Theory Qualifier

January 15, 1997

Instructions

Please read these instructions and all questions C A R E F U L L Y before proceeding.

You will have $2\frac{1}{2}$ hours. The exam is closed-book. All questions are of equal weight. You must attempt to answer all of them to earn an unconditional pass. All algorithms should be accompanied by a proof of correctness and complexity analysis unless otherwise directed.

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, so be sure to show all work. Unclear or irrelevant arguments will be penalized.

GOOD LUCK!

1. Consider the DFA with states $\mathbb{Z}_5 = \{0, 1, 2, 3, 4\}$, input alphabet $\{0, 1\}$, start state 0, final state 0, and transition function

$$\delta(q, i) \quad = \quad (q^2 - i) \bmod 5 \ , \quad q \in \mathbb{Z}_5, \ i \in \{0, 1\} \ .$$

Prove that this DFA accepts exactly the set of binary strings containing an even number of 1's.

2. We say that $\mathcal{A}$ is a *k-approximation algorithm for* π if for all inputs x,

$$\tfrac{\pi(x)}{k} \quad \le \quad \mathcal{A}(x) \quad \le \quad k \cdot \pi(x) \ .$$

Consider the following approximation algorithm for the maximum matching problem.

Run through the list of edges in arbitrary order. For each edge (u, v), add (u, v) to M if $M \cup \{(u, v)\}$ is still a matching.

Here $\pi(x)$ is the number of edges in a maximum matching on graph x and $\mathcal{A}(x)$ is the number of edges in the matching produced by the algorithm.

   (a) How quickly can you implement this algorithm? (The answer "In about 3 or 4 hours" is incorrect.)

   (b) Show that this algorithm is a 2-approximation algorithm for the maximum matching problem.

   (c) Give an example showing that this algorithm is not a c-approximation algorithm for any $c < 2$.

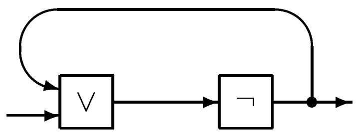
3. Suppose you want to maintain a set of intervals. Assume that intervals are given by their endpoints; i.e., the interval $[5, 9]$ is the set of integers between 5 and 9 inclusive. The operations you will have to do are: **insert** (x, y), **delete** (x, y), **search** (x, y), and **count** (i). The operation **insert** (x, y) inserts the new interval $[x, y]$, **delete** (x, y) deletes the interval $[x, y]$ from the set if it exists, **search** (x, y) checks if the interval $[x, y]$ is in the set, and **count** (i) outputs the number of intervals that contain point i. For example, in the set of intervals $\{[1, 5], [3, 6], [4, 6]\}$, **count** $(5) = 3$ and **count** $(2) = 1$. Design a data structure that allows you to do all of the above operations in $O(\log n)$ time, where n is the number of intervals in the set. You may use any data structures that have been discussed in the texts without explaining how they work or writing code for them.

Hint. Note that **count** (i) is the number of intervals with low endpoint at most i minus the number of intervals with high endpoint less than i.

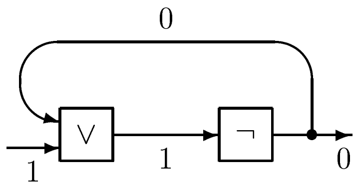
4. Consider circuits composed of and-, or-, and not-gates connected by wires. We allow circuits to have loops, but we assume that there are nonzero propagation delays along wires and through gates. Such a circuit could be *unstable* in the sense that voltages may oscillate due to loops.

   Formally, we say that an assignment of voltages (either 0 or 1) to wires is *stable* if the voltages on the input and output wires of any gate are consistent with the type of that gate. This means that for an or-gate, the output voltage is 1 iff at least one of the two input voltages is 1; for an and-gate, the output voltage is 1 iff both input voltages are 1; and for a not-gate, the output voltage is 1 iff the input voltage is 0.

   For example, consider the following circuit:



   If we assign 0 to the input wire, there is no stable assignment of the remaining voltages; the voltage on the output wire will oscillate between 0 and 1. However, if we assign 1 to the input wire, the following is a stable assignment of voltages:



   (a) Give a circuit with one input wire that has no stable assignment with the input wire set to 0 and two stable assignments with the input wire set to 1.

   (b) Prove that the problem of deciding whether a given circuit has more than one stable assignment is *NP*-complete. You may use the circuit described in part (a) even if you didn't answer part (a).