

Assignment 2

COMP 531: Theory of Computation (Winter 06)

Due Feb 16 (Thu), before class

Instructions. Follow the instructions from the first assignment. If you think there's an error in some problem, please let me know asap (navin@cs.mcgill.ca). All problems are worth 10 points.

Problem 1. Show that if $\text{NP} \subseteq \text{TIME}(n^{\log n})$ then $\text{PH} \subseteq \bigcup_{k \geq 0} \text{TIME}(n^{\log^k n})$.

Problem 2. Prove that if $\text{NP} \subseteq \text{P}/\log$ then $\text{NP} = \text{P}$.

Problem 3. VC-dimension (named after Vapnik and Chervonenkis) is an important concept in many areas, in particular in Learning Theory. It gives information about the difficulty of learning a given concept. Given a collection $\mathcal{S} = \{S_1, \dots, S_m\}$ of subsets of a finite set U , the VC-dimension of \mathcal{S} is the size of the largest set $X \subseteq U$ such that for every $X' \subseteq X$, there is an i for which $S_i \cap X = X'$ (we say that X is shattered by \mathcal{S}). A boolean circuit C that computes a function $f : \{0, 1\}^m \times \{0, 1\}^n \rightarrow \{0, 1\}$ succinctly represents a collection \mathcal{S} of 2^m subsets of $U = \{0, 1\}^n$ as follows: the set S_i consists of exactly those elements x for which $C(i, x) = 1$. Finally, the language VC-DIMENSION is the set of pairs (C, k) for which C represents a collection of subsets \mathcal{S} whose VC-dimension is at least k .

What is the smallest class in PH in which you can put VC-DIMENSION? Prove your answer. This problem asks only for the proof of membership in a class but not for a completeness proof.

Problem 4. Show that any language decided by a k -tape nondeterministic TM in time $t(n)$ can be decided by a 2-tape nondeterministic TM also within time $t(n)$.

Remark. For deterministic TMs, it is known that a language decidable by a k -tape deterministic TM in time $t(n)$, where $t(\cdot)$ is time-constructible, can also be decided by a 2-tape DTM within time $O(t(n) \log t(n))$ (this is tricky) no better simulation seems to be known. The result of this problem enabled us to prove a tighter time hierarchy theorem for nondeterministic time than for deterministic time.

Problem 5. For any positive integer a and time-constructible function t let $\Sigma_a \text{TIME}(t)$ denote the class of languages computable by TMs running in time t with at most $a - 1$ alternations starting in a state labeled with existential quantifier. Define Π_a similarly. For example, $\Sigma_1 \text{TIME}(\text{poly}) = \text{NP}$ and $\Pi_1 \text{TIME}(\text{poly}) = \text{coNP}$. Show that

$$\Sigma_a \text{TIME}(t) \not\subseteq \Pi_a \text{TIME}(o(t)).$$

$$\Sigma_a \text{TIME}(o(t)) \subsetneq \Sigma_{a+1} \text{TIME}(t)$$

Problem 6. Prove that there is an oracle C such that $\text{NP}^C \neq \text{coNP}^C$.

Problem 7. Suppose that A and B are two oracles. One of them is an oracle for QBF and the other one is for some arbitrary unknown language. We don't know which one of A and B is the oracle for QBF. Give an algorithm that has access to both A and B and that is guaranteed to solve QBF in polynomial time.

Problem 8. Read the discussion of *padding* from Section 2.3 (page 33) in the text. This technique may be useful in solving the following problems but is not essential.

Show that

1. If $L = P$ then $PSPACE = EXP$.
2. $P \neq SPACE(n^{100})$.

Problem 9. In class we proved deterministic time hierarchy and space hierarchy theorems and a nondeterministic time hierarchy theorem. State and prove a *nondeterministic space hierarchy theorem*. Make it as *tight* as possible.

You may make use of the results that we proved in class; if you do, please clearly refer to the result.

To see what “tight” means in this context recall that the deterministic time hierarchy theorem said $TIME(o(\frac{t(n)}{\log t(n)})) \subsetneq TIME(t(n))$ and the deterministic space hierarchy theorem said that $SPACE(o(s(n))) \subsetneq SPACE(s(n))$. So here the space hierarchy theorem is tighter than the time hierarchy theorem.

Note. The following problems are not to be handed in. But please do attempt them.

Problem 10. [Not to be handed in.] Define

$$\text{MIN-CIRCUIT} = \{ \langle C \rangle \mid \text{No circuit } C' \text{ with fewer gates than } C \text{ computes the same function as } C \}.$$

Given a polytime algorithm for SAT, show how to solve MIN-CIRCUIT in polytime.

As we saw in class, $\text{MIN-CIRCUIT} \in \Pi^2$, but we don't know if $\text{MIN-CIRCUIT} \in \Delta_2 = P^{NP}$. We have a seemingly paradoxical situation: Given a polytime algorithm for SAT we know how to solve MIN-CIRCUIT in polytime. But given an oracle for SAT we don't know how to solve MIN-CIRCUIT in polytime. The oracle and the algorithm have the same input/output behavior. How then can one be more useful than the other.

Problem 11. [Not to be handed in.] It's a remarkable fact that most of the complexity classes that we have studied so far have complete problems. (PH is a possible exception; later in the course we will see some more classes for which we don't know if they have complete problems.) What's wrong with the following argument claiming that P cannot have complete problems.

Suppose P has a complete language L . Then $L \in TIME(n^k)$ for some constant k . Since L is P -complete all problems in P reduce to it via logspace reductions. In particular, a language A in say, $L \in TIME(n^{2k+1}) - L \in TIME(n^{2k})$ reduce to L (we know the existence of such problems from the deterministic time hierarchy theorem). But then this implies that $L \in TIME(n^k)$, but that's impossible by the time hierarchy theorem and our assumption that $L \in TIME(n^{2k+1}) - TIME(n^{2k})$.

Problem 12. [Not to be handed in.] Is $TIME(n^3)$ closed under reductions?