Homework #5: P and NP
Due: see Canvas

This is our last regular homework, concerning Chapter 7 (TIME, P and NP). We will possibly have some makeup or review problems concerning the any later material. As usual, you don't have to do all the problems, see Canvas for details.

Written Problems

Problem 1 Let 4COLOR $= \{\langle G\rangle : \text{graph } G \text{ is colorable with 4 colors}\}$. Describe a polytime function f reducing 4COLOR to CNFSAT, so that if G has n vertices and m edges, then $f(\langle G\rangle) = \langle\phi\rangle$, where ϕ is a cnf-formula with $O(n)$ variables and $O(m+n)$ clauses. Give a precise formulas (in terms of n and m) for the number of variables in your formula. Also, for each clause size k used by your formula, give a precise formula for the number of clauses of size k.

Problem 2 Let E3SAT be like 3SAT, except each clause must have exactly three literals, without duplicate literals or constants (True or False). Argue E3SAT is NP-complete.

Problem 3 Suppose there is a constant $c > 0$, so that no deterministic algorithm for 3SAT runs in time $O(2^{cm})$, where m is the number of clauses in the 3cnf-formula. (This is a conjecture, stronger than $P \neq NP$. The hard case is "sparse" formulas, where the number of variables n is a constant fraction of m.) Argue a similar exponential lower bound on the time to decide CLIQUE. (Use c, and the number of vertices V, in your answer.)

Problem 4 A is a language which we use as an oracle below. A could be SAT, or it could be a language chosen by an adversary, trying to trick us. Finish this program:

$M^A =$ "On input x:
1. If $x \notin A$, reject. *// first oracle call*
2. If we cannot parse x as $\langle\phi\rangle$ (a boolean formula ϕ), reject.
3. Let n be the number of boolean variables in ϕ, call them $x_1, x_2, \ldots, x_n$.
4. *// finish this: $O(n)$ more oracle calls ...* "

Your program should have these properties:

(i) M^A runs in polynomial time, and makes $O(n)$ oracle calls.
(ii) If $A =$SAT, then $L(M^A) =$SAT.
(iii) For any choice of A, $L(M^A) \subseteq$SAT.

That is: if $A =$ SAT, your program accepts all satisfiable formulas. But no oracle A can convince your program to accept an unsatisfiable formula. (For this problem you should write out the program, and just briefly argue why it has the three properties.)

Problem 5 Do Problem 7.31 (7.30 in the international edition), about final exam scheduling.

Problem 6 Given an integer $N \geq 2$, let $p(N)$ denote the largest prime factor of N, for example $p(120) = 5$. Let $b_k(N)$ denote the kth bit of N written in binary, where $b_0(N)$ is the least significant bit. For example $b_0(6) = b_3(6) = 0, b_1(6) = b_2(6) = 1$. Define the language FACTORBIT $= \{\langle N, k\rangle : N \geq 2, k \geq 0,\ b_k(p(N)) = 1\}$. We suppose these integers are encoded in binary.

Problem 6(a). Show that FACTORBIT is in NP. (Guess and check what?)

Problem 6(b). Show that FACTORBIT is in co-NP.

Problem 6(c). Argue that if FACTORBIT were in P, then we could factor any positive integer N (given in binary) in polynomial time.

Remark: You may use the AKS primality test, which decides whether an n-bit integer is prime in $O(n^6)$ time. FACTORBIT is one of the few remaining interesting languages in NP$\cap$co-NP, that is not known to be in P. If it is in P, that would be bad news for the RSA cryptosystem.

Problem 7 Suppose V is a *polynomial time verifier* (see pages 293–294). That means there is a polynomial $p(n)$, so that V decides whether to accept the pair $\langle w, c\rangle$ in time at most $p(|w|)$. V is a *verifier* for the language $\{w \in \Sigma^* : V \text{ accepts } \langle w, c\rangle \text{ for some } c \in \Sigma^*\}$. NP is exactly the class of languages with polynomial time verifiers. In our "guess-and-check" framework, you can think of w as the original input, c is the string that we "guess", and V does the "check" step.

Suppose we change the definition, and allow V to use $p(|\langle w, c\rangle|)$ time (where $p(n)$ is still some polynomial). With this modified definition, what class of languages do we get, instead of NP? (The issue is that c could now be much longer than w, and V will still have enough time to read it.)

Problem 8 Let $\phi(\mathbf{x}, \mathbf{y})$ denote a boolean formula with $2n$ boolean variables: $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$. Let EA be the language $\{\langle\phi(\mathbf{x}, \mathbf{y})\rangle : \exists \mathbf{x} \in \{0,1\}^n\ \forall \mathbf{y} \in \{0,1\}^n\ \phi(\mathbf{x}, \mathbf{y})\}$. In English, this says: "there exists some assignment for the $\mathbf{x}$ variables, so that for all assignments of the $\mathbf{y}$ variables, $\phi(\mathbf{x}, \mathbf{y})$ is true." P=NP, show EA is in P. (Note: we do not believe that EA is in NP.)

Remark: In practice, many NP problems are approachable by reducing them to SAT, and then giving the resulting formula to a "SAT solver". If I can figure out a way to turn that into a nice homework problem, I may add one more.