Lecture 9. More Careful Understanding of the Complexity of NP Problems

We can formally define the notion of the "hardest problem in NP", namely, the NP-completeness, and identify the class of NP-complete problems as the class of hardest problems in NP. But it is still possible to give finer analysis of the complexity of NP and related problems. Here we explain some of them.

9.1 The class coNP

Recall that the definition of NP problems. A problem L is in NP if and only if there exist a polynomial $q_L(\cdot)$ and a polynomial-time computable predicate $R_L(\cdot,\cdot)$ that satisfy the following for any $x \in \{0,1\}^*$ (let $\ell = |x|$):

$$\begin{array}{lcl} x \in L \text{ (i.e., } x \text{ is a 'yes' instance)} & \Rightarrow & \exists w : |w| = q_L(\ell)\ [\ R_L(x,w) = 1\ ], \\ x \notin L \text{ (i.e., } x \text{ is a 'no' instance)} & \Rightarrow & \forall w : |w| = q_L(\ell)\ [\ R_L(x,w) = 0\ ], \end{array}$$

where by "$x \in L$" we mean that x is a 'yes' instance, and by "$x \notin L$" we mean that x is a 'no' instance.

Note that this yes/no checking system is not symmetric. That is, for every 'yes' instance x, we have a witness verifying it, whereas we have *no* witness for every 'no' instance x. Then we can consider a yes/no checking system that is opposite to this system. A problem characterized with this yes/no checking system is called coNP.

We say a problem L is coNP if and only if there exist a polynomial $q_L(\cdot)$ and a polynomial-time computable predicate $R_L(\cdot,\cdot)$ that satisfy the following for any $x \in \{0,1\}^*$ (let $\ell = |x|$):

$$\begin{array}{lcl} x \in L \text{ (i.e., } x \text{ is a 'yes' instance)} & \Rightarrow & \forall w : |w| = q_L(\ell)\ [\ R_L(x,w) = 1\ ], \\ x \notin L \text{ (i.e., } x \text{ is a 'no' instance)} & \Rightarrow & \exists w : |w| = q_L(\ell)\ [\ R_L(x,w) = 0\ ], \end{array}$$

We also use coNP to denote the set of coNP problems.

For example, it is easy to show that the problem PRIME is coNP because we can easily give a witness that a given number x is not prime. (Note that it can be shown that PRIME is also in NP in a quite different way.)

We believe that NP $\neq$ coNP. For example, consider the 3SAT problem. We can easily see that there is a witness (more precisely, there is a way to give a witness) to a 'yes' instance; on the other hand, it is hard to imagine that there is some witness (more precisely, there is some way to give a witness) to a 'no' instance. If we believe that NP $\neq$ coNP; then we have NP $\cap$ coNP $\subsetneq$ NP. On the other hand, it seems unlikely that NP $\cap$ coNP = P. That is, we have the following conjecture.

Conjecture. $\mathrm{P} \subsetneq \mathrm{NP} \cap \mathrm{coNP} \subsetneq \mathrm{NP}$.

If this conjecture is true, then the class NP $\cap$ coNP is the class of problems with intermediate hardness between P problems and NP-complete problems.

9.2 Problems closely related to NP

While NP problems are "decision problems" with the above yes/no checking system, there are some problems that are often regarded (naively) as NP problems. Let us see examples.

search3SAT
Instance: A 3CNF formula F.
Task: Compute a satisfying assignment of F if it exists, and outputs 'no answer' if it does not exist.

Formally, it is a problem for computing a witness of a 'yes' instance. This type of problem is called an *NP search problem*. Note that there may be more than one witnesses, and it is often useful if we can fix one witness. Here is one way to choose one witness out of many witnesses.

searchLexfirst3SAT
Instance: A 3CNF formula F.
Task: Compute the lexicographically the first satisfying assignment of F if it exists, and outputs 'no answer' if it does not exist.

The next example is in general called an *NP optimization problem*.

Max3SAT
Instance: 3CNF formula F.
Question: Compute an assignment that satisfies the maximum number of clauses of F.

From the above naming rule, this should be called searchMax3SAT. But Max3SAT (or MAX3SAT) has been used in the literature, and we simply follow this convention.

The complexity of these problems is closely related to that of NP. In fact, if P = NP, then they all are computable in polynomial-time. But exactly speaking, they are not NP problems. Then how can we characterize these problems. Here as a tool for giving such characterization, we introduce a generalization of the polynomial-time reducibility.

The heart of polynomial-time reductions is to provide a way to design an algorithm for some problem A based on an assumed program for another problem B. We generalize this idea. Consider any problem X, and assume that X is solvable *magically* in $O(1)$ time. Then by using this assumed program for X, we design an algorithm for a given problem Y. If some program `IsY` solves Y in polynomial time by using the assumed program for X as a subroutine, then we say that Y is *polynomial-time solvable relative to* X. In other words, `IsY` is regarded as a program solving Y by asking queries to some *oracle* for X, an imaginary mechanism that can answer the question $y \in X$? instantly for any given y.

For any problem X, we define a class P^X to be the set of problems that are polynomial-time solvable relative to X. Furthermore, we define a class P^{NP} to be the set of problems that are polynomial-time solvable relative to some X in NP. These classes are called *relativized complexity classes*. The following relation is clear from the definition.

Theorem 9.1 If $X \in \mathrm{P}$, then $\mathrm{P}^X = \mathrm{P}$. Furthermore, if $\mathrm{P} = \mathrm{NP}$, then $\mathrm{P}^{\mathrm{NP}} = \mathrm{P}$.

We can show that search3SAT, searchLexfirst3SAT, and Max3SAT are all in P^{NP}, in fact, they are all in $\mathrm{P}^{\mathrm{3SAT}}$.

Homework exercise from Lecture 9

Homework rule: Choose one of the basic problems or the advanced problem, and hand your answer in at the next class (for the basic problem) and at the next[2] class (for the advanced problem). (If you cannot attend the next class, you can submit your answer via email before the class.) You do not have to write a long answer. Usually one page would be enough. I will decide OK or NG, and you can get one point (for a basic problem) and two points (for an advanced problem) by each OK answer.

* For writing an answer, you may use Japanese.

Basic problems

Choose one of the basic problems and hand your answer in at the next class. (If you cannot attend the next class, you can submit your answer via email before the class.) You do not have to write a long answer. Usually one page is enough. I will decide OK or NG, and you can get one point by each OK answer. You can try one of the advanced problems. In this case, you can spend one week to solve the problem, and you need to hand your answer in at the next[2] class. You can get two points by each OK answer to the advanced problem. (You cannot try both basic and advanced problems for each homework.)

* For writing an answer, you may use Japanese.

Basic problems

1. Suppose that NP = coNP and prove formally* that the following Max3SATnum problem is in NP. (* Need to prove based on the definition of the class NP.)

   Max3SATnum
   Instance: 3CNF formula F over n variables, and an integer $k \geq 0$.
   Question: Is k the maximum number of clauses of F that are simultaneously satisfiable?

2. Show a polynomial-time algorithm that solves the following searchHAM problem by using the HAM problem as an oracle.

   searchHAM
   Instance: A graph $G = (V, E)$.
   Task: Compute a Hamiltonial circuit of G.

3. Show a polynomial-time algorithm that solves search3COLOR by using *some* problem in NP as an oracle.

   search3COLOR
   Instance: A graph $G = (V, E)$.
   Task: Compute a proper three coloring of G.

An advanced problem

1. Suppose that NP = coNP and prove formally* that the following Lexfirst3SAT problem is in NP. (* Need to prove based on the definition of the class NP.)

Lexfirst3SAT

Instance: 3CNF formula F over n variables, and an integer i, $1 \leq i \leq n$.

Question: What is the ith bit of the lexicographically the first satisfying assignment of F? (Yes/No $\Leftrightarrow$ 1/0)