Theory of Computation
Depth Exam
Fall 1991

Directions: Answer any FOUR of the seven questions below.

1. The input is a sequence of numbers $x_1, x_2, \ldots, x_n$. Prove that any comparison based deterministic algorithm that selects a number from this set that is in the smaller half (i.e., less than or equal to the median) must make at least $\lfloor \frac{n}{2} \rfloor$ comparisons.

2. For this problem you are to consider comparison-based sorting algorithms. At each point in the algorithm's execution, a subset of the $n!$ possible orderings of the data will be consistent with the comparisons so far performed. Call such a sorting algorithm *wise* if at each step, a comparison is made that eliminates a constant fraction of the possible orderings.

   (i) Can any $\Theta(n^2)$ sorting algorithm be wise?

   (ii) Exhibit a wise sorting algorithm that runs in time $O(n \log n)$.

3. A typical load balancing problem that arises in scheduling jobs on a parallel computer is the following. k jobs must be scheduled on p processors. The number of steps that the jobs take are $h_1, h_2, ..., h_k$. Assume for simplicity that there is no cost in scheduling, preempting or rescheduling a job. Also, all jobs are ready to be scheduled initially (none depend on others).

   Suppose first that the number of steps the jobs take are unknown to the scheduler. Show that the parallel time to complete all jobs on the p processors is at least $max\{n/p, h\}$ where $n = \sum_{i=1}^{k} h_i$ and $h = max_i\{h_i\}$. Show how to schedule the jobs preemptively so that they are completed in parallel time $n/p + h$. Next suppose that the h_i are known. Show how to schedule the jobs nonpreemptively so that they are completed in parallel time $n/p + h$.

4. The *k-vertex cover problem* is: given a graph G, does G have a vertex cover of size at most k? This is a restriction of the general vertex cover problem, where k is now considered a fixed constant. A vertex cover of a graph is a set S of vertices which "covers" the edges of the graph, that is, for every edge of the graph, one of its endpoints is in S. Give a linear time algorithm for the k-vertex cover problem. (The constants in the running time may be functions of the constant k).

   Why does this not contradict the fact that the general vertex cover problem is NP-Complete?

5. A tree is a *computation tree* of an alternating Turing machine (ATM) M on a string w if its nodes are labeled with configurations of M on w such that the descendents of any internal node labeled by a universal (existential) configuration include all (resp. one) of the successors of that configuration. Such a tree is *accepting* if the root is labeled with the initial configuration and all leaves are accepting configurations.

   A language L is accepted within *tree-size* bound $z(n)$ if for every string $w \in L$ there is at least one accepting computation tree for M on w of size (number of nodes) $\leq z(n)$, where n is the length of w. Further, for $w \notin L$, there is no accepting computation tree. Similarly, L is accepted within simultaneous tree-size and space bounds $z(n)$ and $s(n)$ if for every $w \in L$, there is an accepting computation tree with $\leq z(n)$ nodes, each of which is labeled by a configuration using space $\leq s(n)$.

   Suppose that L is accepted by an ATM with simultaneous tree-size and space bounds $z(n)$ and $s(n)$. Show that L is also accepted by the following nondeterministic Turing machine. The Turing machine has two types of storage: regular worktape storage, which is limited to $O(s(n))$ space, and a stack, which is not limited in its space. In addition, the NTM must run in time $O(z(n))$ on inputs in the language.

6. Let $USAT$ be the set of Boolean formulas which have a unique satisfying assignment. Show that $USAT$ is co-NP hard. Prove that if $USAT$ is NP-complete then the polynomial time hierarchy collapses.

7. A tree automaton is a generalization of a deterministic finite state automaton, where the inputs are trees, rather than strings. More precisely, the input is a finite, oriented binary tree such that every node has either 0 or 2 ordered children (i.e. one is the left and one is the right child). The internal nodes of the tree are labeled with symbols from a finite alphabet Σ and the leaves are labeled with the empty string.

   A *tree automaton* is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where, just as for standard finite state automata, Q is a set of states, $F \subseteq Q$ is the set of accepting states, q_0 is the initial state, Σ is the input alphabet, in this case, the alphabet of symbols labeling the tree nodes, and δ is the transition function, $\delta : Q \times Q \times \Sigma \rightarrow Q$.

   The transition function associates a state with each node of a tree, inductively as follows. The state associated with the leaves is defined to be the initial state. Let n be an internal node labeled with symbol a and let the states associated with n's left and right children be q_1 and q_2, respectively. Then the state associated with n is $\delta(q_1, q_2, a)$. A tree is *accepted* by the automaton if the state associated with the root is an accepting state.

   For example, the following tree is accepted by the automaton $(\{q_0, q_1\}, \{a, b\}, \delta, q_0, \{q_0\})$ where δ is defined so that $\delta(q_0, q_0, b) = q_1$ and $\delta(q_1, q_0, a) = q_0$.

   (i) State and prove a "pumping lemma" for tree automata that generalizes the standard pumping lemma for finite state automata. Your lemma should describe how, given a fixed tree automaton, a "sufficiently large" tree accepted by the automaton can be used to define an infinite family of trees that are all accepted by the tree automaton.

   (ii) Exhibit a family of trees that is not accepted by any tree automaton (i.e. is "non-regular" with respect to tree automata). Use your pumping lemma to justify your choice.