CSE 431 – Theory of Computation. Spring, 2014. Instructor: James R. Lee

ASSIGNMENT 6. Due Thursday, May 21st, in class (or via email to cse431-staff@cs before class starts)

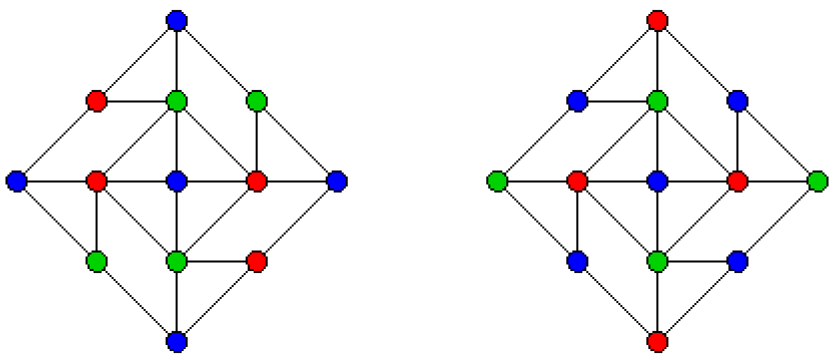
1. Recall the graph 3-coloring problem:

   3-COL = { $\langle G \rangle$: G is a 3-colorable graph }

   Also from class, we saw the problem

   PLANAR-3-COL = {$\langle G \rangle$: G is a planar 3-colorable graph }

   Your goal is to prove that 3-COL $\leq_P$ PLANAR-3-COL, thereby proving that PLANAR-3COL is NP-complete. You should use the following gadget to uncross edges:

   Do the problem in three parts:

   (a) Given colors $c_1, c_2 \in \{R, G, B\}$, observe that there is always a way to 3-color the graph so that the opposite east-west corners are colored c_1 and the opposite north-south corners are colored c_2. Note that possibly $c_1 = c_2$. (That this is true follows from the two colorings given above.)

   (b) Show that **any** 3-coloring of the gadget must have the property that opposite corners have the same color.

   (c) Use this to reduce 3-COL to PLANAR-3-COL. Remember that if the edges x-y and u-v cross, then the gadget should remove the edge crossing, but enforce the same constraints that x/y must be colored differently and u/v must be colored differently.

2. A **monomial** in variables $x_1, x_2, \ldots, x_n$ is a product $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$, where the a_i's are natural numbers. An **integral polynomial** in $x_1, x_2, \ldots, x_n$ is a sum of monomials with integer coefficients. For instance,

$$p(x_1, x_2, x_3) = 4x_1x_2 - 7x_1x_3^2 + 11\, x_1^2x_2^2x_3 + 2$$

A root $(z_1, z_2, \ldots, z_n)$ of a polynomial p in n variables is a sequence of numbers such that $p(z_1, z_2, \ldots, z_n) = 0$. A root is integral if all the z_i's are integers.
Consider the language:

INTEGRAL-ROOT = { $\langle p \rangle$: p is a polynomial with an integer root }

a) Show that 3-SAT $\leq_P$ INTEGRAL-ROOT.
b) Does this imply that INTEGRAL-ROOT is NP-complete? What's the difficulty?

3. So far we have talked about **decision** problems where we simply want YES or NO answers, like: Is a Boolean formula ϕ satisfiable? Maybe if P=NP then answering such questions is easy, but actually **finding** the solution (in this case, the satisfying assignment) is still hard! In this problem, you will show that this isn't the case.
a) Show that if P=NP, there is a polynomial time algorithm that, given a Boolean formula ϕ, actually outputs a satisfying assignment. [Hint: If P=NP, then given a formula ϕ, there is poly-time algorithm to see if ϕ has a satisfying assignment. Use this algorithm to **FIND** a satisfying assignment by figuring it out bit-by-bit. In other words, figure out a good value for x_1 then for x_2 and so on. You will do this by running the satisfiability-checker many times on modifications of ϕ.]
b) Show that if P=NP, there is a polynomial-time algorithm that produces a 3-coloring of a graph G is such a coloring exists.

OPTIONAL PROBLEM (You may do this problem for extra credit, OR you can do it instead of the first three problems!)

Prove that if P=NP, then you can break the RSA cryptosystem (http://en.wikipedia.org/wiki/RSA_(cryptosystem)) . In other words, show that given someone's public key, you can compute their private key in polynomial time.