15-451 Algorithms, Fall 2003

Homework # 7 **due: Thursday December 4, 2003**

Please hand in each problem on a separate sheet and put your **name** and **recitation** (time or letter) at the top of each sheet.

Remember: written homeworks are to be done individually. Group work is only for the oral-presentation assignments.

Problems:

(50 pts) 1. [Different kinds of SAT]

(a) Given a CNF formula, we know the question "does it have a satisfying assignment?" is NP-complete. In fact, the question "does there exist an assignment that satisfies *exactly* one literal per clause?" is also NP-complete. However, the question "does there exist an assignment that satisfies an *odd* number of literals in each clause" *can* be solved in polynomial time.

Give a polynomial-time algorithm to solve this last problem. That is, given a CNF formula, your algorithm answers whether or not there exists an assignment that satisfies an odd number of literals in every clause. Hint: think about modular arithmetic.

(b) Let $\mathcal{A}$ be the set of CNF formulas that have a satisfying assignment, let $\mathcal{B}$ be the set of CNF formulas that have an assignment satisfying an odd number of literals per clause, and let $\mathcal{C}$ be the set of CNF formulas that have an assignment satisfying exactly one literal per clause. Notice that $\mathcal{A} \supseteq \mathcal{B} \supseteq \mathcal{C}$. Your result from part (a) implies the following strange situation: even though $\mathcal{A}$ and $\mathcal{C}$ are NP-complete sets, membership in $\mathcal{B}$ can be decided in polynomial time. Given some formula ϕ, if your algorithm says NO, then we know $\phi \notin \mathcal{C}$, and if your algorithm says YES, then we know $\phi \in \mathcal{A}$.

Suppose we now let $\mathcal{A}$ be the set of pairs (G, k) such that G is a graph with a vertex cover of size k or less. Let $\mathcal{C}$ be the set of pairs (G, k) such that G has a vertex cover of size $k/2$ or less. Notice that if $(G, k) \in \mathcal{C}$ then clearly $(G, k) \in \mathcal{A}$ also, so $\mathcal{A} \supseteq \mathcal{C}$. Describe a set $\mathcal{B}$ such that $\mathcal{A} \supseteq \mathcal{B} \supseteq \mathcal{C}$ but membership in $\mathcal{B}$ can be decided in polynomial time. Hint: think approximation algorithms.

(25 pts) 2. [Ultra-fast long division].

Your company, Codes-R-Us is about to announce a new cryptographic system. The system is based on the assumption that computing exponentially far out digits in the decimal expansion of a fraction is hard. Show that this system is not founded on a good assumption. In particular, give a polynomial time algorithm for the following problem.

Input: integers (a, b, n) in binary notation, where $a < b$.

Let $0.d_1 d_2 d_3 \cdots$ be the decimal expansion of the fraction $\frac{a}{b}$.

Output: d_n.

Note: the key thing here is that your algorithm's running time should be polynomial in $\log n$. The standard way of doing long division would instead be polynomial in n. In particular, the standard long division would look like this:

for $i = 1$ to n do:
  $d_i = 10a$ div b;
  $a = 10a$ mod b;

where "div" is integer division.

(25 pts) 3. [Realizing degree sequences] You are the chief engineer for Graphs-R-Us, a company that makes graphs to meet all sorts of specifications.

(a) A client comes in and says he needs a 4-node directed graph in which the nodes have the following in-degrees and out-degrees:

$$d_{1,in} = 0, d_{1,out} = 2$$
$$d_{2,in} = 1, d_{2,out} = 2$$
$$d_{3,in} = 1, d_{3,out} = 1$$
$$d_{4,in} = 3, d_{4,out} = 0$$

Is there a directed graph, with no multi-edges or self loops, that meets this specification? If so, what is it?

(b) This type of specification, in which the in-degrees and out-degrees of each node are given, is called a *degree sequence*. The question above is asking whether a given degree sequence is *realizable* — that is, whether there exists a directed graph having those degrees.

Find an efficient algorithm that, given a degree sequence, will determine whether this sequence is realizable, and if so will produce a directed graph with those degrees. The graph should not have any self-loops, and should not have any multi-edges (i.e., for each directed pair (i, j) there can be at most one edge from i to j, though it is fine if there is also an edge from j to i). Hint: think network flow.