Special Topics in Complexity Theory, Spring 2006, Yap
Homework 2
Due: Wed Mar 8.

INSTRUCTIONS

- REMEMBER OUR HOMEWORK RULES: you prepare a solution in writing, and then we will have appointments in which you need to give an oral defence of your solution. I will grade BOTH the oral defence AND your written solution.

---

1. (10 Points)
Show that $\sqrt{x}$ is computable.
Note: $\sqrt{x}$ is defined iff $x \geq 0$, and $\sqrt{x} \geq 0$ when defined. Recall how we define the computability of a partial real function f: on inputs that are not in the domain of f, the OTM must loop.

2. (15 Points)
Let $f : [0,1] \to \mathbb{R}$ be a computable total function. Define $\max(f) = \max\{f(x) : x \in [0,1]\}$.
(a) Show that $\max(f)$ is a computable real.
(b) Can this be extended to the case where f is partial?

3. (15 Points)
We know that all computable functions are continuous. Show that the converse is false.

4. (15 Points)
Prove Theorem 2.22 (p.59) in the text book. This concerns a sequence of piecewise linear functions f_n that approximate a function $f \in C[0,1]$ computable in polynomial time.

5. (30 Points)
In class we discussed the Polynomial Time Hierarchy $\{\Sigma_i^P, \Delta_i^P, \Pi_i^P : i \geq 1\}$. whose initial level is just $\Delta_1^P = P$, $\Sigma_1^P = NP$ and $\Pi_1^P =$ co-NP. Give upper bounds on the complexity of the following languages by placing them in appropriate classes of the Polynomial Time Hierarchy.
(i) Consider finite undirected (unlabeled) graphs. If G is such a graph on n vertices, let $\overline{G} \in \{0,1\}^*$ denote the obvious representation of Boolean matrix that in turn represents G. Thus $|\overline{G}| = n^2$. Clearly, each G has up to $n!$ representations (by permuting the vertices, we get different matrices). Consider the following language:

$$L_1 = \{\overline{G} : \ \overline{G} \text{ is the lexicographically smallest representation of } G\}.$$

(ii) Consider Boolean functions which are represented by Boolean circuits

C. Each C is given by a triple $(G, n, g_1, g_2, \ldots, g_m)$ where
(1) G is a DAG (directed acyclic graph) on the vertex set $V = \{1, \ldots, n+m\}$, such that vertex $n+m$ is a sink of the DAG, and the set of sources is the set $\{1, 2, \ldots, n\}$;
(2) Each of the g_i's indicates a type of Boolean gate: either $\neg$ (NOT-gate), $\vee$ (OR-gate), or $\wedge$ (AND-gate). We can encoded three gate-types by the integers $0, 1, 2$. The non-source node $n+i \in \{n+1, \ldots, n+m\}$ is associated with the gate-type g_i. The in-degree of a NOT-gate must be 1, and the in-degree of the other types is at least 1.

The circuit $C = (G, n, g_1, \ldots, g_m)$ computes a Boolean function $f_C : \{0,1\}^n \to \{0,1\}$ where we view vertex $n+m$ as the output. The **size** of C is m. Let $\overline{C}$ denote some representation of C. We say $\overline{C}$ and $\overline{C'}$ are equivalent if C and C' computes the same Boolean function. E.g., G can be represented by the Boolean matrix as in part (i). It is can be checked in polynomial time whether $\overline{C}$ is well-formed (i.e., represents a Boolean function as specified above). Consider the language

$$L_2 = \{\overline{C} : \ C \text{ has minimum size among equivalent circuits}\}.$$