Algorithmic Complexity

CS601A — *Homework 4*
Spring 2004 — *Due: April 27, 2004*
Prof. Rebecca N. Wright

Each problem is worth 20 points. If at all possible, please use an appropriate text-processing package to write up your solutions, such as LaTeX or MS-Word. You may, if you wish, leave space for mathematical formulas and drawings and insert them by hand. If you must turn in hand-written work, please make sure it is legible.

Problem 1 (Subgraph Isomorphism):
The Subgraph Isomorphism problem is defined as follows:

> Subgraph Isomorphism: Given two undirected graphs $G_1 = (V_1, E_1)$ and $H = (V_2, E_2)$, does G contain a subgraph that is isomorphic to H? (That is, if we ignore differences between the names of the edges, does G contain H? More precisely, is there a subset $V \subseteq V_1$ and a subset $E \subseteq E_1$ such that $|V| = |V_2|$, $|E| = |E_2|$, and there is a one-one function $f : V_2 \to V$ satisfying $\{u, v\} \in E_2$ if and only if $\{f(u), f(v)\} \in E$?)

Show that Subgraph Isomorphism is **NP**-complete, using Hamiltonian Circuit (the undirected version) for the reduction.

Problem 2 (Graph 3-Colorability): The Graph 3-Colorability problem is defined as follows:

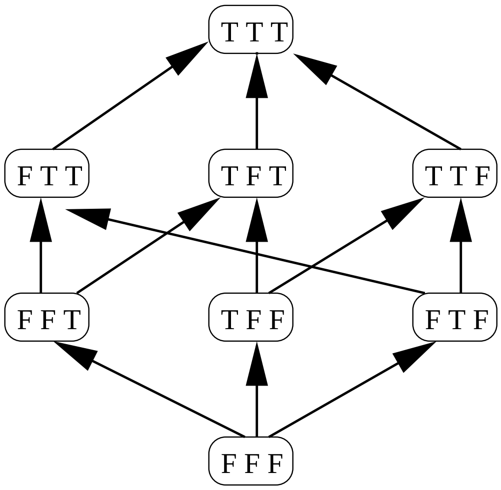
> Graph 3-Colorability: Given an undirected graph $G = (V, E)$, is G 3-colorable? (That is, is there a function $f : V \to \{1, 2, 3\}$ such that $f(u) \neq f(v)$ whenever $\{u, v\} \in E$?)

Show that Graph 3-Colorability is **NP**-complete. [Hint: You may wish to use 3Sat.]

Problem 3 (Monotone Boolean Functions):
A Boolean function f is *monotone* if changing an input from **false** to **true** cannot change the value of the function from **true** to **false**. We will show that a Boolean function f is monotone if and only if f can be represented by a Boolean circuit using only AND and OR gates.

Part a (10 points): Show that if f can be represented by a Boolean circuit using only AND and OR connectives, then f is monotone. [Hint: one approach to prove this is by induction on the number of gates in the Boolean expression.]

Part b (10 points): To show the converse, you may find it helpful to consider the following graph representing the possible truth assignments to the variables of an n-ary Boolean function: we construct a graph G_n, where the nodes of G_n are the 2^n possible truth assignments to $x_1, \ldots, x_n$, with an edge from node V_1 to node V_2 if the truth assignment V_2 can be obtained from the truth assignment V_1 by changing exactly one variable from **false** to **true**. For example, for $n = 3$, G_n is the following graph:



where, for example the node V = TFF represents the truth assignment x_1 = **true**, x_2 = **false**, x_3 = **false**.