To Prove: ① $CNF_2 \in P$ Sandipan Dey   CMSC 651   HW-8

② CNF_3 is NP-Complete

40/30

Proof:

10

① Construct a TM M   /* Assumption, there are m variables $x_1, x_2, \ldots, x_m \in \varphi$ */

$M(\langle\varphi\rangle)$ /* $\varphi = \bigwedge_{j=1}^{n} C_j$ */

linear time in m

1. First ~~find~~ scan the input to find ~~all~~ # of variables in φ ~~(let m variable are there)~~. /* $m \leftarrow$ # of variables, store m on worktape */

~~polynomial~~ polynomial time in m, so polynomial in input

2. ~~If~~ Repeat the following steps $\forall i = 1, \ldots, m$.

2.1. if x_i is ~~not~~ present in φ once (in a single clause) assign $x_i \leftarrow T$,
" $\neg x_i$ " " " " once " " $x_i \leftarrow F$.

2.2. if x_i is present twice in the same clause in φ
if at least one of the occurences are non-negated, $x_i \leftarrow T$
else if both the " " negated (i.e, $\neg x_i$) then
assign $x_i \leftarrow F$

2.3. if x_i is present twice in two different clauses C_1 & C_2
if $x_i \in C_1$ and $x_i \in C_2$, $x_i \leftarrow T$
$\neg x_i \in C_1$ " $\neg x_i \in C_2$, $x_i \leftarrow F$
$x_i \in C_1$ " $\neg x_i \in C_2$, merge the clauses
in the following manner: if $C_1 = x_i \vee \varphi_1$, $C_2 = \bar{x}_i \vee \varphi_2$

/* polynomial in m */

$C_1 \wedge C_2 \xrightarrow[\text{to}]{\text{merge}} \varphi_1 \wedge \varphi_2$ /* it's easy to see that in this case truth ~~val~~ value of x_i does not have any role in satisfiability of φ */

$x_1 \vee \cdots \vee x_r$ $y_1 \vee \cdots \vee y_s$ $x_1 \vee \cdots \vee x_r \vee y_1 \vee \cdots \vee y_s$

merge with OR

3. If φ is simplied to be true (T) accept, ow reject.

M always halts and ~~accepts~~ decides CNF_2 in polynomial time, hence
$CNF_2 \in P$ (Proved).

② (1) $CNF_3 \in NP$

Proof: The following is a verifier V for CNF_3:

$V(\langle \varphi \rangle, c)$ /* C is a set of assignments of the variables in φ */

polynomial time in m, so polynomial time in input

1. Test whether c contains assignments to all the variables in φ.
2. Assign the variables x_i with values from C, $\forall i = 1, \dots, m$ and simplifify. Check if it evaluates to true (T).
3. If both pass, accept, ow, reject.

Alternative proof: Construct a NTM M,

$N(\langle \varphi \rangle)$

polynomial in m, so polynomial in input

1. Nondeterministically guesses the assignment values of every variable x_i, $i = 1, \dots, m$.
2. Assign the variables, simplify and check if φ evaluates to T
3. If yes, accept, ow reject.

(2) $3SAT \leq_m^p CNF_3$ (CNF_3 is NP-hard)

Proof: We have to construct a polynomial time computable function f s.t. $f: \langle \varphi_{3SAT} \rangle \to \langle \varphi_{CNF_3} \rangle$ where φ_{3SAT} is satisfiable iff φ_{CNF_3} is satisfiable. $[f(\langle \varphi_{3SAT} \rangle) = \langle \varphi_{CNF_3} \rangle]$

Construction: let's construct a turing machine F that computes f:

$F(\langle \varphi_{3SAT} \rangle)$

$\varphi \leftarrow \varphi_{3SAT}$; /* copy input onto the worktape */

1. Pick the first variable x_i that appears more than 3 times in the formula φ. If there is none, go to step 4.
2. /* Suppose x_i occurs in n places in φ $(n \geq 3)$ as $\underbrace{(y \vee \varphi_1)}_{C_1} \wedge \underbrace{(y \vee \varphi_2)}_{C_2}, \dots, \underbrace{(y \vee \varphi_n)}_{C_n}$, where each y is either x_i or $\overline{x_i}$. $\forall C_i$ are 3 literal clauses */

$(\forall i)$ remove $(y \vee \varphi_i)$. Introduce n ~~new~~ fresh variables $z_1, z_2, \ldots, z_n$.

Add clauses $(z_1 \vee \underset{2\text{ literals}}{\varphi_1}) \wedge (z_1 \to \bar{z}_2) \wedge (z_2 \vee \varphi_2) \wedge (z_2 \to z_3) \wedge \cdots \wedge (z_n \vee \varphi_n) \wedge (z_n \to z_1)$

$$\varphi \equiv \underbrace{(z_1 \vee \varphi_1)}_{3\text{ literals}} \wedge \underbrace{(\bar{z}_1 \vee z_2 \vee F)}_{3\text{ literals}} \wedge \cdots \wedge \underbrace{(z_n \vee \varphi_n)}_{3\text{ literals}} \wedge (\bar{z}_n \vee z_1 \vee F)$$

/* $z_i = x_i \Leftrightarrow y = x_i$
$= \bar{x}_i \Leftrightarrow y = \bar{x}_i$ */

/* F = falsity

/* $z_1 \to z_2 \to z_3 \to \cdots \to z_n \to z_1$ ensures that all variables get the same truth value in any assignment */

3. Goto step 1.

4. output φ as φ_{CNF_3}.

F runs in polynomial time on input φ_{3SAT} and computes f.

Hence f is a polynomially computable function.

(3) $\langle \varphi_{3SAT} \rangle$ is satisfiable $\Leftrightarrow$ $\langle \varphi_{CNF_3} \rangle$ is satisfiable.

Proof: $(\Rightarrow)$ $\exists$ an assignment
$x_1 = b_1, x_2 = b_2, \ldots, x_n = b_n$ where $b_i \in \{0,1\}$
s.t. $\langle \varphi_{3SAT} \rangle \Big|_{\substack{x_i = b_i \\ \forall i = 1,\ldots,n}} = T$.

If all the variables in φ_{3SAT} appeared ≤ 3 times then $\langle \varphi_{CNF_3} \rangle = \langle \varphi_{3SAT} \rangle$ and this assignment satisfies φ_{CNF_3} as well trivially.

If not, let x_i be the first variable which occurred $n > 3$ places in φ_{3SAT}, hence, by construction the clauses $(y \vee \varphi_i)$ were replaced by $(z_i \vee \varphi_i)$ $\forall i = 1, \ldots, n$ and new n clauses were introduced. $(z_1 \to z_2) \wedge (z_2 \to z_3) \wedge \cdots \wedge (z_n \to z_1)$

Assign z_1 by truth value b_1 if $y = x_1$ or by truth value $\bar{b}_1$ if $y = \bar{x}_1$. All z_i's are to be assigned by same truth value which guarantees that all get the truth value according to x_i was assigned in φ_{3SAT}

This is true $\forall i = 1, \dots, m$.

Hence φ_{CNF_3} is satisfiable.

($\Leftarrow$) $\exists$ an assignment that satisfies φ_{CNF_3}. If # of variables in φ_{CNF_3} and φ_{3SAT} are same then the same assignment trivially satisfies φ_{3SAT} as well. If not, ~~∃~~ find ~~a~~ set of variables ~~z_1, z_2~~ $z_1, z_2, \dots, z_n$ that were assigned the same truth value in φ_{CNF_3} assignment and replace them by a single variable ~~y~~ y ~~and assign it the same~~, this corresponds to a variable in φ_{3SAT}. Assign the variable by the same assignment value. ~~φ_{CNF_3}~~ Repeat this procedure until all the variables in φ_{3SAT} are assigned. Obviously, if φ_{CNF_3} is satisfiable, φ_{3SAT} is also satisfiable.

(1), (2) & (3) $\Rightarrow$ CNF_3 is NP-Complete. ✓

7.36. $P = NP \Rightarrow SAT \in NP = P \Rightarrow SAT \in P$

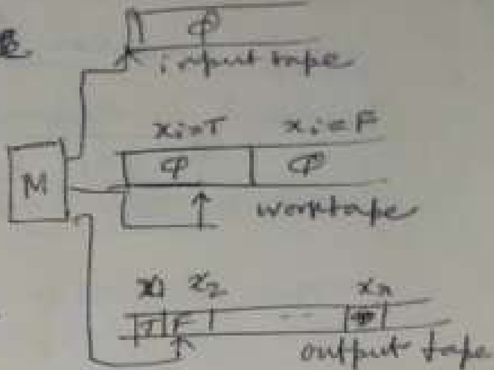
10

~~Let's construct a machine~~ $\Rightarrow \exists$ a turing machine M_{SAT} that decides SAT given a formula φ as input, i.e., M_{SAT} outputs 'yes' iff φ is satisfiable, o/w M_{SAT} outputs 'no' and halts.

M_{SAT} runs in polynomial time and halts.

Let's construct another machine $M_{SATASSIGN}$ that takes as input a formula φ and its variables $x_1, x_2, \dots, x_n$ and outputs a satisfying assignment for φ (if one exists) in polynomial time and halts. The $M_{SATASSIGN}$ runs the desired polynomial time algorithm.

$M_{SATASSIGN}(\langle \varphi \rangle, \langle x_1, x_2, \ldots, x_n \rangle)$



1. Start with copying φ on its worktape

2. $\forall i = 1, \ldots, n$ do the following

   2.1. Create 2 copies of φ on the worktape

   2.2. Substitute $x_i = T$ on the first copy and simplify to get $\varphi_{x_i=T}$
   $x_i = F$ on the second copy ... $\varphi_{x_i=F}$.

   2.3. Run $M(\langle \varphi_{x_i=T} \rangle)$ on input $\varphi_{x_i=T}$, if it outputs 'yes', $\varphi \leftarrow \varphi_{x_i=T}$ and delete the other copy. Write 'T' next on the output tape.
   else run $M(\langle \varphi_{x_i=F} \rangle)$ on input $\varphi_{x_i=F}$ and if it outputs 'yes', $\varphi \leftarrow \varphi_{x_i=F}$ and delete the other copy. Write 'F' next on the output tape
   else ~~outp~~ Write 'No such assignments exists' on the output tape and halt.

   2.4. Goto 2.

Clearly, $M_{SATASSIGN}$ presents a polynomial time algorithm that finds a satisfying assignment of φ assuming $\underline{P = NP}$.

7.39. The 2x3 window used in the proof of Cook-Levin theorem

10 basically ~~checks that~~ ensures that the window is 'legal' iff the top row is a legal configuration and the bottom row ~~follows~~ is also another valid configuration that correctly follows the another.

If ~~we~~ 2x2 window is ~~too~~ used instead, it's possible that the validity of successive configuration will fail, i.e., all 2x2 window configurations may be legal but the top & bottom e.g. Consider the following 2×3 window:

a	q_1	b
q_2	a	q_3

The window is not legal and will be caught by a 2×3 window. Although the top row is a valid configuration, the bottom row is not, since a TM can't at the same time be two different states!

Now consider 2×2 windows instead and let's show that it can't catch the problem, since it only checks the following two windows:

a	q_1
q_2	a

which can be a valid transition: $\delta(a, q_1) \rightarrow (a, q_2, L)$
and

q_1	b
a	q_3

which ~~again~~ can be another valid transition again:
$\delta(a, q_1) \rightarrow (a, q_3, R)$

Both the checks can pass, ~~but the~~ hence the test can fail as can be seen.

7.28. 10 SET-SPLITTING is NP-Complete

Proof: ① SET-SPLITTING $\in$ NP

The following is a verifier for SET-SPLITTING:

$V(\langle S, C, c\rangle)$ /* where $C = \{C_1, C_2, \ldots, C_k\}$, $C_i \subseteq S$, and c be an assignment of colors to the elements of S */

polynomials in size of set

1. Test whether C is a collection of subsets of S.
   /* i.e., $\forall i = 1, \ldots, k$, test if $C_i \subseteq S$ */
2. Test whether the color assignments in c are only blue and red.
3. $\forall i = 1, \ldots, k$, ~~check whether~~ do the following:
   3-1. Check if all the elements of C_i are colored with the same color.
   3-2. If 'yes', output 'No' and Halt, o/w goto step 3
4. output 'yes'.

3SAT $\leq_m^P$ SET-SPLITTING

We have to construct a polynomial time computable function f,

$f: \langle\varphi\rangle \to \langle S, C\rangle$ s.t. φ is satisfiable iff ~~SET SPLITTING~~ no C_i in C has all its elements colored with the same color. ~~the~~ $[f(\varphi) = (S,C)]$.

~~F~~ Let construct ~~for~~ a turing machine F that computes f:

$F(\langle\varphi\rangle)$

1. Constructs the set $S = \{x_i \in \varphi\} \cup \{\bar{x}_i \mid x_i \in \varphi\} \cup \underset{\text{new special element}}{F}$
2. Constructs the subset $S_{x_i} = \{x_i, \bar{x}_i\}$, for each variable $x_i \in \varphi$

   $S_{c_j} = \{x_{j_1}, x_{j_2}, x_{j_3}, \underset{\text{new element}}{F}\}$ for each clause $c_j = \{x_{j_1}, x_{j_2}, x_{j_3}\}$. $C = \underbrace{\bigcup_j}_{\text{over clauses}} S_{c_j} \cup \underbrace{\bigcup_i}_{\text{over variables}} S_{x_i}$

3. output $\langle S, C\rangle$.

The construction is obviously polynomial.

(⇒) Suppose φ is satisfiable. Consider coloring of elements in S. Color the special element F 'red'. ∀ variable x_i that is assigned 'false', color the elements x_i and $\bar{x}_i$ red and blue respectively. For each variable x_i that is assigned 'true', color elements x_i and $\bar{x}_i$ blue and red respectively.

As long as the assignment is satisfying, this leaves no set monochromatic. ∀ variable x_i the set $S_{x_i} = \{x_i, \bar{x}_i\}$ has at least one red and one blue element. ∀ clause c_j, the set C_j has at least one red (F) element and because some literal has a value of true

~~has~~ S_c ~~has~~ must have a blue element as well.

($\Leftarrow$) Suppose $(S, C) \in$ SETSPLITTING. Fix some coloring of S with 2 colors such that every set has at least one element of both colors.

Consider the following assignment to variables of ϕ. $\forall$ var x_i, assign it 'True' if its color differs from that of the special element F. Assign x_i 'False' if its color is same as that of F.

Hence, Each clause c in ϕ is satisfied, since S_c has at least one element x_i or $\bar{x}_i$ that is colored differently than F.