Approximation Algorithms **Homework 2**
Date: Sept 21, 2005 **Due:** Sept 28, 2005

Note: In all these problems: if we ask you to show an approximation ratio of ρ, you should tell us about the best approximation ratio you did find, be it better or worse than ρ.

1. **Multiway Cut.** Recall the MULTIWAY CUT problem: given a graph $G = (V, E)$ with each edge e having a *capacity* c_e, and a subset $S = \{s_1, s_2, \ldots, s_k\} \subseteq V$ of *terminals*, find a set of edges $E_{cut} \subseteq E$ of minimum capacity such that each connected component of $G \setminus E_{cut}$ contains at most one terminal.

   We saw a greedy algorithm in class that gave a $2(1 - \frac{1}{k})$-approximation for the problem. Here's another "greedier" approximation algorithm that Mohit proposed in class:

   > Find a s_i-s_j min-cut whose capacity is smallest amongst all s-s' min-cuts with $s, s' \in S$. Delete all edges in this cut. Recurse on the resulting components.

   Show that this algorithm is also a $2(1 - \frac{1}{k})$-approximation for the problem.

2. **Hitting Set.** Given a set U, and a family $\mathcal{F} = \{S_1, S_2, \ldots, S_k\}$ of subsets of U, a *hitting set* for $\mathcal{F}$ is a subset $H \subseteq U$ such that $H \cap S_i \neq \emptyset$ for all $1 \leq i \leq k$. The (MINIMUM) HITTING SET problem seeks to find a hitting set of the smallest cardinality.

   Show that the SET COVER problem discussed in class has a ρ-approximation if and only if the HITTING SET problem has a ρ-approximation.

3. **Better Makespan Minimization.** In class, we saw that Graham's List Scheduling algorithm gave a 2-approximation for the problem of scheduling n jobs on m parallel machines to minimize the makespan. Construct an instance on which this algorithm does no better than a 2-approximation.

   Chris had proposed another "greedy" algorithm: sort all the jobs in non-increasing order of sizes, and run List Scheduling with this order of jobs. Show that this gives a 1.5-approximation for the problem.

4. **Steiner Tree.** An instance of the MINIMUM COST STEINER TREE problem consists of a graph $G = (V, E)$ with positive costs c_e for each edge $e \in E$, and a subset $R \subseteq V$ of *required vertices* or *terminals*. The goal is to find a minimum cost set of edges $E_{ST} \subseteq E$ that *spans* the set R; i.e., any pair of terminals in R has a connecting path in E_{ST}. Let $|R| = k$.

   Since the edge set E_{ST} has minimum cost, it will have no cycles and hence be a tree: this is called a minimum cost *Steiner* tree on R.

   (a) Let $\widehat{G} = (V, \binom{V}{2})$ be a complete graph on the same set of nodes V: set the cost $\widehat{c}_e$ of edge $e = \{u, v\}$ to be the shortest-path distance according to the edge "lengths" c_e between u and v in G. ($\widehat{G}$ is called the *metric completion* of G.)

   For any set R of terminals, show that T^*, the optimal Steiner tree on R in G has the same cost as $\widehat{T}^*$, the optimal Steiner tree on R in the metric completion $\widehat{G}$.

(b) Consider the subgraph $\widehat{G}[R]$ be the graph induced by the set of terminals in the graph $\widehat{G}$, and let $\widehat{T}$ be the minimum spanning tree in $\widehat{G}[R]$ according to the edge costs $\widehat{c}_e$. Show that $\widehat{c}(\widehat{T}) \doteq \sum_{e \in \widehat{T}} \widehat{c}_e$, the cost of $\widehat{T}$, is at most $2(1 - \frac{1}{k}) \sum_{e \in \widehat{T}^*} \widehat{c}_e$.

(c) Show how to use the tree $\widehat{T}$ to find a tree T in G with cost $c(T) = \sum_{e \in T} c_e$ at most $\widehat{c}(\widehat{T})$. Infer that the algorithm you have developed is a $2(1 - \frac{1}{k})$-approximation to the minimum cost Steiner tree problem.