Theory Qualifier
June 1, 1988

Instructions

Four hours, closed book. All questions are equal weight.

There are two parts, a *basic part* and an *advanced part*, each consisting of four questions. Of the four advanced questions, there is one from each of the following three areas:

1. recursive function theory and logic,
2. algorithm design and analysis,
3. computational complexity.

In addition, there is one "wild card" question. For a *basic pass*, you must answer all four basic questions satisfactorily. For an *advanced pass*, you must answer all four basic questions and any two of the four advanced questions satisfactorily. Indicate clearly which two advanced questions you are attempting; only two will be graded.

The questions will be graded according to the following criteria, in order of importance:

1. correctness and completeness
2. clarity, precision, and conciseness
3. optimality of result

Partial credit will be awarded where appropriate. Unclear presentations and irrelevancies will be penalized. *Please read all questions carefully before answering.*

GOOD LUCK!

Part I—Basic

1. Let $G = (V, E)$ be an directed acyclic graph such that for all $u, v \in V$, either there is a path from u to v or there is a path from v to u in G. Prove that there is a vertex u such that for all $v \in V$, there is a path from u to v in G. Give an $O(|E|)$ algorithm for finding such a vertex.

2. (a) Prove that the following problem is coNP-complete. You may assume Cook's theorem.

   Instance: A Boolean formula φ in disjunctive normal form.

   Question: Is φ a tautology? I.e., do all truth assignments satisfy φ?

   (b) Suppose that the statement is modified as follows:

   Instance: A Boolean formula φ in disjunctive normal form in which each disjunct has at most k literals.

   What is the minimum k for which the problem remains coNP-complete? Give a polynomial-time algorithm for smaller k.

3. Say which of the following connectives correctly relates the pairs of language classes (a–e) given below:

   (i) $=$

   (ii) $\subseteq$, but not known to be equal

   (iii) $\subset$ (proper inclusion—included in, but not equal)

   (iv) not known to be related

   (v) unrelated

   (a) CFL's, P

   (b) regular languages, languages accepted by two-way nondeterministic finite automata

   (c) recursive languages, context-sensitive languages

   (d) $\bigcup_{k \geq 0} DSPACE(n^k)$, $\bigcup_{k \geq 0} NSPACE(n^k)$

   (e) $NLOGSPACE$, co$NLOGSPACE$ ($NLOGSPACE =$ nondeterministic logspace)

4. Prove that $\{a^i \mid i \text{ is a prime}\}$ is not a context-free language.

Part II—Advanced

1. (recursive function theory and logic) Let $D_0, D_1, \ldots$ be a standard enumeration of polynomial-time Turing machines. Let

$$MinP = \{i \mid \forall j \; L(D_i) = L(D_j) \rightarrow i \leq j\} \; .$$

   Prove that:

   (a) *MinP* is recursively enumerable;

   (b) *MinP* is not recursive.

2. (algorithm design and analysis) The problem of checking whether a given directed graph has a Hamiltonian cycle is *NP*-complete. Consider the following relaxation of the problem: does the given directed graph contain a set of disjoint cycles which cover every vertex exactly once? Give a polynomial-time algorithm for this problem.

3. (computational complexity) Suppose you are given an oracle that tells the number of satisfying truth assignments to a Boolean formula. Suppose further that you are given a source of randomness which, when queried with a positive integer m, returns a random integer from the set $\{0, 1, \ldots, m-1\}$, all m possibilities equally likely. Using these constructs, devise a polynomial-time algorithm which, given a Boolean formula φ, either responds "unsatisfiable" if φ is unsatisfiable, or outputs a random satisfying truth assignment. That is, if φ has $k \geq 1$ satisfying truth assignments, then the probability that any particular one will be output is $\frac{1}{k}$.

4. (wild card) Implement a box which takes three Boolean inputs P, Q, R and produces the three Boolean outputs $\neg P$, $\neg Q$, $\neg R$. Your box may use any number of AND and OR gates, but may use only two NOT gates.

   Extra credit. Prove that your solution is essentially unique.