Problem Set 10 Solutions

Note the unusual day for this (minimal) assignment being due.

Problem 1. Let \( SAT_{20} = \{ \langle \phi \rangle : \phi \text{ has at least twenty different satisfying assignments} \} \). Show that \( SAT_{20} \) is NP-complete.

First, it is easy to see that \( SAT_{20} \in NP \). On input of a Boolean formula \( \phi \), a verifier could guess twenty Boolean assignments, \( t_1, t_2, t_3, \cdots, t_{20} \), and then verify that these assignments are different from one another and that each satisfies \( \phi \). In other words, the certificate consists of 20 distinct satisfying assignments \( t_1 \) through \( t_{20} \).

To show that \( SAT_{20} \) is NP-hard, we show that \( SAT \leq_p SAT_{20} \). The function \( f \) that maps an instance \( \phi \) of SAT to an instance \( \phi' \) of SAT20 works as follows:

\[
\phi' = \phi \land (x_1 \lor x_2 \lor x_3 \lor x_4 \lor x_5),
\]

where each \( x_i \) is new variable (they don’t occur in \( \phi \)). This reduction is certainly polynomial time. Now if \( \phi \) is unsatisfiable, certainly \( \phi' \) is, too, since we have only conjuncted an additional term. But if \( \phi \) has some satisfying assignments \( t \), than \( \phi' \) has at least 31 (and therefore at least 20) satisfying assignments, (corresponding to the 32 different ways of extending \( t \) to the new variables \( x_1, x_2, x_3, x_4, x_5 \).

Problem 2. A graph \( G = (V, E) \) is said to be \( k \)-colorable if there is a way to paint its vertices using colors in \( \{1, 2, \ldots, k\} \) such that no adjacent vertices are painted the same color. Let \( G3C \) denote the language of encodings of 3-colorable graphs. Let \( G4C \) denote the language of encodings of 4-colorable graphs. The language \( G3C \) is NP-Complete. (We will prove this on Monday.) Use this to prove that \( G4C \) is NP-Complete, too.

First, it is easy to see that \( G4C \) is in NP. Given a graph \( G = (V, E) \) you need only guess a coloring \( c : V \rightarrow \{1, 2, 3, 4\} \) and then verify that \( c(x) \neq c(y) \) for all \( \{x, y\} \in E \) (as well as \( c(x), c(y) \in \{1, 2, 3, 4\} \)). Clearly this takes a polynomial amount of time.

Now we have to show \( G3C \leq_p G4C \). Given a graph \( G = (V, E) \) (an instance of the G3C problem) we produce a graph \( G' = (V', E') \) as follows: \( V' = V \cup \{z\} \), and \( E' = E \cup \{\{x, z\} : x \in V\} \), where \( z \) is a name for a vertex not in \( V \). That is, \( G' \) is constructed by adding a new vertex to \( G \) and connecting it to every node of \( G \). Clearly if \( G \) is 3-colorable then \( G' \) will be 4-colorable; just use the new color for the newly-added vertex. Conversely, if \( G' \) is 4-colorable then \( G \) must be 3-colorable, since the color used for vertex \( z \) has to be different from the color used on every other vertex, and so restricting the coloring \( c' \) of \( G' \) to the nodes of \( G \) will immediately give a 3-coloring of \( G \), apart from the names of the colors used. Finally, observe that the reduction itself is polynomial-time computable.