Harvard University
Computer Science 121

Problem Set 9

Due Tuesday, December 7, 2010 at 1:20 PM.
Submit a single PDF (lastname+ps9.pdf) of your solutions to cs121+ps9@seas.harvard.edu
Late problem sets may be turned in until NO LATE DAYS at 1:20 PM with a 20% penalty.
See syllabus for collaboration policy.

Name

Problem set by !!! Your Name Here !!!

with collaborator !!! Collaborators' names here !!!!

PROBLEM 1 (5 + 5 points)

(A) Let DOUBLE-SAT $= \{\langle\phi\rangle : \phi$ is a boolean formula with at least two satisfying assignments $\}$. Show that DOUBLE-SAT is NP-complete.

(B) Let k be a natural number, and let $\mathcal{C}$ be a finite collection of finite sets. Then $\mathcal{C}$ is said to be a k-HITTING SET if there is a set H of size at most k that intersects every member of $\mathcal{C}$. That is, a finite collection C of finite sets is a k-HITTING SET if there exists a set H of size at most k such that $S \cap H \neq \emptyset$ for every $S \in \mathcal{C}$. Prove deciding k-HITTING SET is NP-complete. [*Hint*: Reduce from VERTEX COVER.]

PROBLEM 2 (5+5+2 points)

Consider the following two languages:

$$A = \{\langle D\rangle : D \text{ is a DFA and } D \text{ accepts at least one string}\}$$

$$B = \{\langle D_1, D_2, ..., D_k\rangle : \text{Each } D_i \text{ is a DFA and all of the } D_i \text{ accept at least one string in common}\}$$

(A) Show that $A \in \mathrm{P}$.

(B) **THIS PROBLEM HAS BEEN CORRECTED by changing "**NP**-complete" to "**NP**-hard."**

Show that B is NP-hard by giving a reduction from 3-SAT to B.

(C) We can convert an instance of B into an instance of A by applying the product construction (See p.45-46 of *Sipser*) $k-1$ times in succession. Does this show that $\mathrm{P} = \mathrm{NP}$? Why or why not?

PROBLEM 3 (5+2 points)

Recall that Co-NP $= \{L : \overline{L} \in \text{NP}\}$. It is unknown whether or not NP $=$ Co-NP. Note that NP $=$ Co-NP if and only if NP is closed under complement.

(A) Prove that if NP $\neq$ Co-NP, then P $\neq$ NP. (Aside: Nonetheless, we can't rule out the possibility that NP $=$ Co-NP and yet P $\neq$ NP.)

(B) To prove that NP $=$ Co-NP, it would suffice to show that for every $L \in$ NP, $\overline{L} \in$ NP. Suppose $L \in$ NP. Then there exists a nondeterministic Turing machine M that decides L in polynomial time. Consider the new Turing machine M', which is identical to M except that its accept and reject states are reversed.

What language does M' decide? Explain *briefly*.

PROBLEM 4 (5+5+10 points)

A *search problem* is a mapping $S : \Sigma^* \to P(\Delta^*)$ from strings ("instances") to sets of strings ("valid solutions"). An algorithm M *solves* a search problem S if for every input $x \in \Sigma^*$ such that $S(x) \neq \emptyset$, M outputs some solution in $S(x)$.

An NP *search problem* is one for which there exists a polynomial p and a polynomial-time algorithm V such that for every x and y

1. $y \in S(x) \Rightarrow |y| \leq p(|x|)$, and
2. $y \in S(x) \Leftrightarrow V$ accepts $\langle x, y \rangle$.

Informally, these conditions say that valid solutions are short and can be verified efficiently.

(A) Show that the SAT-SEARCH problem "given a satisfiable boolean formula φ, find a satisfying assignment" is an NP search problem

(B) Prove that SAT $\in$ P if and only if the SAT-SEARCH problem can be solved in polynomial time.

(C) Prove that P $=$ NP if and only if every NP search problem can be solved in polynomial time.

PROBLEM 5 (BONUS +1 points)

A language is in $\text{NST}(S, T)$ if it is accepted by a nondeterministic Turing Machine that runs simultaneously in space $S(n)$ and time $T(n)$. Show that $\text{NST}(S(n), T(n)) \subseteq \text{DSPACE}(S(n) \log T(n))$. Then show that actually $\text{NST}(S(n), T(n)) \subseteq \text{DSPACE}(S(n) \log(T(n)/S(n)))$ (an improvement, for example, if $T(n) = S(n) \log S(n)$).