CS331: Homework #10

Due on April 25, 2013 at 11:59pm

Professor Zhang 9:00am

Josh Davis

Problem 1

Let G be a undirected and connected graph.

Prove that $EULERIAN-CYCLE = \{\langle G\rangle : G \text{ has an Eulerian cycle}\}$ is in P.

Proof. To prove that $EULERIAN-CYCLE$ is in P, we will first prove that G has an Eulerian cycle iff every vertex in G has an even number degree. Then we will use this fact to create a TM to show that $EULERIAN-CYCLE$ is in P. We break up the first task into two parts.

Part One If G has an $EULERIAN-CYCLE$, then every vertex has an even degree.

This is easy to prove, for every vertex $v \in G$, in order to visit every edge only once, whenever we visit a vertex on an unused edge, there must be an accompanying unused edge to let us leave the vertex. This also applies to the very first/last vertex in the cycle. One edge must be used to leave it at the start of the cycle and one edge must be used to exit onto it at the end of the cycle.

Thus if a graph G has an $EULERIAN-CYCLE$, then every vertex has an even degree because we need to enter it by an unused edge and leave it by an unused cycle.

Part Two If every vertex has an even degree, then G has an $EULERIAN-CYCLE$.

To show this, we will use a proof by induction.

Base Case We consider the two base cases: one graph with one vertex, and one graph with three vertices.

Consider the base case with just one vertex. Since there are zero edges, zero is an even number and thus a single vertex is an Eulerian cycle to itself.

Next consider the base case with three vertices connected by an edge between all of them. Since there are three edges and three vertices, the graph is in the form of a triangle. If we count the degree for each vertex we can see that the number equals two for all three vertices. Thus it holds for the base case.

Induction Step We now consider a graph G with n vertices and k edges. We wish to show that a graph with n vertices and $k+1$ edges has an Eulerian cycle.

According to the definition, G is connected and each vertex has an even degree. TODO.

Proof Incomplete: I can't for the life of me figure out how to solve this part of the induction. I've found a proof here on page 2: `http://www.cs.mcgill.ca/~ethan/TA/251/eulerian.pdf` but can't understand it enough to rewrite it. I'm just including this as a link to show that the proof is solvable.

Now we will show that $EULERIAN-CYCLE$ is in P by constructing a TM M that runs in polynomial time.

Since we have constructed a TM that runs in polynomial time, $EULERIAN-CYCLE$ is in P and our proof is complete. $\square$

Problem 2

Let $TSP = \{\langle G, K\rangle : G$ has a Hamiltonian cycle of length less than $k\}$.

Prove that TSP is NP-complete.

Proof. To prove that TSP is NP-complete, we will show that $HAMCYCLE \leq_p TSP$. This will show that TSP is NP-complete based on **Theorem 7.36**.

First we will show that $TPS \in NP$. This is easy because we can easily construct a verifier that given a certificate c, we just start at the first vertex in c and iterate over it, adding up the lengths until there are no more vertices in c. If this is less than k, then we accept, else we reject. Thus since this is polynomially verifiable, $TSP \in NP$.

Now to prove that TSP is NP-complete, we need to show that a problem that is NP-complete can be polynomially reduced to it. We will now show that $HAMCYCLE \leq_p TSP$.

To do the reduction, we need to have a polynomial time computable function. We will construct a TM M that computes the function.
$M =$" On input string $\langle G, s, t\rangle$ where G is a graph and s and t are vertices:

1. Check that the given input is a valid encoding of a Hamilton cycle from s to t.
2. Start with $k = 0$.
3. Iterate over the Hamilton cycle from s to t, add one to k each time.
4. Output $\langle G, k\rangle$ where k is our newly computed value."

This works as a reduction because we are assigning a k value based on a graph that has a Hamiltonian cycle to it. Thus it is reducing $HAMILTON - CYCLE \leq_p TSP$.

Since we know that $HAMILTON - CYCLE$ is NP-complete, we know that TSP is also NP-complete by **Theorem 7.36**. Thus our proof is complete. □

Problem 3

Prove that a language is in $co - NP$ iff it has a polynomial time disqualifier.

Proof. To prove that a language is in $coNP$ iff it has a polynomial time disqualifier, we will break it up and prove it in two parts.

Part One If a language $L \in coNP$, then it has a polynomial time disqualifier.

Suppose L is in $coNP$. This means that its complement, $\overline{L}$ is in NP. Since $\overline{L}$ is in NP, there exists a verifier V that can confirm that a given instance is accepted in polynomial time. Given this verifier, we can now construct a disqualifier D for L to shot that $L \in coNP$.

$D =$" On input w

1. Run the verifier V on input w.
2. If V accepts, *reject*.
3. If V rejects, *accept*."

Since V already runs in polynomial time, we can just use it to construct our disqualifier. Thus it has a polynomial time disqualifier.

Part Two If a language L has a polynomial time disqualifier, then $L \in coNP$.

Assume there is a polynomial time disqualifier D for a language L. This means that we can check that an input w is verifiably *no* for its existance in L.

Since we have a disqualifier, we can tell polynomially which problems aren't in the language $\overline{L}$. By definition, this is what $coNP$ is. Thus we can see that if we have a polynomial time disqualifier for a language L, then $L \in coNP$.

Since we have proven both sides, we have shown that a language is in $coNP$ iff it has a polynomial time disqualifier and our proof is complete. □

Problem 4

Part One Prove that $coNP$ is closed under polynomial-time reductions; that is, if $L_1 \leq_p L_2$ and $L_2 \in coNP$, then $L_1 \in coNP$.

Proof. To prove that $coNP$ is closed under polynomial time reductions, we will construct a new disqualifier for L_1 that runs in polynomial time which would confirm that $L_1 \in coNP$.

Let there be two languages, L_1 and L_2. Assume that $L_1 \leq_p L_2$. This means that there is a polynomial time function f that reduces L_1 to L_2. Also assume that $L_2 \in coNP$. This means there is disqualifier D that runs in polynomial time.

We will now construct a disqualifier D' for L_1 that runs in polynomial time.

$D' =$" On input w

1. Compute $f(w)$.
2. Run D with newly computed value.
3. If D accepts, *accept*.
4. If D rejects, *reject*."

This works because we know that if there is a polynomial time reduction for two languages, that $w \in L_1 \iff f(w) \in L_2$. Thus if the computed word is in L_2, then it must be in L_1 and vice versa.

Thus since we have shown that we can construct a disqualifier for L_1, we have shown that $coNP$ is closed under polynomial time reductions. □

Part Two Prove that if a $coNP$-complete language L is in NP then $coNP = NP$.

Proof. Assume that L is $coNP$-complete as well as in NP. According to the definition of $coNP$-complete, every A in $coNP$ is polynomial time reducible to L.

This means that if a language is in NP and is $coNP$-complete, it can be polynomially reduced to it by all other languages in $coNP$. This also means that verifying L can be done in polynomial time. Since all other languages in $coNP$ can be reduced to L, then we can just construct a new polynomial reduction that reduces to L and then verifies it. Thus we can see that all languages in $coNP$ would then be able to be verified in polynomial time, thus $coNP = NP$ and our proof is complete. □

Note: A very similar idea is used for if a language is NP-complete and is in P, then $P = NP$ in **Theorem 7.35**.

Problem 5

Part One For any language class C, P^C is closed under complementation; that is, $L \in P^C \iff \overline{L} \in P^C$.

Proof. Assume that $L \in P^C$. That means that there is a polynomial time oracle for L. To show that $\overline{L} \in P^C$, we can construct a new polynomial time oracle TM M.

We construct M as follows:

$M =$" On input w

1. Run the oracle and query if w is in L
2. If the oracle responds yes, *reject*
3. If the oracle responds no, *accept*."

This works because we can just query the oracle for L and just do the opposite for the complement of L. Thus for any class C, P^C is closed under complementation and we conclude our proof. □

Part Two For any language class C that is closed under complementation, $C \subseteq coNP \iff C \subseteq NP$.

Proof. We will prove that $C \subseteq coNP \iff C \subseteq NP$ in two parts.

Part One If $C \subseteq coNP$, then $C \subseteq NP$.

Assume there exists a class of languages $C \subseteq coNP$ that is closed under complementation. This means there is a language $L \in C$ and also in $coNP$.

Since $L \in coNP$, this means that $\overline{L} \in NP$. But since C is closed under complementation, this then means that the complement of $\overline{L}$ is in NP. Since the complement of $\overline{L}$ is just L, this mean that $L \in NP$. Thus if C is a class closed under complementation and $C \subseteq coNP$, then $C \subseteq NP$ and our proof is complete.

Part Two If $C \subseteq NP$, then $C \subseteq coNP$.

Assume there exists a class of languages $C \subseteq NP$ that is closed under complementation. This means there is a language $L \in C$ and also in NP.

Since C is closed under complementation, this means that $\overline{L}$ is in NP as well. By definition, this means that $L \in coNP$. Thus if C is closed under complementation and $C \subseteq NP$, then $C \subseteq coNP$ and our proof is complete. □

Part Three Prove $P^P \subseteq P$.

Proof. We will show if a language is in P^P, then that implies it is in P. Which means $P^P \subseteq P$.

Assume that there exists a language $L \in P^P$. This means that L has a TM M that runs in polynomial time and uses an oracle.

We can easily see that to prove membership in P, we just need to construct a new TM such that it runs in polynomial time. This is easy to do since M already runs in polynomial time. We just run M in our new TM which means that $L \in P$. This implies that $P^P \subseteq P$. Thus our proof is complete. □

Part Four Prove that $NP^P \subseteq NP$.

Problem 5 continued on next page...

Proof. We will show if a language is in NP^P, then that implies it is in NP. Which means $NP^P \subseteq NP$.

Assume that there exists a language $L \in NP^P$. This means that L has a nondeterministic TM M that runs in polynomial time and uses an oracle for languages in P.

We can see that to prove that $L \in NP$, we just need to prove that it can be verified in polynomial time. To do so, we can just construct a new verifier V that can run in polynomial time. We just take the certificate and make sure it is valid and then query the oracle which will show whether or not $LinNP$. This will imiply that $NP^P \subseteq P$. Thus our proof is complete. $\square$