Solutions to Homework I
CS6520
Computational Complexity

February 15, 2005

Problem: Show that deciding if an instance of 2-SAT is satisfiable is in P.
Solution: The idea is eliminate one variable from the system at every step. This can be done as follows:

- If x or $\bar{x}$ occurs as a singleton in some clause, set x accordingly. If both occur, then reject.
- If $\bar{x}$ does not occur in any clause, set x to True (and vice versa).
- Hence x and $\bar{x}$ both occur but only in clauses of size 2. Assume these caluses are $x \vee y_i$ and $\bar{x} \vee z_j$. Replace these clauses by the clauses $y_i \vee z_j$ for every i, j. Eliminate all duplicate clauses.

Since the number of $2-SAT$ clauses in n-variables is $O(n^2)$ and the number of variables reduces by 1 at every step, the running time is $O(n^3)$. To prove correctness, we will show that the system $x \vee y_i$, $\bar{x} \vee z_j$ has a solution iff $y_i \vee z_j$ for all i, j has a solution. The forward direction is easy. Assume that $y_i \vee z_j$ for all i, j has a solution. Then it must be that all y_is are True or all z_js are True. Else there is some i, j so that y_i and z_j are set to False. Then the clause $y_i \vee z_j$ is unsatisfied. We now set x_i accordingly. $\Box$

Another solution to this Problem (see Papadimitriou's book) is to construct a graph. This in fact shows that 2-SAT is NL-complete.

Problem: Show that deciding if there is an assignment that satisfies at least K out of M clauses is NP-complete.
Solution: Given graph $G(V, E)$ with n vertices and m edges, we define an instance of 2-SAT as follows:

- Add $\bar{x}_i$ fpr $1 \leq i \leq n$ as a clause for every vertex in V.
- Add n copies of $x_i \vee x_j$ for each edge (i, j)

There is an assignment satisfying $nm + n - k$ clauses iff G has a vertex cover of size k. (check this!) □

For a reduction from 3-SAT, see Papadimitriou.

Problem: The class EXP is defined as

$$\text{EXP} = \bigcup_k \text{DTIME}(2^{n^k})$$

Show that NP ⊆ EXP and co-NP ⊆ EXP.
Solution: If $L \in NP$, there is a deterministic polynomial time verifier V such that for every $x \in L$, there is y of size $|x|^k$ such that $V(x, y)$ accepts. We construct a machine V' that runs $V(x, y)$ on all possible ys. V' accepts if there is some y where V accepts and rejects otherwise. V' runs in time 2^{n^k}. This shows $NP \subseteq EXP$. Since EXP is closed under complement so $co - NP \subseteq EXP$. □

Problem: Show that if P = NP, there is a polynomial time algorithm to find a satisfying assignment to a 3-SAT formula if such an assignment exists.
Solution: If $P = NP$ there exists a polynomial time algorithm A to decide if a $3 - SAT$ is satisfiable. Assume we have a satisfiable $3 - SAT$ instance ϕ. To find a solution, we try $x_1 = 0$ and see if the resulting 3-SAT ϕ_0 is also satisfiable. If it is not, we set $x_1 = 1$ (this instance ϕ_1 has to be satisfiable). Repeat for remaining variables. □

Does such an equivalence hold for every NP-complete problem?

Problem: Let BIPARTITE denote the language of all (undirected) graphs which are bipartite. Show that BIPARTITE ∈ NL.
Solution: We describe an NL machine for $BIPA\bar{R}TITE$. Recall that G is non-bipartite iff it contains an odd cycle.

We keep a counter k for path length. We guess a start vertex v and store it. For $k \leq n$ we guess the next vertex u on the path. If it happens that $u = v$ and k is odd, we accept since the graph must contain an odd cycle.

Now since NL=co-NL, we have $BIPARTITE \in NL$. □

Problem: A directed graph is *strongly connected* if for every pair of vertices (u, v) there is a directed path from u to v in G. Show that the problem of deciding whether a graph is strongly connected is NL-complete.
Solution: We reduce $STCONN$ to the problem of deciding if a graph is strongly connected. Given $G(V, E)$ with (s, t), we add directed edges from every vertex $v \neq s, t$ to s and from t to v. It is easy to show that this graph is strongly connected iff there is a path from s to t in G. $\square$

Problem: Deciding if the following equation has an integer solution is in P.

$$\sum_{i=1}^{n} a_i X_i = b$$

Solution: Compute $g = GCD(a_1, \cdots, a_n)$. Accept if g/b. If $g \not| b$, there is not solution. Else, there exist y_is such that $\sum_i y_i a_i = g$. Hence $X_i = \frac{b}{g} y_i$ is a solution. Finally GCD of n numbers can be computed in P. $\square$

In this question, it is crucial that we are allowed arbitrary integers. If we restrict to positive integers, the problem is again NP-complete. In other words, deciding whether there is an integer solution to the equation

$$\sum_{i=1}^{n} a_i X_i = b \qquad\qquad 0 \leq X_i$$

is NP-complete.

Problem: Show that the following problem is NP-hard. Given a polynomial $P(X_1, \cdots, X_n)$ with integer coefficients, the problem is to decide whether the following equation has an integer solution :

$$P(X_1, \cdots, X_n) = 0$$

Solution: By reduction from knapsack.

$$P(X_1, \cdots, X_n) = (\sum_i a_i X_i - b)^2 + \sum_i X_i^2 - X_i$$

Note that $X_i^2 - X_i$ is 0 at $0, 1$ and positive for all other integers. $\square$

Note that one cannot guess a solution since the size may not be polynomially bounded in the sizes of the coefficients of P. In fact this problem (finding an integer solution to an equation) was Hilbert's tenth problem and it was shown to be undecidable. Adleman and Manders showed that the following is NP-complete: find an integer solution to

$$aX_1^2 + bX_2 + c = 0$$

For a language $L \subseteq \{0,1\}^*$, and a function $f(n)$ (assume that $f(n)$ is computable in time $O(f(n))$), let $L_f \subseteq \{0,1,\#\}^*$ denote the following language :

$$L_f := \{x\#^{f(|x|)} \mid x \in L\}$$

Problem: Suppose that $L \in$ DTIME($f(n)$). Then show that $L_f \in$ DTIME($O(n)$). Show similar results for non-deterministic time classes and deterministic space classes.
Solution: To show $L_f \in DTIME(O(n))$, we first check that the input is of the form $x\#^*$. We then check $x \in L$. We then check that the number of #s is indeed $f(n)$. If so accept. □

Problem: Show that if $f(n)$ is a polynomial function, then $L \in$ P iff $L_f \in$ P.
Solution: If $L \in P$, clearly $L_f \in P$. Assume that L_f in P. Given an input x to test membership in L, we simply pad it with $f(n) = poly(n)$ # symbols and run L_f on it. Since L_f runs in polynomial time in input, it is also polynomial in $|x|$. □

Problem: Show that P $\neq$ DSPACE($O(n)$).
Solution: Assume equality holds. Take a language $L \in DSPACE(O(n^2))$. By taking $f(n) = n^2$, we can get $L_f \in DSPACE(O(n)) = P$. Now by the previous problem we have $L \in P = DSPACE(O(n))$. But this violates the Space hierarchy theorem, since there are languages in $DSPACE(O(n^2))$ which are not in $DSPACE(O(n))$. □

Problem: Define the class NEXP as

$$\text{NEXP} := \cup_k \text{NTIME}(2^{n^k})$$

Prove that if P = NP then EXP = NEXP.
Solution: Assume $P = NP$. Take a language $L \in NTIME(2^{n^k})$. By

taking $f(n) = 2^{n^k}$, we can get $L_f \in NP = P$. By the above arguement, this gives an exponential time deterministic machine for L, hence $NEXP \subseteq EXP$. $\Box$