15-451 Algorithms, Spring 2007

Homework # 4 **due: Tuesday, March 27, 2007**

This homework is to be presented orally in groups of three. Oral Homework signups will operate differently. We will be making available times online on Wednesday (for viewing only, not for signups–any emails sent for signups will be ignored). On Thursday at the end of class, we will have signup sheets with the times available, first come first served. After class, these sheets will be available on Michelle's door, until 5pm Friday. **You must sign up in groups of three. Unauthorized signups with less than 3 individuals will be struck from the schedule. If you need help finding a group, please let your TA know. Groups must sign up by 5pm. If your group hasn't signed up by 5pm, we reserve the right to penalize your group's final score**. In addition, regardless of what day you sign up, help from TAs must be sought before the first group is graded. After that, out of fairness to groups that have already presented, the TAs will not answer non-clarification questions (i.e., no hints). Start your homework early!

Problems:

1. **2Implication** Consider the following problem, which we will call 2Implication. In 2Implication, you are given a set of *implications* of the form $a \Rightarrow b$ where a and b are literals, *i.e.* Boolean variables or negations of Boolean variables. You want to find a way to assign to each of the variables the value **true**, or the value **false**, such that *all* the implications evaluate to **true**.

   Note that (**false** $\Rightarrow b$) evaluates to **true** for all values of the literal b, and (**true** $\Rightarrow b$) evaluates to the value of b.

   For example, here is an instance of 2Implication:

   $$(\neg x_1 \Rightarrow \neg x_2) \wedge (x_1 \Rightarrow \neg x_3) \wedge (\neg x_1 \Rightarrow x_2) \wedge (x_3 \Rightarrow x_4) \wedge (x_1 \Rightarrow x_4)$$

   It is possible to make this instance evaluate to **true** by setting x_1 and x_4 to **true**, and x_2 and x_3 to **false**.

   (a) Are there any other variable assignments that make this 2Implication formula true? If so, give them all.

   (b) Give an instance of 2Implication, with four variables, such that there is no variable assignment that makes the formula true.

   Given an instance I of 2Implication, with n variables and m implications, we can construct a directed graph $G_I = (V, E)$ as follows.

   - G_I has $2n$ nodes, one for each variable and its negation.
   - G_I has $2m$ edges: for each implication $(a \Rightarrow b)$ of I (where a, b are literals), G_I has an edge from a to b, and an edge from $\neg b$ to $\neg a$.

(c) Use this construction to represent the instance of 2Implication given above, and the instance you designed for part (b).

(d) Show that if G_I has a strongly connected component containing both x and $\neg x$ for some variable x, then there is no variable assignment that makes I true.

(e) Now show the converse of part (d). In other words, show that if none of G_I's strongly connected components contain both a literal and its negation, then there is a variable assignment, such that the instance I evaluates to true.

(f) Use your answers to the previous parts to give a linear time algorithm for solving 2Implication i.e., an algorithm that gives the variable assignment if it exists, or concludes that there is no variable assignment that can make the instance true.

2. **Currency Trading** Let $c_1, c_2, ..., c_n$ be various currencies. For example, c_1 might be the USD, c_2 the EURO and c_n the AUD. For any two currencies c_i and c_j, there is an exchange rate $r_{i,j}$. If currencies c_i and c_j have exchange rate $r_{i,j}$ then you can purchase $r_{i,j}$ units of currency c_j with one unit of c_i. The exchange rates satisfy the following condition: $r_{i,j} \cdot r_{j,i} < 1$. This means that if you start with one unit of c_i, convert it to c_j, and then convert it back to c_i, you will have less than one unit of c_i. The difference represents the transaction cost.

   (a) Give an efficient algorithm for the following problem: Given a set of exchange rates $r_{i,j}$ and two currencies s and t, find the most advantageous sequence of currency exchanges for converting currency s to currency t. Hint: represent the currencies and rates by a graph whose edge lengths are real numbers.

   Exchange rates are updated frequently to represent the changing supply and demand for currencies. Sometimes the exchange rates satisfy the following property: there exists a sequence of currencies $c_{i_1}, c_{i_2}, ..., c_{i_k}$ such that $r_{i_1,i_2} \cdot r_{i_2,i_3} \cdots r_{i_{k-1},i_k} \cdot r_{i_k,i_1} > 1$. So by starting with one unit of currency c_{i_1} and then successively converting it to currencies $c_{i_1}, c_{i_2}, ..., c_{i_k}$, and then finally back to c_{i_1}, you can actually end up with more than one unit of c_{i_1}. These opportunities last for only a fraction of a minute on the currency exchange, but they are a risk free way of making money.

   (b) Using the graph representation you used in part (a), give an efficient (definitely polytime, but hopefully very fast) algorithm for detecting the presence of this type of anomaly.

3. **Order Graphs** We define an *Order Graph* as a directed graph, $G = (V, E)$, where, for every pair of vertices u, v there is either an edge from u to v or an edge from v to u. In other words, for every pair $u, v \in V$ there is exactly one directed edge between u and v in E. No self-loops are allowed.

   A vertex v is said to *lead* the order graph if for every other vertex $u \in V$ one of the following conditions holds:

   1. there is a directed edge from v to u
   2. there exists another vertex w, such that there is a directed edge from v to w, and a directed edge from w to u.

   (a) Prove that every order graph has a *lead* vertex.

   The strongly connected component graph (or condensation) of G, is the graph obtained by representing each maximal strongly connected component by a single vertex.

   (b) Show that the strongly connected component graph, $G_s = (V_s, E_s)$, of an order graph G gives a transitive ordering of the vertices. For example, if $(a, b) \in E_s$ and $(b, c) \in E_s$ then $(a, c) \in E_s$.

   (c) Prove that every order graph has a Hamiltonian path.

   (d) Give an efficient (polytime) algorithm to find a Hamiltonian path in an order graph G. Explain why your algorithm is correct and give its runtime.