CS5311 Quiz 2 Solution (Spring 2002)

1. Do the following problems:

   (a) **[15 points]** Show that there are languages not in **P**.

   Answer: Languages that can be recognized by Turing machines are r.e. Therefore, if we can find a language that is not r.e., this language cannot be recognized by any Turing machine, and, as a result, cannot be in **P**. We have seen so many such languages. EMPTY, TOTAL, ONE-TO-ONE are a few examples.

   We can also look at the problem from a different angle. All languages in **P** are computable because their Turing machine programs always halt and return true or false. Therefore, a r.e. but not recursive language would serve our purpose. The halting problem and the set K are good examples. ■

   (b) **[5 points]** Does this imply $\mathbf{P} \neq \mathbf{NP}$? Answer the question first and provide an elaboration. You will receive **no** credit if you do not provide an elaboration/proof.

   Answer: This result does not imply $\mathbf{P} \neq \mathbf{NP}$ if the language we found is not in **NP**. To show that $\mathbf{P} \neq \mathbf{NP}$, the language used in the previous problem must be in **NP**. However, we do no know if such language exists. ■

2. Do the following problems:

   (a) **[5 points]** Define **PSPACE**-hard.

   Answer: A language L is **PSPACE**-hard if for every **PSPACE** language Q we have $Q \leq_{\mathrm{P}} L$, where $\leq_{\mathrm{P}}$ is polynomial time reduction. Some authors may use log-space reduction. Note that the use of polynomial-space reduction does not make much sense because it may require exponential time. ■

   (b) **[15 points]** What is the relationship between the set of **PSPACE**-hard problems and the set of **NP**-hard problems? State your claim first and prove it. You will receive **no** credit if you only provide a claim without a proof.

   Answer: A **PSPACE**-hard language is also a **NP**-hard language. Since $\mathbf{NP} \subseteq \mathbf{NPSPACE}$ and $\mathbf{NPSPACE} = \mathbf{PSPACE}$, we have $\mathbf{NP} \subseteq \mathbf{PSPACE}$. Let L be a **PSPACE**-hard language. Then, every language in **PSPACE** reduces to L in polynomial time. Since $\mathbf{NP} \subseteq \mathbf{PSPACE}$, every problem in **NP** also reduces to L in polynomial time, and L is **NP**-hard. ■

3. **[15 points]** Two problems A and B are *polynomial-time equivalent*, denoted as $A \equiv_{\mathrm{P}} B$, if $A \leq_{\mathrm{P}} B$ and $B \leq_{\mathrm{P}} A$. Let VC_n and IS_n be the vertex cover problem of size n and the independent set problem of size n. Let $\mathrm{VC} = \cup_{n \geq 0} \mathrm{VC}_n$ and $\mathrm{IS} = \cup_{n \geq 0} \mathrm{IS}_n$. Show that $\mathrm{VC} \equiv_{\mathrm{P}} \mathrm{IS}$.

   Answer: What we need to do is to show **(1)** an instance in VC reduces to an instance in IS, and **(2)** an instance in IS reduces to an instance in VC. Then, $\mathrm{VC} \equiv_{\mathrm{P}} \mathrm{IS}$.

   First, notice the following important property between vertex cover and independent set:

Given a graph $G = (V, E)$ and $A \subseteq V$, A is a vertex cover of G if and only if $V - A$ is an independent set of G.

This can be proved easily. If A is a vertex cover, then every edge of G is incident to at least one vertex in A. Consider u, $v \in V - A$. If $\overline{uv}$ is an edge, then u or v must be in A since A is a vertex cover. However, this is a contradiction. Therefore, $V - A$ is an independent set. Conversely, suppose $V - A$ is an independent set. Consider an edge $\overline{uv} \in E$. Since $V - A$ is an independent set, u and v cannot be members of $V - A$ at the same time. That is, either u or v is a member of A. Hence, an edge of E is incident to a vertex in A, and A is a vertex cover.

Consider an instance (G, k) of VC (*i.e.*, finding a vertex cover of size k). Since A is a vertex cover if and only if $V - A$ is an independent set of G, an instance of vertex cover of size k reduces to an instance of independent set of size $|V| - k$ in time $O(|V| + |E|)$, and we have VC $\leq_{\mathrm{P}}$ IS. The other direction is the same. Thus, we have IS $\leq_{\mathrm{P}}$ VC and VC $\equiv_{\mathrm{P}}$ IS. ∎

4. **[15 points]** Show that **co-NP** $\subseteq$ **PSPACE**.

<u>Answer:</u> We proved in class that **PSPACE** = **co-PSPACE**. Let $L \in$ **co-NP**. Then, we have the following:

$$\begin{aligned} L \in \textbf{co-NP} &\Rightarrow \bar{L} \in \textbf{NP} \\ &\Rightarrow \bar{L} \in \textbf{PSPACE} \quad (\text{since } \textbf{NP} \subseteq \textbf{PSPACE}) \\ &\Rightarrow L \in \textbf{co-PSPACE} \\ &\Rightarrow L \in \textbf{PSPACE} \quad (\text{since } \textbf{PSPACE} = \textbf{co-PSPACE}) \end{aligned}$$

Therefore, **co-NP** $\subseteq$ **PSPACE** holds. ∎

5. **[15 points]** Show that 2SAT is in **NL**.

<u>Answer:</u> 2SAT was studied in Homework #4. Given a 2SAT Boolean expression ϕ in CNF, a directed graph $G = (V, E)$ is constructed as follows: The vertex set V contains all x's and $\bar{x}$'s, where x is a variable, and E is the edge set that contains arcs $p \rightarrow q$ and $\bar{q} \rightarrow \bar{p}$ if $\bar{p} \vee q$ is a clause. We proved the following important property:

Boolean expression ϕ is not satisfiable if and only if there is a path from a variable x to $\bar{x}$ (and hence a path from $\bar{x}$ to x).

This proposition provides a fast way to show 2SAT is in **co-NL**. Moreover, we also know that **NL**=**co-NL** from a proposition proved in class:

$$\boxed{\textbf{NSPACE}(f(x)) = \textbf{co-NSPACE}(f(x)) \text{ for all functions } f \geq \log_2.}$$

Therefore, as long as we can show 2SAT $\in$ **co-NL**, we have 2SAT $\in$ **NL**. To this end, we have to construct a non-deterministic Turing machine that can reject ϕ in log-space.

A non-deterministic Turing machine can examine the input Boolean expression ϕ, guess an initial variable x and a path starting from x, one vertex at a time, and verify if this path goes from x to $\bar{x}$ and back to x. The verify portion checks **(1)** the edge $p \to q$ is an edge of graph G (*i.e.*, $\bar{p} \vee q$ or $q \vee \bar{p}$ is a clause of ϕ), **(2)** x is the first and the last vertex of the path, and **(3)** $\bar{x}$ is encountered in the path before the end of the path. Since the vertices are guessed one at a time, it only take $O(\log_2 n)$ space to store the *current* vertex, where n is the number of vertices which is at worse twice the number of variables. Therefore, ϕ can be rejected within a space bound $O(\log_2 n)$ and hence 2SAT $\in$ **co-NL**. Since **NL** = **co-NL**, we have 2SAT $\in$ **NL**. ∎

6. **[15 points]** Show that $\mathbf{NL} \subseteq \mathbf{P}$.

   Answer: The following proposition was proved in class as a direct consequence of the Reachability problem:

   > A non-deterministic Turing machine with space bound $O(p(x))$ can be simulated with a deterministic Turing machine with time bound $O(k^{\log_2 n + p(n)})$ for some $k > 1$.

   Let $L \in \mathbf{NL}$. Then, L is accepted by a non-deterministic Turing machine with a space bound $O(\log_2 n)$. Setting $p(x)$ to $\log_2 n$ in the above proposition, L can be accepted by a deterministic Turning machine with time bound $O(k^{2\log_2 n}) = O((k^2)^{\log_2 n})$. Since $a^{\log_2 b} = b^{\log_2 a}$, we have $(k^2)^{\log_2 n} = n^{\log_2 k^2}$ and the deterministic Turing machine that accepts L has a time bound $O(n^{\log_2 k^2})$. As a result, $L \in \mathbf{P}$. Note that k is a constant. ∎