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

November 11, 2019

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.

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

1. V is a polynomial-time algorithm that takes two inputs x and w.
2. 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.

Consider this lab a **success** if you complete problem 2 and make progress on problems 3,4.

1. **Transitivity of polynomial-time reductions.** Show the following:

$$\text{If } A \leq_{\text{P}} B \text{ and } B \leq_{\text{P}} C \text{ then } A \leq_{\text{P}} C.$$

2. **Verifier Debugging.** Recall the THREE-COLORING problem: 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*.

   Consider the following verifier for THREE-COLORING. The witness we request is a valid three coloring of the undirected graph $G = (V, E)$, which is specified as a list of two-digit binary strings $w = w_1 w_2 \ldots w_k$ where we interpret

$$w_i = \begin{cases} 00, & \text{vertex } i \text{ is colored BLUE} \\ 01, & \text{vertex } i \text{ is colored GREEN} \\ 10, & \text{vertex } i \text{ is colored RED} \end{cases}$$

```
threeColoringVerifier(G = (V, E), w)
for each w_i in w
    if w_i = 11
        return NO
    for j from i + 1 to |w|
        if w_i = w_j and (i, j) ∈ E
            return NO
return YES
```

This verifier is not quite right.

Give an example witness w and graph G which is *not* three-colorable, such that

$$\textsc{threeColoringVerifier}(G, w) = \textsc{yes}$$

3. Rewrite $\textsc{threeColoringVerifier}$ so that it is a valid verifier for $\textsc{Three-Coloring}$.

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

   (a) $\textsc{Independent-Set}$.

   (b) $\textsc{Vertex-Cover}$.

   (c) $\textsc{Sat}$. The input for $\textsc{Sat}$ 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 $\textsc{yes}$ iff there exists a truth assignment that *satisfies* every clause.

   (d) $\textsc{Factoring}$. Given numbers n, k written in binary, output $\textsc{yes}$ iff n is divisible by d for some $1 < d \leq k$.

   (e) $\textsc{Not-Factoring}$. Given numbers n, k written in binary, output $\textsc{yes}$ iff n is **NOT** divisible by d for any $1 < d \leq k$.

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

   $\textsc{Primes}$: Given a number n written in binary, output $\textsc{yes}$ iff n is a prime number.

5. Consider the following decision variant of the Subset Sum problem we saw a few weeks ago. $\textsc{Subset-Sum}$ takes n integers $w_1, \ldots, w_n$ and an integer W and outputs $\textsc{yes}$ iff there exists a subset of $S = \{1, \ldots, n\}$ such that $\sum_{i \in S} w_i = W$.

   (a) Give a polynomial-time verifier for $\textsc{Subset-Sum}$.

   (b) Show that $\textsc{Vertex-Cover} \leq_{\mathrm{P}} \textsc{Subset-Sum}$.

[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.