How to Prove That a Language is NP-Complete

There are two components to the definition of NP-completeness: being NP-hard and being in NP. A language must satisfy both of these components to be NP-complete. A proof that a language $L$ is NP-complete must have the following parts:

1. An NTM $M$ that decides $L$ in nondeterministic polynomial time, showing that $L \in \text{NP}$.
   (i) A proof of $M$’s correctness. In particular, you must show that $M$ accepts a string if and only if that string is in $L$.
   (ii) A proof that $M$ runs in nondeterministic polynomial time.

2. A polynomial-time reduction from an NP-hard language $A$ to $L$, showing that $L$ is NP-hard.
   (i) A proof of the reduction’s correctness. All “yes” instances in $A$ must map to “yes” instances in $L$, and all “no” instances in $A$ must map to “no” instances in $L$.
   (ii) A proof that the reduction runs in polynomial time.

3. Finally, conclude that $L$ is NP-complete.

In this and future problem sets, proving that a language is NP-complete means including all of these components.
The following questions are lab problems.

Lab Problem 1

In the Fangorn Forest there are various kinds of trees including Ents, Huorns, and others. Saruman needs fuel for the furnaces at Isengard, so he decides to send his Uruk-Hai to chop down some trees in the forest. He wants to cut down at least one of each kind of tree, but in order to avoid provoking the wrath of Treebeard, he also wants to avoid completely eliminating any type of tree. Given a finite set of trees $S$ in the forest and a collection of subsets $C_1, \ldots, C_k$ where $C_i$ is the set of trees of type $i$, he wants to know whether it is possible to decide on a logging plan such that no $C_i$ has all of its elements cut, nor all of its elements uncut.

The language Saruman would like to recognize is thus:

$$L = \{\langle S, C \rangle \mid \text{there is a valid logging plan given trees } S$$
$$\text{ and types } C = \{C_1, \ldots, C_k\}\}$$

Prove that $L$ is NP-complete.

**Note:** The collection $C$ is not necessarily a partition of $S$. Perhaps trees can fall under multiple types, or have no type at all.

Lab Problem 2

In this problem, consider the following language defined for Boolean formulae $\phi$ over variables $x_1, x_2, \ldots, x_n$:

$$\text{DOUBLESAT} = \{\langle \phi \rangle \mid \phi \text{ has at least two satisfying assignments}\}$$

Prove that DOUBLESAT is NP-complete.
Lab Problem 3

Recall that a **clique** of size $n$ in a graph $G = (V, E)$ is a subset of $V$ of size $n$ such that every pair of vertices in the subset is connected by an edge in $E$.

In this problem, you will show that the language CLIQUE is NP-complete:

$$\text{CLIQUE} = \{(G, n) \mid G = (V, E) \text{ is an undirected graph and } G \text{ contains a clique of size at least } n\}$$

To prove that CLIQUE is NP-hard, you can use the following reduction from 3SAT to CLIQUE:

*Given a 3CNF Boolean formula, convert it to an undirected graph as follows: Create a vertex for each literal in each clause. Draw an edge from every vertex to every other vertex unless the two vertices correspond to literals from the same clause, or the two vertices correspond to literals that are negations of each other.*

Let $k$ be the number of clauses in the 3CNF Boolean formula. Your job is to show that the constructed graph will have a clique of size $k$ if and only if the Boolean formula is satisfiable.

a. Show that you understand the reduction by converting the following 3CNF Boolean formula to the associated graph:

- **Literals:** $X = \{x_1, \overline{x}_1, x_2, \overline{x}_2, x_3, \overline{x}_3\}$
- **Clauses:** $C = \{(x_1, x_2, x_3), (\overline{x}_2), (x_2, \overline{x}_3)\}$

**Note:** Recall that 3CNF means there are at most three literals in each clause.

b. Explain why any satisfiable 3CNF Boolean formula is translated into a graph with a clique of size at least $k$ using the above reduction.

c. Explain why having a clique of size at least $k$ in a graph resulting from this reduction implies that the original 3CNF Boolean formula was satisfiable.

d. Describe an NTM that determines whether a graph has a clique of size at least $k$. Use this NTM to explain why CLIQUE is in the class NP.

e. How do parts (b), (c), and (d) help show that CLIQUE is NP-complete? This can be a brief sentence or two.

**Hint:** Is there a connection between this reduction and the textbook’s reduction from 3SAT to INDEPENDENTSET?