HW 10 (due Tuesday, at noon, April 29, 2014)

CS 473: Fundamental Algorithms, Spring 2014 Version: **1.02**

Make sure that you write the solutions for the problems on separate sheets of paper. Write your name and netid on each sheet.

Collaboration Policy: The homework can be worked in groups of up to 3 students each.

---

1. (30 PTS.) Steiner tree

Given an undirected graph $G = (V, E)$ and subset of nodes $S \subseteq V$ called terminals, a Steiner tree for S in G is a tree $T = (V', E')$ such that T is a subgraph of G and T' contains all the terminals. A node $v \in V' \setminus S$ is said to be a Steiner node. Consider the following decision problem: given a graph $G = (V, E)$, a set of terminals $S \subseteq V$, and an integer k, is there a Steiner tree for S in G that contains at most k Steiner nodes? Prove that this problem is NP-complete. *Hint:* Use a reduction from the Set Cover problem.

2. (30 PTS.) Finding Kites in graphs

A kite is a graph on an even number of nodes, say $2n$, in which n of the nodes form a clique and the remaining n vertices are connected in a "tail" that consists of a path joined to one of the nodes in the clique. Given a graph G and an integer k, the KITE problem asks for a subgraph which is a kite that contains $2k$ nodes. Prove that KITE is NP-Complete.

3. (40 PTS.) Max 2-SAT

(A) (2 PTS.) Consider two boolean variables x and y. Write a 2CNF formula that computes the function $\neg(x \wedge y)$.

(B) (17 PTS.) Given a connected graph $G = (V, E)$ with n vertices and m edges, we want to compute its maximum size independent set. To this end, we define a boolean variable for every vertex of V. Describe how to write a 2CNF formula that is true if and only if the vertices that are assigned value 1 are all independent.

(C) (10 PTS.) Describe how to compute a 2CNF formula from G such that there is an assignment that satisfies at least Δ clauses of this formula if and only if there is an independent set in G of size k (or larger). To make things easy, you are allowed to duplicate the same clause in your formula as many times as you want. Naturally, the algorithm for computing this formula from G should work in polynomial time (and of course, you need to describe this algorithm). (The final formula size has to be polynomial in n and m.) What is the value of Δ as a function of n and m?

(D) (10 PTS.) Using the above, prove that MAX 2SAT (i.e., given a 2CNF formula, compute the assignment that maximizes the number of clauses that are satisfied) is an NP-HARD *optimization* problem. That is, show that if one can solve MAX 2SAT in polynomial time then one can solve 3SAT in polynomial time.

(E) (1 PTS.) As you know, one can solve 2SAT in linear time. Why does this does not imply that MAX 2SAT can be solved in polynomial time?