CSC 2401 Introduction to Computer Complexity - Fall 2001
Lecture #14, Tuesday, Octoer 30, 2001
Lecturer: Allan Borodin
Scribe: Lap Chi Lau

Lecture #14

NP Search Problem

We know that $L \in$ NP if and only if there exists a polynomial $q(n)$ and a polytime computable predicate Rsuch that $L = \{x \mid \exists y$ with —y— $\leq q(|x|$ and $R(x, y)$ is true$\}$. $R(x, y)$ is a binary predicate that can be decided within some polynomial time, polynomial in $|x|$ and $|y|$ (and since we are only interested in polynomially bounded y, this is the same as saying polynomial in $|x|$). Intuitively speaking, $x \in L$ iff x has a "short" certificate y with $|y| \leq q(|x|)$ and a "efficient" polynomial time algorithm which takes x and y as input and to verify if y is a certificate of x.

Can we always reduce an NP search problem to its associated decision problem? Let L be a NP language with verification predicate R as above. Let $S_{L(R)}$ be the search problem to find a certificate y given $x \in L$. If $x \notin L$, then return "NO". We show that whenever L is NP-complete, $S_{L(R)}$ can be reduced to L. Define $\hat{L} = \{(x, u) \mid \exists v \ R(x, uv)$ where $|uv| \leq q(|x|)\}$. $\hat{L} \in$ NP by definition. Since L is NP-complete, $\hat{L} \leq_p L$. By using $\hat{L}$, we can create increasingly long prefixes of a certificate bit by bit analogous to the way we created the solution item by item in the knapsack problem. Therefore, $S_{L(R)} \leq_{p,T} \hat{L} \leq_p L$.

What if L is not complete?. We can formulate the compositeness language as $L = \{x | x$ is a binary represenation of an integer $(x)_2$ and $\exists y : (y)_2$ is a proper divisor of $(x)_2\}$. In this formulation, the associated search problem is essentially the factoring problem and our belief is that factoring cannot be reduced to testing compositeness!!

The Polynomial Time Hierarchy and Polynomial Space

We know that P $\subseteq$ NP $\subseteq$ PSPACE but neither of these inclusions is known to be proper. It could be the case that P = PSPACE and thus P = NP = PSPACE although we strongly believe that this is not the case. Assuming they are different, we show that there may be a much more refined structure between NP and $PSPACE$. There is an interesting analogy between computablity theory and complexity theory and this motivates the polynmial time hieracrchy we will now develop. In this analogy, P $\sim$ computable, NP $\sim$ semi-computable (recursively enumerable) and PH (polynomial time hierarchy) $\sim$ arithmetic hierarchy.

The polynomial time hierarchy of complexity classes is introduced based on nondeterministic time bounded oracle machines. This hierarchy lies between the class P and the class PSPACE and the class NP is in the first level of the hierarchy. Recall that an oracle is a black box which takes just one step to return the answer of a query. But we are charged the cost of formulating the query. A nondeterministic (deterministic) oracle Turing machine is a nondeterministic (deterministic) Turing machine equipped with an oracle. Let M be an oracle NTM and B be an oracle function. We write $M^B(x)$ to denote the computation of M on input x, using B as an oracle, $t_M^B(x)$ be the shortest time to accept x and $L(M^B)$ to denote the language accepted by M^B.

$L \in P^B$ if there exists a deterministic polynomial time bounded M^B such that $L = L(M^B)$. We define NP^B similarly. Let $\mathcal{C}$ be a time bounded complexity class. We define P^C as follows.
$P^{\mathcal{C}} = \{L \mid \exists B \in C, \exists$ deterministic polynomial time M^B such that $L = L(M^B)\}$. We define $NP^{\mathcal{C}}$ (sometimes written as $NP(\mathcal{C})$) similarly.

For integer $k \in N$, complexity classes Δ_k^p, Σ_k^p and Π_k^p are defined as follows:

- $\Sigma_0^p = \Pi_0^p = \Delta_0^p = P$
- $\Sigma_{k+1}^p = NP(\Sigma_k^p)$
- $\Pi_k^p = co - \Sigma_k^p$
- $\Delta_{k+1}^p = P(\Sigma_k^p)$

By the definition, $\Sigma_1^p = NP(\Sigma_0^p) = NP(P) = NP$ and $\Pi_1^p =$ co-NP. The class PH is defined to be the union of the Σ_k^p over all $k \geq 0$.

Finally, we note (as we will show) that
$\Sigma_k^P \cup \Pi_k^P \subseteq \Delta_{k+1}^P \subseteq \Sigma_{k+1}^P \cup \Pi_{k+1}^P \subseteq PSPACE$