AMIT YADAV

Assignment #4

H4.1

(a)

step (i): given problem $\in NP$

Sol: Since it is a special case of SAT, so $\in NP$.

step (ii): It is NP-hard.

We show this by showing 3-SAT $\leq_m$ given problem

Sol: (1) Let 'ϕ' is a formula in 3-SAT form. For every boolean-var (V) in 'ϕ', introduce a new variable V'.

(2) Now, for every clause in 'ϕ', if there is an negative literal ($\neg V$), then replace it by V'.

That is $(a \vee \neg b \vee \neg c)$ becomes $(a \vee b' \vee c')$.

(3) Add the following new clauses to modified 'ϕ';

For every v variable, add

$$(V \text{ or } V') \wedge (\neg V \text{ or } \neg V')$$

(4) In The new generated formula (ϕ')

Every clause either has all three positive literals, or exactly two literals.

(5) Now, run the algorith for given problem over input ϕ' and the valuation of original variables (i.e. of ϕ) will satisfy ϕ. (If satisfiable)

why? Because ϕ and ϕ' are equivalent w.r.t. original vars

Therefore, 3-SAT $\leq_m$ given problem.

(P.S.: Size of new formula ϕ' is $(m + 2n)$ clauses & $2n$ vars. if ϕ has m-clauses and n-variables)

$\therefore$ Given problem is NP-complete.

(b) To show: 1-IN-3-SAT is NP complete

Sol: (i) It belongs to NP.

For n-variables, $\exists$ 2^n guesses.

Each guess can be checked in polynomial time w.r.t. clauses.

$\therefore$ 1-in-3-SAT $\in$ NP.

(ii) It is NP-hard.

We show that 3-SAT $\leq_m$ 1-in-3-SAT.

1 For any clause in ϕ (which is in 3-SAT) $(a \vee b \vee c)$ introduce 6-variables d, e, f, g, h, i.

2 Replace $(a \vee b \vee c)$ clause with

$T \equiv (a \vee d \vee g) \wedge (b \vee e \vee g) \wedge (c \vee f \vee \perp)$

$\wedge (d \vee e \vee h) \wedge (f \vee g \vee i)$

T is satisfiable according to 1-in-3 SAT iff $(a \vee b \vee c)$ is satisfiable acc. to SAT.

Proof

($\Rightarrow$) If T is satisfiable in 1 in 3 SAT $\Rightarrow$ $(a \vee b \vee c)$ is SAT.

(By showing that if $(a \vee b \vee c)$ is UNSAT, then T is UNSAT too)

So, let a, b, c = false (all three)

Then, f must be true, due to clause 3 in T.

then, g, i " " false " " " 5 in T.

then, d, e must be true " " , 1, 2 in T.

This contradicts clause 4 according to 1-in 3 SAT.

($\Leftarrow$) If $(a \vee b \vee c)$ is SAT then T is 1-in-3 SAT.

Proof: Well, for all valuations of a, b, c s.t. $(a \vee b \vee c)$ is true, T is true for some valuations of (d, e, f, g, h, i) and some valuations of (a, b, c).

3 ~~For~~ Therefore, $\forall x$ for SAT-checking, $R(x)$ can be obtained using step 1 & 2 such that $x \in SAT$ iff $R(x) \in$ 1-in 3 SAT.

$\therefore$ 1-in-3 SAT is NP-complete.

H4.2 Domination SET decision problem

(i) Dominating SET $\in NP$.

Given: $G = (V, E)$, and K

Sol: From set V, we can pick at-most K vertices.

$\therefore$ # choices $= O(2^{|V|})$

Using a N-TM, check if the picked set, satisfies the dominating set property:

Time $= O(|E|)$

because each edge will be checked atmost two-times.

$\therefore$ Polynomial time using N-TM.

$\therefore$ Dominating Set $\in NP$.

(ii) Vertex Cover $\leq_m^{Log}$ Dominating Set

Sol: Given $G = (V, E)$, K.

We will find $R(G) = G'$ s.t

$R(G) \in$ D.S. iff $G \in$ V.C.

Steps for $R(G) = (V', E')$

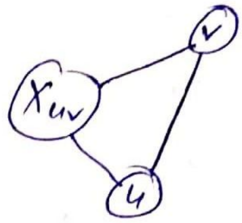
1. For every edge in E, (u,v) introduce a vertex X_{uv} in G', keeping original vertices and edges from G, intact.

2. Introduce the following edges in G':
for all $u, v \in V$,

$(u, X_{uv}) \in E'$

$(v, X_{uv}) \in E'$

That is 

becomes 

(3) We got our G'.

<u>Claim</u>: $R(G) = G' \in D.S.$ iff $G \in V.C.$ with given integer K.

($\Leftarrow$) V.C. of size atmost K in G $\Rightarrow$ D.S. in G' of size atmost k.

<u>Proof</u>:

Same set of V.C. in G works as D.S. in G'.
How?

(i) V.C. ~~covers~~ dominates all the vertice $\in V$ in G' (i.e. the original vertices).
Because any edge from vertex v ~~...~~ such that $v \notin V.C$ is connected to a vertex u s.t. $u \in V.C.$ (by definition, since (u,v) is covered)

(ii) V.C. also dominates new vertices, i.e. for any $u, v \in V$, X_{uv} is connected to ~~...~~ u and v; and one of u and v $\in V.C.$ (because (u,v) is covered)

$\therefore$ D.S of G' $:=$ V.C of G.

($\Leftarrow$) D.S. of size atmost k $\Rightarrow$ V.C. of size atmost k in G
in G'

(Assuming G doesn't have 0-degree vertices. If it has, just replace left side k with k + #(0-degree vertex). because 0-degree vertex don't have any ~~vertices~~ edges.)

Let V.C. = { }

① If $v \in$ D.S s.t. $v \in V$, then keep v in V.C.

② If $v \in$ D.S. s.t. $v \in V'$ but $v \notin V$, ~~then~~ (i.e. X_{uv} vertex), then keep any one of its neighbour in V.C.

Thus obtained set works as V.C. of G.

Because if X_{uv} is dominated by some vertex in G' means (u,v) edge is covered in G.

$\therefore$ We got our $R(G) = G'$ s.t. $R(G) \in$ D.S. iff $G \in$ V.C.

$\therefore$ D.S. is NP-complete.

SPACE ~~Time~~ - analysis for $R(G)$:

It only stores two variable for iterating over $\forall$ edge (u,v). Thus, SPACE = $O(2 \times \log|V|) = O(\log|V|)$

$\because$ ($|V|$ can be stored in $\log|V|$ bits).

H4.3

(i.) Since, A,TM just need to store $k|x|$ to keep counts of the size, ∴ it needs $O(1)$ space.
Also, the number of steps required are size of the input × k.

∴ Time = $O(|x|)$

∴ linear time and $O(1)$ space.

(ii) (a) $U \in space(n)$

U just needs to run M over x, which takes $|x|$ space.

∴ $U \in space(n)$

(b) For every input x of any language $L \in SPACE(n)$, we can find $f(x)$ ~~M accepts x in Space(n)~~

s.t. $x \in L$ iff. U accepts $f(x)$.
→ (Just ~~Run~~ simulate 'M' on x, and output the answer)
Using (i), it takes linear-time and $O(1)$ space to find $f(x)$.

∴ U is linear-complete for SPACE (n).

(iii) Since U is linear complete for space(n)
and if $P =$ space(n)
then U is linear-complete for P also.
(because $U \in P$ and for, $L \in P$, $L \leq^{O(1)}_{linear-time} U$).

(iv) Since if $P =$ space(n)

$\Rightarrow$ U is linear complete for P

$\Rightarrow$ Any any $L \in P$ can be reduced to U in linear-time

$\Rightarrow$ $L \in TIME(U)$

$\Rightarrow$ $L \in TIME(n^r)$ $\forall L \in P$

$\Rightarrow$ $P \in TIME(n^r)$

But what about a language (L') which take (n^{r+1}) time

$L' = TIME(n^{r+1}) \in P \in TIME(n^r)$

$\Rightarrow$ $TIME(n^{r+1}) \in TIME(n^r)$

$\Rightarrow$ Contradiction

$\therefore$ $P \neq SPACE(n)$