or $T_i$ to include in your team. Your goal is to minimize the total size of your team, i.e., the union of all the sets chosen.

Give a poly-time 2-approximation algorithm for this problem. *Hint: Use LP-rounding.*

**(2) [10 points]** Consider the following problem.

Dominating Set: We are given an undirected graph $G = (V, E)$, as well as weights $w(v)$ for each $v \in V$. For each vertex $v \in V$, define $N(v) = \{w \in V | (v, w) \in E\} \cup \{v\}$. In other words, $N(v)$ is the set of all neighbors of $v$, plus the node $v$ itself. A *dominating set* $C \subseteq V$ is a set of vertices such that for every vertex $v \in V$, either a neighbor of $v$ is in $C$, or $v$ itself is in $C$. The goal is to find a minimum-weight dominating set.

(a) Give an LP-relaxation and its dual for the Dominating Set problem. Your primal linear program should have a variable $x_v$ for every $v \in V$; those are the only variables you should need for that LP.

(b) Let $\Delta$ be the maximum degree of $G$ (assume $\Delta \geq 1$). Give a primal-dual approximation algorithm for Dominating Set with an approximation ratio of $\Delta$.
*Hint:* This algorithm should add only a single new node to $C$ in every iteration. Begin by forming a $(\Delta + 1)$-approximation algorithm, and then improve it to a $\Delta$-approximation.