CS389: Computational Complexity Homework II

Zihao Ye

November 14, 2016

PROBLEM 2.3

Firstly, we prove that if there exists such a vector $\mathbf{x}$ that $A\mathbf{x} = \mathbf{b}$, then there is a solution $\mathbf{x}$ whose representation takes a number of bits that is polynomial in the representation of $A, \mathbf{b}$.

Suppose we use (a, b) to denote a ration number $\frac{a}{b}$, it takes $\log(a) + \log(b)$ bits memory to save this number.

Assume A is a full-rank and its size $n = m$(otherwise, we could reduce some redundant items to make it a full-rank and square($n = m$) one).

The *Cramer's rule* shows us that $x_i = \frac{|A_i|}{|A|}$ where A_i's i-th column is $\mathbf{b}$ and other columns are the same as A.

Consider how many bits it take to represent $|A|(|A_i|)$. Since $|A| = \sum_{\sigma \in S_n} sgn(\sigma) A_\sigma$, and A_σ is the product of n items in the matrix A. According to our denotion (a, b), each A_σ takes a number of bits that is polynomial to the representation of $A, \mathbf{b}$.

$$|A| = \sum_{\sigma \in S_n} sgn(\sigma) A_\sigma, abs(|A|) \leq n! \times \max\{A_\sigma\}$$

Thus it will take no more than $O(n \log n + P(|x|))$ bits to record $|A|$, while $|x|$ is the size of input($A, \mathbf{b}$), it's not difficult to show $n = O(|x|)$.

For the same reason $|A_i|$ could be represented by a number of bits that is polynomial to $|x|$, which shows that there is feasible solution $\mathbf{x}$.

This conclusion implies we could nondeterministically enumerate the possible solutions with polynomial size. And to decide whether it's a correct solution in polynomial time. LINEQ is a NP-problem.

PROBLEM 2.6

(B)

Since (b) implies (a), I only prove the second statement.

Construct a NDTM with five tapes: first one as input, second one as nondeterministic choice, third one as justification, the fourth and fifth one are served as preserving $prev(i)$ and $inputpos(i)$.

Use the same technique in Cook-Levin reduction, record z_i, $inputpos(i)$ and $prev(i)$ only. And z_i will take c bits as memory, $inputpos(i)$ and $prev(i)$ will take $\log(T(n)) = O(\log(n))$ bits as memory respectively.

Firstly we guess a polynomial-length string to represent the computation of M, then simulate this computation, move and modify the cells in fourth(prev) tape and fifth(inputpos) tape as the simulation goes by. And justify whether the movement and transition is valid in the third(justification) tape.

Thus we only need to guess $T(n)(c + O(\log(n)))$ bits which is a polynomial of inputsize n. The justification will take $O(n)$ time since we only need to guarantee three things:

- z_1 encodes the initial snapshot.
- $z_{T(x)}$ encodes the final snapshot.
- $z_{i-1}, z_i, z_{prev(i)}, y_{inputpos(i)}$ must satisfy the relationship imposed by transition function.

If we could lookup items in transition function table in $O(1)$(however it only depends on the TM so it's always $O(1)$ with resect to input size), the total time-complexity will be $O(n)$.

PROBLEM 2.13

(A)

Since each certificate of $x \in L$ correspond to a definite computation(with the help of oblivious UTM). Then we only need to ensure our validation of the nondeterministic choice of computation will not introduce extra variables.

$$x \in L \iff \exists y \in \{0,1\}^{|x|+p(|x|)}.\exists z_1, \cdots, z_{T(|x|)}.\varphi_x(y, z)$$

While the most important portion of $\varphi_x(y, z)$ is to decide whether

$$z_i = next(z_{prev}(i), z_{i-1}, y_{inputsize(i)})$$

according to the transition function.

However z_i consist of a boolean function from z_{i-1} to current state. Each boolean function could be represented as $f : \{0,1\}^l \to \{0,1\}$ and it could be implemented by

$$\bigwedge_{v \in \{0,1\}^l \wedge f(v)=0} C_V(z_1, \cdots, z_l)$$

In which C_V must be a CNF. But the original construction is a DNF:

$$(x_1 \neq y_1) \vee (x_2 \neq y_2) \vee \cdots \vee (x_n \neq y_n) = (\neg x_1 \wedge y_1) \vee (x_1 \wedge \neg y_1) \vee \cdots (\neg x_n \wedge y_n) \vee (x_n \wedge \neg y_n)$$

There are two methods to convert it to a CNF one. Since we don't want to introduce new variables(what we need is a parsimonious reduction). We need to generate a new CNF in the first way:

$$\begin{aligned}(\neg x_1 \vee x_1 \vee \neg x_2 \vee x_2 \vee \cdots \vee \neg x_n \vee x_n) \quad &\wedge \quad (y_1 \vee x_1 \vee \neg x_2 \vee x_2 \vee \cdots \vee \neg x_n \vee x_n) \\ &\wedge \quad (\neg x_1 \vee \neg y_1 \vee \neg x_2 \vee x_2 \vee \cdots \vee \neg x_n \vee x_n) \\ &\wedge \quad \cdots \\ &\wedge \quad (y_1 \vee \neg y_1 \vee \neg y_2 \vee y_2 \vee \cdots \vee \neg y_n \vee y_n)\end{aligned}$$

There are 4^n items in total. Thus $\bigwedge_{v\in\{0,1\}^l \wedge f(v)=0} C_V(z_1, \cdots, z_l)$ could be represented by 8^l items. As input size is bounded by $3c + \log n$, The boolean function's size is a polynomial of input size n.

Thus there is a parsimonious Karp-reduction from an NP-problem to SAT.

(B)

Formula with form $u_1 \vee u_2 \vee u_3 \vee u_4 \vee u_5 \vee u_6$ will be transformed to:

$$(u_1 \vee u_2 \vee v) \wedge (\neg v \vee u_3 \vee w) \wedge (\neg w \vee u_4 \vee x) \wedge (\neg x \vee u_5 \vee u_6)$$

But the number of certificates will increase after the reduction, thus we shouldn't introduce extra variables v, w, x, just use another reduction:

$$(u_1 \vee u_2 \vee u_3) \wedge (\neg u_3 \vee u_4 \vee u_5) \wedge (\neg u_5 \vee u_6 \vee u_6)$$

This reduction keeps the number of certificates, and it's also a polynomial one. Thus we derive a parsimonious reduction from SAT to 3SAT.

PROBLEM 2.22

Since IndSet is a NPC problem, we just need to show there exist a Karp-reduction from IndSet to Combinatorial Auction(we use CA to denote it, similarly hereafter) and CA is actually a NP-problem.

However it's not difficult to construct one: Suppose we need to decide whether their is a IndSet of G with size greater or equal to k. Firstly we regard the edges in G as the items in CA. Then every vertex v in G is a pair $\{\langle S_i, x_i\rangle\}$ in CA, where S_i is actually the set of adjacent edges of v and x_i is exactly 1. We just need to decide whether their is a sell strategy with a revenue of at least k. The whole reduction take no more than $O(|E|\log|E|)$ space which is a polynomial of the input size of IndSet.

It's trivial to show that every sell strategy corresponds to a IndSet of G since no edge is permitted to appear twice as an item for auction, and vice versa. Hence we derive CA is a NP-hard problem.

However, by guessing a choice nondeterministically, we could decide whether it's feasible in P-time. Thus CA is a NPC-problem.

PROBLEM 2.33

$$\Psi = \exists_{x\in\{0,1\}^n} \forall_{y\in\{0,1\}^m} s.t. \varphi(x,y) = 1$$

We have CoNP = NP by the assumption that P = NP. Consider the inner part of Ψ, given x, it's a CoNP-problem(so it's a-NP problem), use Cook-Levin reduction, we derive a SAT formula which could be decided in P-time. Thus we could construct a TM M that takes φ and x as input that decides $\forall_{y\in\{0,1\}^m} s.t. \varphi(x,y) = 1$.

In this case $\Psi = \exists_{x\in\{0,1\}^n}.M(\varphi,x) = 1$, which is a typical NP-problem, use the Cook-Levin reduction again, we derive a SAT formula that could be decided in P-time.

Thus Σ_2SAT $\in P$ by the assumption that $P = NP$.