CS46 practice problems 12

These practice problems are an opportunity for discussion and trying many different solutions. It is **not counted towards your grade**, and **you do not have to submit your solutions.** The purpose of these problems is to get more comfortable with reasoning and writing about P, NP, and polynomial-time reductions.

If you are stumped or looking for guidance, **ask**.

1. Show that if coNP $\neq$ NP then P $\neq$ NP.

2. A **vertex cover** in a graph $G = (V, E)$ is a subset $S \subset V$ of vertices where every edge of G has at least one endpoint in the subset.

   $$\textsc{VertexCover} = \{\langle G, k\rangle \mid G \text{ has a } k\text{-node vertex cover}\}$$

   An **independent set** in a graph G is a subset of vertices with no edges between them.

   $$\textsc{IndependentSet} = \{\langle G, k\rangle \mid G \text{ contains an independent set of } k \text{ vertices}\}$$

   Show that $\textsc{IndependentSet} \leq_p \textsc{VertexCover}$.

3. **Boolean formulas, NP , and an application of NFAs.**

   Some useful vocabulary:

   - A **literal** is a Boolean variable or a negated Boolean variable, like x or $\overline{x}$.
   - The symbol "$\vee$" means "or". (This is a **disjunction**.)
   - The symbol "$\wedge$" means "and". (This is a **conjunction**.)
   - A Boolean formula is an expression involving Boolean variables and operations, for example $(\overline{x} \wedge y) \vee (x \wedge \overline{z})$.
   - A **clause** is a disjunction of literals, like $x \vee y \vee \overline{z}$.
   - A formula is **satisfiable** if there is a truth assignment (giving a truth value to each variable) which makes the entire formula evaluate to TRUE.
   - A Boolean formula is in **conjunctive normal form** (CNF) if it is written as the conjunction of clauses, for example:

   $$(x_1 \vee x_2) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4) \wedge (x_5 \vee \overline{x}_1 \vee x_6) \wedge (x_3)$$

   (a) Show that $\textsc{Satisfiability} \in$ NP, where

   $$\textsc{Satisfiability} = \{\langle\phi\rangle \mid \phi \text{ is a satisfiable CNF Boolean formula}\}$$

   There are two ways to do this: either give a nondeterministic polynomial-time decider, OR give a deterministic polynomial-time verifier. You should try both techniques for showing $\textsc{Satisfiability} \in$ NP. (They will have very similar details!)

   (b) Define the language:

   $$L = \{\langle\phi\rangle \mid \phi \text{ is a satisfiable CNF formula where each variable appears at most twice}\}$$

   Show that $L \in P$.

(c) For a CNF formula ϕ with m variables and c clauses, show you can construct in polynomial time an NFA with $O(cm)$ states that accepts all *non*satisfying assignments, represented as binary strings of length m.

(This implies that if $P \neq NP$, then NFAs cannot be minimized in polynomial time.)