CS 21 Decidability and Tractability **Winter 2014**

Solution Set 5

Posted: February 28

If you have not yet turned in the Problem Set, you should not consult these solutions.

1. (a) False. By the Time Hierarchy Theorem there are problems that are solvable in time $O(2^{3(2n)^{100}})$ that are not in TIME$(2^{n^{100}})$. Nevertheless, a problem with an algorithm running in time $O(2^{3(2n)^{100}})$ is in EXP.

   (b) True. If both $P = NP$ and $NP = EXP$, then $P = EXP$. But we know that $P \neq EXP$, by the Time Hierarchy Theorem.

   (c) False. Reductions may have running time $O(n^{k'})$ for $k' > k$, which spoils the straightforward argument. Formally the proof that it is false goes as follows. The premise implies $P = NP$. But using the Time Hierarchy Theorem exactly as we did in (a), there are languages in P that do not have algorithms running in time $O(n^k)$.

   (d) False. The empty language $\emptyset$ and its complement cannot be NP-complete, because no other language can reduce to them. To see this, suppose we have a reduction f from $A \in NP$ to $\emptyset$ (where $A \neq \emptyset$). Pick an arbitrary $w \in A$. Since f is a reduction, we must have $f(w) \in \emptyset$, which is impossible.

   (Note, however, that if $P = NP$, then every language other than $\emptyset$ and $\overline{\emptyset}$ is NP-complete. Take an arbitrary language $L \in NP$, and pick some $w \in L$, and some $w' \notin L$. Here's a reduction from an arbitrary language $A \in NP$ to L: given an instance x, our reduction f decides in polynomial time whether $x \in A$ or not, and in the first case it outputs w, in the second case it outputs w'.)

   (e) True. By the definition of "complete." Every language in NP is in EXP since $NP \subseteq EXP$, and thus it reduces to every EXP-complete language.

2. First, note that 3-COLORABLE is in NP because given an assignment of colors to the vertices of the graph G, we can verify in polynomial time that each edge has different colors at its endpoints.

   To show that 3-COLORABLE is NP-hard, we reduce from 3SAT. Given an instance ϕ of 3SAT, our reduction produces the following graph G: we have three special vertices A, B, C, and one vertex for each literal, x_i and $\neg x_i$. We have a triangle on $A, x_i, \neg x_i$ for each i. Notice that any 3-coloring of the graph so far must assign distinct colors to B and C, which we will call (suggestively) T and F, respectively. Also each pair x_i and $\neg x_i$ must be colored with T and F, respectively, or F and T, respectively. We can thus think of the coloring of the "literal vertices" as a truth assignment.

   Now, for each clause $(\ell_1 \vee \ell_2 \vee \ell_3)$ appearing in ϕ, we add a copy of the gadget from the problem set. We identify the three grey vertices on the left with the three vertices ℓ_1, ℓ_2, and ℓ_3. We identify the grey vertex on the right with the vertex B. This reduction runs in polynomial time.

If we started with a YES instance of 3SAT, then we claim that the reduction produces a YES instance of 3-COLORABLE. Consider a satisfying assignment for ϕ. We color A, B, and C with the colors RED, T and F, respectively. We then color x_i and $\neg x_i$ with colors T and F, respectively, if x_i is true in the satisfying assignment, and we color x_i and $\neg x_i$ with colors F and T, respectively, if x_i is false in the satisfying assignment. So far this is a valid 3-coloring. Now notice that every one of the clause gadgets has among its left three grey nodes at least one node that is colored T (since every clause has at least one true literal in the satisfying assignment). Moreover, its right grey node is colored T. Thus using the observation in the problem set we can extend the 3-coloring to a 3-coloring of the clause gadgets, obtaining a 3-coloring of the entire graph G.

Now, if G is a YES instance of 3-COLORABLE, then we claim that the reduction started with a satisfiable formula ϕ. Suppose we have a 3-coloring of G, then A, B, and C must be colored with three distinct colors, and let us call the color assigned to B "T" and the color assigned to C "F". As noted each pair x_i and $\neg x_i$ must be colored with T and F respectively, or F and T respectively. For each clause gadget, the rightmost grey node is colored with T, and so by the observation in the problem set, the only way it can be 3-colored is if at least one of its leftmost grey nodes are colored with T. But this means that we can set x_i to true if it is colored T and x_i to false if it is colored F, and the resulting assignment must satisfy every clause. Thus the formula ϕ is satisfiable, as required.

3. $(3,3)$-SAT is in NP for the same reason 3-SAT is. We show that it is NP-hard by reducing from 3-SAT.

   Given a 3-CNF formula ϕ, we perform the following transformation to obtain 3-CNF ϕ': for each x_i we replace the m_i occurrences of x_i with fresh variables $y_{i,1}, y_{i,2}, \ldots, y_{i,m_i}$, and we add the following m_i clauses:

$$\begin{aligned}(\neg y_{i,1} \vee y_{i,2})\\ (\neg y_{i,2} \vee y_{i,3})\\ (\neg y_{i,3} \vee y_{i,4})\\ \ldots\\ (\neg y_{i,m_i-1} \vee y_{i,m_i})\\ (\neg y_{i,m_i} \vee y_{i,1})\end{aligned}$$

   Note that the extra clauses are logically equivalent to:

$$y_{i,1} \Rightarrow y_{i,2} \Rightarrow \ldots \Rightarrow y_{i,m_i} \Rightarrow y_{i,1}$$

   and hence any assignment satisfying ϕ' must set all of these variables to the same value. Such an assignment can be turned into a satisfying assignment for ϕ by setting $x_i = y_{i,1}$ for all i. Similarly, any assignment satisfying ϕ can be turned into a satisfying assignment for ϕ' by setting $y_{i,j} = x_i$ for all i, j. This shows that "yes maps to yes" and "no maps to no," and thus $(3,3)$-SAT is NP-hard as required.

4. We first prove the following lemma regarding the 10 clauses given on the problem set: any assignment to x, y, z that sets at least one to true can be extended to an assignment to x, y, z, w

that satisfies at most 7 of the 10 clauses, while the assignment to x, y, z that sets all of them to false cannot be extended to an assignment to x, y, z, w that satisfy more than 6 of the 10 clauses. Moreover it is impossible to satisfy more than 7 of the 10 clauses simultaneously.

The second part is easier: if x, y, z are all false, then setting w to true satisfies 4 clauses, while setting w to false satisfies 6 clauses. For the first part, observe that the clauses are symmetric in x, y, z. Thus we need only consider 3 cases: (a) exactly one of x, y, z is true, (b) exactly 2 of x, y, z are true, and (c) exactly 3 of x, y, z are true. In case (a), we can set w to false to satisfy 1 clause in the first row, 3 in the second row, and 3 in the last row, for a total of 7. In case (b), we can set w to true to satisfy 3 clauses in the first row, 2 in the second row, and 2 in the last row, for a total of 7. In case (c) we can set w to be true to satisfy 4 clauses in the first row, no clauses in the second row, and 3 clauses in the last row, for a total of 7. In each of these three cases, we see that setting w the other way doesn't help: in case (a) we would satisfy only 6 clauses; in case (b) we would satisfy 7 clauses; in case (c) we would satisfy only 6 clauses. This proves the "moreover" part of the lemma.

Now we proceed with proving MAX2SAT is NP-complete. Observe that it is in NP, because given a formula ϕ and an integer k, together with an assignment A, it is easy to verify in polynomial time that the assignment satisfies at least k of the clauses of ϕ.

Now, we show MAX2SAT is NP-hard by reducing from 3SAT. Given an instance ϕ of 3SAT with m clauses, we produce a 2-CNF formula ϕ' by replacing each clause $(\ell_1 \vee \ell_2 \vee \ell_3)$ with the 10 clauses in the problem set, with ℓ_1, ℓ_2, ℓ_3 in place of x, y, z. We use a fresh variable w for each clause of ϕ. The reduction sets the bound $k = 7m$. This reduction runs in polynomial time (it produces a 2-CNF ϕ' with $10m$ clauses).

Suppose ϕ is satisfiable by an assignment A. Then by the lemma above, we can extend A to an assignment to ϕ' that satisfies at least $k = 7m$ clauses of ϕ' simultaneously, which implies that (ϕ', k) is a YES instance of MAX2SAT.

Now, suppose that there is an assignment that simultaneously satisfies at least $k = 7m$ clauses of ϕ'. Since the maximum number of clauses that can be satisfies within each group of 10 clauses is 7, this means that *every* group of 10 clauses has 7 clauses satisfied by the assignment. But by the lemma this means that this same assignment must satisfy every clause of ϕ, because if it didn't satisfy even a single clause, then the associated group of 10 clauses of ϕ' would only have 6 clauses satisfied by the current assignment. Thus we conclude that ϕ must be satisfiable.