AMIT YADAV

Amit

Exam 2

April, 07, 2020

(1) (a)

First, the definition of NP is given as:

$L \in NP$ iff $\exists$ a p-time verifier V s.t

$x \in L \Leftrightarrow \exists y, |y| \leq p(|x|)$ s.t. $V(x,y) = 1$.

Now, lets look at the first def. of coNP:

$coNP = \{L \mid \bar{L} \in NP\}$

$\therefore$ $L \in coNP$ iff $\bar{L} \in NP$

$\Leftrightarrow \exists$ a p-time verifier V s.t.

$x \in \bar{L} \Leftrightarrow \exists y, |y| \leq p(|x|)$ s.t $V(x,y) = 1$

$\Leftrightarrow x \in L \Leftrightarrow \neg(\exists y$ s.t $V(x,y)=1)$

$\Leftrightarrow x \in L$ iff $\forall y, |y| \leq p(|x|)$ s.t. $V(x,y) \neq 1$

$\Leftrightarrow x \in L$ iff $\forall y$ $V'(x,y) = 1$

where $V'(z) = \begin{cases} 0 & \text{if } V(z) = 1 \\ 1 & \text{else} \end{cases}$

$\therefore$ Our definition of coNP becomes:

$L \in coNP$ iff $\exists$ a p-time verifier V' s.t

$x \in L \Leftrightarrow \forall y, |y| \leq p(|x|), V'(x,y) = 1$

We can now see that the (ii) def. of coNP is exactly the same.

i.e. 1. V' halts on all inputs (because its p-time)

2. V' always produces 0 or 1 and

3. $x \in L \Leftrightarrow \forall y$ with $|y| \leq p(|x|)$, $V'(x, y) = 1$.

1.(b) <u>To prove</u>: TAUT is coNP-complete.

<u>Sol</u>:

① TAUT is in coNP.

<u>Proof</u>: TAUT = $\{\phi \mid \phi$ is tautology$\}$

$\therefore \phi \in$ TAUT $\Leftrightarrow \phi$ is tautology

$\Leftrightarrow \forall t \quad \phi(t) =$ True (where t is truth assignment of bool. var. of ϕ)

Now, we know that size of $t = |t|$ is poly. size w.r.t. boolean vars of ϕ.

Also, if $\phi(t) =$ True can be checked in p-time.

$\therefore$ TAUT is in coNP by using 1.a.(ii) def of coNP.

3/3

(2) TAUT is coNP-hard.

Let $L \in coNP$ be a language.

We will show a reduction $R : L \leq TAUT$ s.t.

$x \in L$ iff $R(x) \in TAUT$.

We know that $\bar{L} \in NP$ and that SAT is NP-complete. Let R' be the reduction from $\bar{L}$ to SAT.

Way to construct R:

For a given x:

(1) Compute $R'(x)$.

(2) $R(x) = \neg R'(x)$

Claim: $x \in L \Leftrightarrow R(x) \in TAUT$.

Proof

$$x \in \bar{L} \Leftrightarrow R'(x) \in SAT$$

$$\equiv x \in L \Leftrightarrow R'(x) \notin SAT$$

$$\equiv x \in L \Leftrightarrow \neg R'(x) \in TAUT$$

$$\Leftrightarrow R(x) \in TAUT$$

∴ Our claim is correct.

Now, we need to show that the reductions takes log-space.

Since, R' takes log-space and $R(x) = \neg R(x)$, which doesn't take any extra space. ∴ R takes log space.

Also, $R(x)$ is p-time computable because $R'(x)$ is p-time computable and $R(x) = \neg R'(x)$ is also p-time.

$\therefore$ TAUT is coNP-hard.

Hence, TAUT is coNP-complete.

3/3

6/6

(2) To show: Almost Monotone Cirucit SAT is NP-complete.

Sol: (1) It is in NP.

Proof: For a given circuit, guess a truth assignm. of its inputs and check whether the output is 1.

Truth assign. is clearly a small certificate and checking takes p-time w.r.t. #gates.

$\therefore$ It is in NP.

(2) It is NP-hard.

Proof: We will show this by a reduction from NP-complete probem 3-SAT to Almost MC SAT.

For a given 3-SAT formula ϕ, with vars $|X|=n$ we construct a monotone circuit of gates $\alpha = \{\alpha_1, \alpha_2 \dots\}$

(1) $\alpha_1, \dots, \alpha_n = x_1, \dots, x_n$ (input gates)

(2) $\alpha_{n+1}, \dots, \alpha_{2n} = Not(x_1), \dots, Not(x_n)$

(3) For every clause $C_i = (a \vee b \vee c)$, we introduce 2-OR gates

$$\alpha_{2n+2i-1} = OR(\alpha_a, \alpha_b)$$

$$\alpha_{2n+2i} = OR(\alpha_c \; \alpha_{2n+2i-1})$$

where if $a = x_j$, then $\alpha_a = \alpha_j$. If $a = \neg x_j$ then $\alpha_a = \alpha_{j+n}$. IIIly for b and c.

④ Then we introduce $(m-1)$ And gates.

for $i = 1 \ldots m-1$

$$\alpha_{2n+2m+i} = AND(\alpha_{2n+2i}, \alpha_{2n+2m+i-1}).$$

⑤ Our final output will be $\alpha_{2n+2m+m-1}$

Claim : $x \in$ 3-SAT $\Leftrightarrow R(x) \in$ Almost M.C. SAT.

Proof : As per our construction of $R(x)$, our circuit is clearly almost-monotone.
And since, the structure of x in 3-SAT is exactly same as that of $R(x)$, therefore x is satisfiable $\Leftrightarrow R(x)$ is SAT.

Time analysis for $R(x)$:

Time for step ① and ② $= O(|x|)$
step ③ and ④ $= O(|m|)$

$\therefore R(x)$ is p-time computable.

Ok. Would have been illustrative to have a direct transformation CIRCUIT-SAT ≤ am-CIRCUIT-SAT, but strictly speaking this was not asked.

Space analysis:

Our $R(x)$ construction stores only three counters. One for c_i, one for x_i and one for α_i.

$\therefore$ SPACE $= O(\log|m|)$ (size of 3-SAT ϕ).

$\therefore$ Almost monotone circuit SAT is NP-hard and hence, $\boxed{\text{NP-complete}}$

8/8

③ (a) EXACT INDPENDENT SET.

Given $G=(V,E)$ and $k \geq 1$

To find: If size of largest IND. set is exactly k.

Sol: ① We need to check if $\exists$ a independent set of size k.

② For all other indpendent sets, s', $|s'| \leq k$.

Now $\Delta_2^P = P^{NP}$

So, we have an oracle 'A' for NP. in this case?

What would the oracle be

Let M be a D.T.M. on input $G(V,E)$

① Ask 'A' for Indp. set if $\exists$ an independent size k. This is clearly an NP problem, so we get an answer from 'A' in P-time. If A rejects, then reject.

② Ask 'A' if $\exists$ an independent set of size $\geq k+1$. This is INDP. Set problem which we know is in NP. If 'A' answers Yes, reject

You should fix the oracle 'A' to which you are making the queries.

③ Else accept.

Clearly step 1 and 2 take P-time. ∴ whole algo. runs in P-time, given an oracle for NP.

∴ Exact Ind. set $\subseteq P^{NP} = \Delta_2^P$.

3,5/4

Scanned by CamScanner

3.b

To show: If $\Sigma_i^P = \Pi_i^P$ for some $i \geq 1$, then $\forall j > i$, $\Delta_j^P = \Sigma_j^P = \Pi_j^P = \Sigma_i^P$.

sol: we know that $\Sigma_i^P = \underbrace{\exists \forall \exists \forall \dots}_{i\text{-times}} P$

and $\Pi_i^P = \underbrace{\forall \exists \forall \exists \dots}_{i\text{-times}} P$

Now, if $\Sigma_i^P = \Pi_i^P$,

Then

$$\Sigma_{i+1}^P = \exists \Pi_i^P = \exists \Sigma_i^P = \exists \exists \Pi_{i-1}^P = \exists \Pi_{i-1}^P = \Sigma_i^P$$

and $\Pi_{i+1}^P = \forall \Sigma_i^P = \forall \Pi_i^P = \forall \forall \Sigma_{i-1}^P = \forall \Sigma_{i-1}^P = \Pi_i^P$

and since $\Sigma_i^P \subseteq \Delta_i^P \subseteq \Sigma_{i+1}^P$ and $\Sigma_i^P = \Sigma_{i+1}^P$

$\Rightarrow \Delta_i^P = \Sigma_i^P$.

Now, since for a given 'i', if $\Sigma_i^P = \Pi_i^P$ implies

$\Sigma_{i+1}^P = \Pi_{i+1}^P = \Sigma_i^P = \Pi_i^P = \Delta_i^P$

then using induction, we know that

$\forall j > i$, $\Delta_j^P = \Sigma_j^P = \Pi_j^P = \Sigma_i^P$

Hence, PH collapses to a finite level.

The idea of the calculus is correct, but you should also present more of the details.

2,5/4

6/8

(4) <u>To prove</u>: To find if $G = (V, E)$ has a kernel is NP-complete.

<u>Sol</u>: (1) It is in NP.

<u>Proof</u>: A non-det. TM can guess $K \subseteq V$ and check if K is a kernel.

For checking, on a given K and $G = (V, E)$

(1) For all $u, v \in K$, $(u,v) \notin E$ and $(v,u) \in E$

(2) For all $w \in V/K$, $(u, w) \in E$ for some $u \in K$.

Step 1 take time $O(|K|^2)$ and step 2 takes time $O(|V| \cdot |K|)$ which are poly. w.r.t. $|V|$.

$\therefore$ It is in NP.

(2) <u>It is NP-hard.</u>

<u>Proof</u>: We will show a reduction R from SAT to kernel, s.t. $x \in SAT$ iff $G(V, E) = R(x)$ has a kernel.

<u>Steps</u>: For a given $\phi = C_1 \wedge C_2 \wedge \dots \wedge C_m$ with X as set of bool. vars. and C_i as clauses.

(1) For every clause C_i, create three vertices C_{i1}, C_{i2}, C_{i3} and create edges as below:

$$\begin{tikzcd} & C_{i1} \arrow[dr] & \\ C_{i3} \arrow[ur] & & C_{i2} \arrow[ll] \end{tikzcd}$$

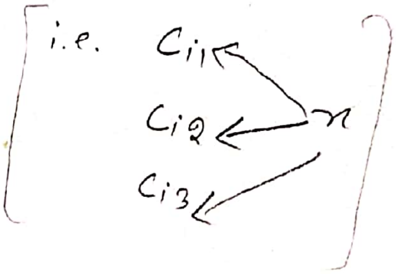
② For every bool. var. $x \in X$, create two vertices x and $\neg x$ and edges as below:

③ For every clause $C_i \in \phi$:

For every variable $x \in X$:

If x appears in C_i, then
$$\begin{cases} (x, C_{i1}) \in E \\ (x, C_{i2}) \in E \\ (x, C_{i3}) \in E \end{cases}$$

i.e.



The resulting graph is our $R(\phi) = G(V, E)$.

<u>Claim</u>: $\phi \in SAT \iff R(\phi)$ has a kernel.

Proof: $(\Rightarrow)$ $\phi \in SAT$

Then Let T is a truth assignt. of X s.t $T \models \phi$.

Then kernel of $R(\phi) = G(V, E)$ can be computed as

$K = \{\}$

For $\forall x \in X$, $K = \begin{cases} K \cup \{x\} & \text{if } T(x) = True \\ K \cup \{\neg x\} & \text{if } T(x) = False \end{cases}$

Now, K is a kernel of $G(V, E)$ because,

① $\forall u, v \in K$, $(u, v) \notin E$ and $(v, u) \in E$ because of step ② in $R(\phi)$ construction.

② If $w \in V/K$, then either $(x, w) \in K$ or $(\neg x, w) \in K$.

This can be seen by step ② and ③ in $R(\phi)$ construction.

$\therefore$ $\phi \in SAT \Rightarrow R(\phi)$ has a kernel.

($\Leftarrow$) $R(\phi)$ has a kernel.

Let K be the kernel, $K \subseteq V$.

① Only one of x and $\neg x$ can be in K.
Because, else, criterion 1. of kernel def. will fail.

② For each clause, one of appearing variables must be true, because of step ③ in $R(\phi)$ construction.
Basically, if a clause is not true, means all variables are false, then atleast one of C_{i1}, C_{i2}, C_{i3} will have no incoming edge from K. which condradicts criterion ② of kernel def.

You could in principle have the variables cover each other. You would thus need an argument to explain why this cannot happen. (which you haven't presented.)

$\therefore$ $\phi \in SAT$

$\therefore$ $\phi \in SAT \iff R(\phi)$ has a kernel.

Time analysis of $R(\phi)$:

Time for step 1 $= O(m)$ $\quad$ $\{m = \#$ clauses$\}$

step 2 $= O(|X|)$

step 3 $= O(m.|X|)$

$\therefore$ $R(\phi)$ is p-time computable w.r.t. # clauses & # vars.

Space analysis:

Since, our algo. needs to store only two counters, one for clauses and one for variables,

$\therefore$ $SPACE = O(\log |X|)$.

$\therefore$ If $G(V,E)$ has a kernel is NP-hard and hence NP-complete.

Construction OK, but the justification for the KERNEL ⇒ SAT direction slightly incomplete.

7/8