CS388T: Answer to Homework 2

Vishvas Vasuki

February 24, 2008

1 Question

Prove that a nonempty finite set is NP complete under polynomial time reductions if and only if P=NP.

1.1 Solution

First, we specify notation and prove two lemmas.

Notation:

- Let L be an arbitrary nonempty finite set of strings.
- We had shown in the answer to the previous assignment that L is accepted by a certain deterministic turing machine in linear time. Let M(L) be that machine.

Lemma 1: If P=NP, L is NP complete under polynomial time reductions.

Proof:

Consider the case where P=NP. Take any language L' in P=NP.

For all other cases, we prove the lemma by reducing L' to L in polynomial time.

P=NP implies that any language which can be decided by a nondeterministic turing machine in polynomial time can be decided by a deterministic turing machine in polynomial time. Let this deterministic turing machine be M(L'). Consider the strings $s \in L$ and $p \notin L$.

Then, consider the polynomial time reduction, $R : L' \to L$, such that $\forall x \in L', R(x) = s$, and $\forall x \notin L', R(x) = p$. This can be performed by altering M(L') to output s or p appropriately. Thus, the reduction is indeed polynomial.

The resulting string, R(x) is decided by M(L) in linear time. Thus, the reduction and M(L), acting together, decide the language L' in polynomial time.

Since this is true of every L', we conclude that if P=NP, M(L) is NP complete under polynomial time reductions.

Lemma 2: If L is NP complete under polynomial time reductions, then P=NP.

Proof:

Consider the case where L is NP complete under polynomial time reductions.

First, we note that L belongs to both P and NP, as the turing machine M(L) exists. We are considering the case where L is NP complete under polynomial time reductions. As $P \subseteq NP$, we note that L is P complete under polynomial time reductions. From these, due to a proposition proved in [1], it follows that "If L is NP complete under polynomial time reductions, then P=NP".

Proof of the main proposition:

As lemmata 1 and 2 are true, the main proposition, "a nonempty finite set is NP complete under polynomial time reductions if and only if P=NP." is true.

2 Question

A linear-time reduction R must produce its output R(x) in $O(|x|)$ steps. Prove that there are no P-complete problems under linear-time reductions.

HINT: Such a problem would be in $TIME(n^k)$ for some fixed $k > 0$.

2.1 Solution

We prove this proposition by reducing its negation to an absurdity.

Let $n = |x|$.

Proposition to disprove: There exists a P-complete problem under linear-time reductions.

Reduction to absurdity:

Suppose that there exists a P-complete problem under linear-time reductions, L.

As $L \in P$, $L \in DTIME(n^{k'})$ for some constant k'. Let k be the minimum constant, for which $L \in DTIME(n^k)$. Due to the application of time hierarchy theorem, we know: $\exists L' | L' \in DTIME(n^{4k}) \wedge L' \notin DTIME(n^k)$. Note that for such an L', there is no linear time reduction to any language in $DTIME(n^k$, because if there were, then L' would belong to $DTIME(n^k)$ too. So, we have shown that L' has no linear time reduction to L.

This contradicts the assumption we started with. Hence, the proposition that there exists a P-complete problem under linear-time reductions is absurd.

Proof of the main proposition: As the negation of the main proposition has been proved to be untrue, the main proposition, "there are no P-complete problems under linear-time reductions." is true.

3 Question

a

Prove that NODE COVER is NP-complete by giving a reduction from INDEPENDENT SET to NODE COVER.

b

Prove that the problem LONGEST PATH is NP-complete. LONGEST PATH is the following problem: given a graph G and an integer k, does G contain a simple path (a path encountering no vertex more than once) with k or more edges?

3.1 Solution

a

Definitions and notations:

The INDEPENDENT SET problem is to find whether an independent set of size k exists. The NODE COVER problem is to find whether a node cover of size k' or less exists.

Let the graph G=(V,E) and the number k be the input provided, where V is the set of all verteces in G and E is the set of all edges in G.

Lemma: An independent set of size k exists if and only if a node cover of size $|V| - k$ exists in G.

Proof: Suppose that an indpependent set I of size k exists. Then, there are no edges between any elements of I. So, every edge in E has one or both endpoints in G-I. Hence, G-I is a node cover of G. Hence, if an independent set of size k exists, then a node cover of size $|V| - k$ exists in G.

Suppose that a node cover N of size $|V| - k$ exists in G. Then, every edge in E has one or both end points in N. This means that any edge in E can have at most one endpoint in the set V-N. In other words, V-N is an independent set. Thus, if a node cover N of size $|V| - k$ exists in G, then an independent set of size k exists in G.

A polynomial time, log-space reduction, R, from INDEPENDENT SET to NODE COVER:

Suppose that the input consists of a representation of the graph G, and the number k. The image of this input under the reduction R will simply consist of the graph G and the number $|V| - k$.

Proof of time and space complexity: This transformation can be done in linear time by a turing machine, without having to use any space in the work-tapes. Hence, the reduction is compelted in polynomial time, and while using log space.

Proof of correctness: Our lemma proves that the input x will be accepted by a turing machine solving INDEPENDENT SET if and only if R(x) will be accepted by a turing machine solving NODE COVER.

Proof of NP completeness: Thus, we can reduce any instance of INDEPENDENT SET to NODE COVER in polynomial time and in log space. As NODE COVER is NP complete, INDEPENDENT SET is in NP. Also, INDEPENDENT SET is atleast as hard as NODE COVER. Furthermore, we can conclude that NODE COVER is NP complete.

b

We want to prove that the problem LONGEST PATH is NP-complete.

LONGEST PATH is the following problem: given a graph G and an integer k, does G contain a simple path (a path encountering no vertex more than once) with k or more edges?

HAMILTONIAN PATH is the following problem: Does a graph G, does it contain a Hamiltonian path (a path which visits each vertex in V exactly once).

Lemma 1: LONGEST PATH is atleast as hard as HAMILTONIAN PATH.

Proof: We observe that the length of a Hamiltonian path is always exactly $|V| - 1$. Thus, HAMILTONIAN PATH is a special case of LONGEST PATH, where the input is the graph G and number $|V| - 1$. Hence, LONGEST PATH is atleast as hard as HAMILTONIAN PATH.

Lemma 2a: A simple path of length k exists if and only if some subgraph H of G, with k+1 verteces, contains a hamiltonian path.

Proof: Suppose that a certain path of length k exists in G. Then, this path itself is a subgraph of G. Hence, if a simple path of length k exists, then a subgraph H of G, with k+1 verteces contains a hamiltonian path.

Suppose that a subgraph H of G, with k+1 verteces contains a hamiltonian path. Then, clearly, this path is also present in G. Hence, a simple path of length k exists if a subgraph H of G, with k+1 verteces contains a hamiltonian path.

From the above implications, we have the proof for the lemma.

Lemma 2b: LONGEST PATH is in NP.

Consider a non deterministic turing machine, M, with the inputs graph G and integer k. First, M nondeterministically outputs a subgraph of length k on one of its tape. Different computational paths of M will end up with different subgraphs. This operation is done in time polynomial in the lenth of the input.

Then, in each computational path, an instance of HAMILTONIAN PATH is run, with the corresponding subgraph, H. From [1], we know that HAMILTONIAN PATH is in NP. Hence, this part of the computation is completed in polynomial time.

Further, due to Lemma 2a, we know that this turing machine correctly solves LONGEST PATH. Hence, we have proved by construction that LONGEST PATH is in NP.

Proof of the final proposition: In [1], HAMILTONIAN PATH is proved to be NP complete. But, we have proved that LONGEST PATH is atleast as hard as HAMILTONIAN PATH, and that LONGEST PATH is in NP.

Hence, by the definition of completeness, LONGEST PATH is NP complete.

References

[1] Christos H. Papadimitriou. *Computational Complexity.* Addison Wesley, November 1993.