CS41 Lab 10: Polynomial-Time Verifiers and Polynomial-Time Reductions

This week, we've started to understand what makes some problems seemingly hard to compute. In this lab, we'll consider an easier problem of *verifying* that an algorithm's answer is correct.

Recall that a *decision problem* is a problem that requires a YES or NO answer. Alternatively, we can describe decision problem as a set $L \subseteq \{0,1\}^*$; think of L as the set of all YES inputs i.e., the set of inputs x such that one should output YES on input x. Let $|x|$ denote the length of x, in bits.

1. **Polynomial-time Verifiers**. Call V an efficient *verifier* for a decision problem L if

   (a) V is a polynomial-time algorithm that takes two inputs x and w.

   (b) There is a polynomial function p such that for all strings x, $x \in L$ if and only if there exists w such that $|w| \leq p(|x|)$ and $V(x, w) = \text{YES}$.

   w is usually called the *witness* or *certificate*. Think of w as some *proof* that $x \in L$. For V to be a polynomial-time verifier, w must have size some polynomial of the input x. For example, if x represents a graph with n vertices and m edges, the length of w could be n^2 or m^3 or $(n+m)^{100}$ but not 2^n.

   Give polynomial-time verifiers for the following problems, none of which are known to have polynomial-time algorithms.

   (a) THREE-COLORING. Given $G = (V, E)$ return YES iff the vertices in G can be colored using at most three colors such that each edge $(u, v) \in E$ is *bichromatic*.

   (b) SAT. Recall the input for SAT is a set of boolean variables $x_1, \ldots, x_n$ along with m clauses $c_1, \ldots, c_m$, where each clause is the OR of one or more literals e.g.

   $$c_j = x_1 \vee \bar{x}_2 \vee \cdots \vee \bar{x}_{k-1} \vee x_k \ ,$$

   where $\bar{x}_i$ represents the negation of x_i.
   Output YES iff there exists a truth assignment that *satisfies* every clause.

   (c) FACTORING. Given numbers n, k written in binary, output YES iff n is divisible by d for some $1 < d \leq k$.

   (d) NOT-FACTORING. Given numbers n, k written in binary, output YES iff n is **NOT** divisible by d for any $1 < d \leq k$.

   Hint: The following problem **is solvable** in polynomial time.[1]

   PRIMES: Given a number n written in binary, output YES iff n is a prime number.

2. **Independent-Set and Vertex-Cover**. Given a graph $G = (V, E)$, an *independent set* is a set of vertices $W \subseteq V$ such that no edges exist between vertices in W; i.e., such that $(u, v) \notin E$. A *vertex cover* is a set of vertices $W \subseteq V$ such that for all edges, at least one endpoint is in W; i.e., for all $(u, v) \in E$ we have $u \in W$ or $v \in W$.

   Here are two decision problems on graphs:

[1] This actually wasn't known until 2002, when Agrawal, Kayal, and Saxena created the AKS primality test. Kayal and Saxena were undergraduates at IIT Kanpur at the time; Agrawal was their advisor.

- INDEPENDENT-SET takes an undirected graph $G = (V, E)$ and integer k and returns YES iff G contains an independent set of size at least k.
- VERTEX-COVER takes an undirected graph $G = (V, E)$ and integer k and returns YES iff G contains a vertex cover of size at most k.

(a) Show that INDEPENDENT-SET $\leq_P$ VERTEX-COVER.

(b) Show that VERTEX-COVER $\leq_P$ INDEPENDENT-SET.

(c) Give polynomial-time vertifiers for INDEPENDENT-SET and VERTEX-COVER.

3. Consider the following decision variant of the Subset Sum problem we saw a few weeks ago. SUBSET-SUM takes n integers $w_1, \ldots, w_n$ and an integer W and outputs YES iff there exists a subset of $\{w_1, \ldots, w_n\}$ that adds up to precisely W.

(a) Give a polynomial-time verifier for SUBSET-SUM.

(b) Show that VERTEX-COVER $\leq_P$ SUBSET-SUM.