Tutorial 9, Design and Analysis of Algorithms, 2019

1. Prove that the following languages are in NP:
   (a) Two coloring: 2COL $= \{G \mid$ graph G has a coloring with 2 colors$\}$, where a coloring of G with c colors is an assignment of a number in $[1..c]$ to each vertex such that no adjacent vertices get the same number.
   (b) Three coloring: 3COL $= \{G \mid$ graph G has a coloring with 3 colors$\}$.
   (c) Connectivity: CONNECTED $= \{G \mid G$ is a connected graph$\}$.
   (d) Which of the above problems is in P. Prove your result.
2. Suppose $L_1, L_2 \in$ NP. Then is $L_1 \cup L_2$ in NP? What about $L_1 \cap L_2$? Prove your result.
3. Prove that allowing the certificate to be of size at most $p(|x|)$ (rather than equal to $p(|x|)$) in the certificate definition of NP, makes no difference. That is, show that for every polynomial-time Turing machine M and polynomial $p : N \to N$, the language

   $$\{x : \exists u \mid |u| \leq p(|x|) \text{ and } M(x, u) = 1\}$$

   is in NP.
4. We have defined a relation $\leq_p$ (polynomial time reduction) among languages. We noted that it is reflexive (that is, $L \leq_p L$ for all languages L) and transitive (that is, if $L_1 \leq_p L_2$ and $L_2 \leq_p L_3$ then $L_1 \leq_p L_3$). Show that it is not symmetric, namely, $L_1 \leq_p L_2$ need not imply $L_2 \leq_p L_1$.
5. For languages $L_1, L_2 \subseteq \{0, 1\}^*$, let

   $$L_1 \oplus L_2 = L_1\{0\} \cup L_2\{1\}$$

   (a) Prove that $L_1 \leq_p L_1 \oplus L_2$ and $L_2 \leq_p L_1 \oplus L_2$.
   (b) Prove that for any languages L, L_1, and L_2 over $\{0, 1\}$, with $L \neq \{0, 1\}^*$, if $L_1 \leq_p L$ and $L_2 \leq_p L$, then $L_1 \oplus L_2 \leq_p L$.

   Notation: $A\{x\} = \{wx \mid w \in A\}$, $B \leq_p C$ means that B reduces to C in polynomial time.
6. Prove that the following language is NP-complete:

   BOUNDED HALTING $= \{(M, 1^t) \mid \exists x, |x| \leq t,$ such that DTM M accepts x within t steps$\}$

   Notation: 1^t *is* 1 *written* t *times, where* t *is an integer.* $|x|$ *is the length the string* x.
7. We define the complexity class **coNP** as

   $$\textbf{coNP} = \{L \mid \overline{L} \in \textbf{NP}\}$$

   We can define coNP-completeness in analogy to NP-completeness: a language is **coNP-complete** if it is in **coNP** and every **coNP** language is polynomial-time reducible to it. Prove that the following language is **coNP-complete**:

   TAUTOLOGY $= \{\phi \mid \phi$ is a tautology - a Boolean formula that is satisfied by every assignment$\}$