Class Note #22
Date: 04/05/2006

[Overall Information]
In today’s class, we discussed the reduction from INDEPENDENT SET to VERTEX COVER. Two new problems were introduced: the SAT problem and the 3-SAT problem. The definition of NP-hard and NP-Completeness were given and we saw Cook's Theorem, that 3-SAT is NP-complete.

[Announcements]
1. The homework#5 was assigned today. Most questions of this homework set are related to NP and reduction. Only one question (the Viterbi algorithm) lies in the scope of dynamic programming.
2. The coming quiz is scheduled on 04/12/2006 starting from 8:30. Make sure to show up on time for the quiz. It will cover NP, reductions, and completeness. The Viterbi algorithm will not be on the quiz.
3. There will be a distinguished lecture tomorrow given by Dr. Gordon Bell of Microsoft Research. It is about a quest to chronicle a person's life by encoding every aspect of one's communications with people and machines. Interested students are encouraged to attend it. The time and location is 3:00-4:30pm, at SAL101.
At the beginning of today’s lecture, we reviewed what we studied about reduction in last lecture. Before facing more difficult and challenging problems, another comparatively easy reduction was studied. That is “INDEPENDENT SET \leq_p VERTEX COVER”. The input and output as well as the real-life application of those two problems were recalled.

What we need to do is to come up with a polynomial time algorithm, which converts graph G and number k into G’ and k’ (may or may not be the same), so that if G has/hasn’t an independent set of size at least k, then G’ has/hasn’t a vertex cover of size at most k’.
A specific 7-node graph was given as an example. Easily, we found an independent set with size 3 and a vertex cover with size 4 in the graph. It seems like that the complementary set of this independent set is the vertex cover we found. Is it always the case?
3. The following lemma states: $S$ (a set of nodes) is an independent set if and only if $V \setminus S$ is a vertex cover.

Using the definition of independent set and vertex cover, the proof is relatively easy. If $S$ is an independent set, then for each edge, at most one endpoint is in $S$. Thus, for each edge, at least one endpoint is in the complement of $S$ since there are exactly two endpoints for each edge. This means exactly that the complement of $S$ is a vertex cover.

The other direction is similar: if $S$ is a vertex cover, then each edge has at least one endpoint in $S$, so no edge has both endpoint in the complement of $S$, and the complement is an independent set.

4. The directly following conclusion is that the largest size of independent set plus the minimal size of vertex cover in the same graph is the total number of graph nodes $n$. 
So what the reduction does is to take \((G, k)\) as input and then set \(G' = G, k' = n - k\). The correctness of this reduction can be proved with the help of the above lemma.

It is also trivial that the reduction algorithm runs in polynomial time (constant time actually, since there is only one subtraction involved).

Thus we have proved that “INDEPENDENT SET \(\leq_p\) VERTEX COVER”. In other words, the VERTEX COVER problem is at least as difficult as the INDEPENDENT SET problem. Going backward through exactly the same thing, we can also prove “VERTEX COVER \(\leq_p\) INDEPENDENT SET”.

Afterwards, the class moved to the definitions of NP-hardness and NP-completeness.

We say a problem X is NP-hard if X is at least as difficult as every
X is NP-Complete means: (1) it is NP-hard; (2) it belongs to NP. Or in other way, we can say X is NP-Complete when it is one of the hardest problems in NP, but no harder.

“Do NP-Complete problems exist?” is the first topic we need to verify.

As we can see P (and thus NP) contains infinite amount of problems (“Is the input the number 1? Is the input the number 2 a number? Is it 3 …?). Thus, there is no reason a priori to believe that there couldn't be more and more and more difficult problems in NP, just like there are larger and larger and larger numbers. For analogy, there is no such thing as “a largest number”. Thus, the existence of NP-complete problems is not obvious a priori.

**Definition (NP-hard, NP-complete):**

A problem X is **NP-hard** if $Y \leq \text{X}$ for every problem $Y \in \text{NP}$. (X is at least as difficult as every other problem in NP.)

A problem X is **NP-complete** if X is NP-hard, and X is NP. (X is among the hardest problems in NP, but no harder.)

**Question:** Do NP-complete problems exist?

Notice: P (and thus NP) contain infinitely many languages (for instance, each language $L = \{x \mid i \leq x \leq i \}$ for just one string $i$). A priori, we cannot rule out that there are problems $x_1, x_2, \ldots$ that $x_1 \leq x_2 \leq x_3 \leq \ldots$ and each $x_{i+1}$ is strictly more difficult than $x_i$. 
6. The existence was proved by Cook/Levin in 1972, for the following problem SAT. The input is a logic formula $\Phi$ with Boolean variables $x_1, x_2, \ldots, x_n$. The goal is to answer “is there an assignment of $x_1, x_2, \ldots, x_n$ values (true/false), which makes $\Phi$ true?”

Suppose a true/false assignment of Boolean variables is given (as a proof), then we can “efficiently” verify the answer, so SAT belongs to NP. But for all we know, coming up with a new solution is very “difficult”.

7. To prove that SAT is NP-hard, we need to show that every single problem in NP (and there are infinitely many) can be reduced to SAT. This is where the most difficult part of proving “SAT is NP-Complete”
lies. We will see an outline of how to prove that next week.

However, once we have proved that SAT is NP-hard, further NP-completeness proofs become a lot easier. Because “≤ₚ” is a transitive relation, we can prove a problem X is NP-hard just by showing SAT ≤ₚ X.

8. The SAT problem is a bit messy to work with, so we also defined the restricted version 3-SAT (also known as 3-CNF-SAT). In the 3-SAT problem, the formula consists of the conjunction of m clauses. Each clause is the “OR” of at most three literals (“xₖ” or “not xₖ”), where xₖ is a Boolean variable. Since a 3-SAT formula can be viewed as a special case of a general SAT formula, we have 3-SAT ≤ₚ SAT with a
“nothing needs to be done” reduction (but we already knew that, because 3-SAT is in NP, and every problem in NP can be reduced to SAT by Cook's theorem). More interestingly, though, one can also show that SAT \( \leq_p \) 3-SAT, so 3-SAT is in fact NP-complete.

9. As a general outline of NP-Completeness proofs:

The first step (usually an easy step) is to show that problem X belongs to NP.

The second step is to prove Y \( \leq_p \) X for some already known NP-hard problem like SAT, 3-SAT, INDEPENDENT SET, or one of the many others. Notice, the reduction direction in the second step is from an old (already known) problem to new (to be proved to be NP-Complete) problem.
At the end of today’s lecture, we saw the general idea of proving 3-SAT is NP-hard, which is to convert the original formula $\Phi$ to an equivalent CNF $\Phi'$, by distributing the formula, and then using new variables to make the clauses shorter, if necessary.

Outline for NP-completeness proofs:

(i) Show that problem $X \in \text{NP}$ (usually easy - good news part)

(ii) Show that $Y \leq_p X$ for some problem $Y$ that is known to be NP-hard (such as SAT, 3SAT, independent set, ...)

Notice: We must reduce from the old problem to the new problem.