Undergraduate Complexity Theory **CMU 15-455, Spring 2017**

HOMEWORK 6

Due: 5:00pm, Thursday March 2

Feature: As before, if your homework is typeset (as opposed to handwritten), you will receive 1 bonus point.

---

1. **(3-coloring with unary constraints.)** (10 points.) Let 3COL+UNARY be the same problem as 3COL, except that there can be additional constraints of the form "*This vertex must be* ⟨*color*⟩". That is, the input is an undirected graph in which some of the vertices may be prelabeled with either R, B, or Y. The task is to decide if the unlabeled vertices can be colored with $\{R, B, Y\}$ so that the whole graph becomes validly 3-colored (no edge having the same colors for its endpoints). Show that 3COL+UNARY $\leq_m^P$ 3COL. Please do this reduction "directly".[1]

2. **(NP-hard but not NP-complete.)** (10 points.) Show that the IMPLICIT-4COL problem (from Homework 5, #3) is NP-hard.

   (Remark: It can be shown that IMPLICIT-4COL is *not* in NP. We may show this later in the course.)

3. **(MAX-2SAT.)** (10 points.) Recall that the 2SAT problem is in P. Show that the analogous "optimization version", MAX-2SAT, is NP-complete. The definition is:

   $$\text{MAX-2SAT} = \{\langle \phi, k \rangle : \phi \text{ is a 2-CNF formula for which there is a truth assignment satisfying at least } k \text{ of } \phi\text{'s clauses}\}.$$

   (Hint: Maybe reduce from NAE-E3SAT. Maybe use six width-2 clauses per NAE constraint.)

4. **(A generic NP-complete problem.)** The Cook–Levin Theorem shows that CIRCUIT-SAT is NP-complete. The cool thing about this theorem is not so much that an NP-complete language *exists*, as that there is a pretty *natural* NP-complete problem, namely CIRCUIT-SAT. If you understand the (verifier-based) definition of NP well, it is not too hard to show that the language

   $$\text{TS} := \{\langle M, x, 1^w, 1^t \rangle : M \text{ is a 2-input TM; } x \text{ is a Boolean string; } w, t \in \mathbb{N}; \exists u \in \{0,1\}^w \text{ such that } M(\langle x, u \rangle) \text{ accepts within } t \text{ steps}\}$$

   is NP-complete. (A small note here: 1^w denotes the string of all-1's of length w, and similarly for 1^t. This is a common complexity theory trick, "encoding numbers in unary", to allow a Turing Machine to run in time polynomial in the number itself, rather than in the length of its base-2 representation.)

   (a) (2 points.) Show that TS $\in$ NP.

   (b) (8 points.) Complete the proof that TS is NP-complete by *directly* showing that every language[2] in NP has a polynomial-time mapping reduction to TS.[3]

[1] That is, please don't reduce 3COL+UNARY to some other well-known NP problem and then say "3COL is NP-complete".

[2] For simplicity, you may concern yourself only with languages over the alphabet $\{0, 1\}$.

[3] That is, please do not appeal to the Cook–Levin Theorem in your proof.