Question 1

Required: Show that if $P = NP$, then every language $A \in P$ except $A = \emptyset$ and $A = \Sigma^*$ is NP-complete.

Proof. Assume $P = NP$. Let $A \in P$ such that $A \neq \emptyset$ and $A \neq \Sigma^*$.

Want to Show: A is NP-complete.

Since $A \in P$ and $P = NP$, we have that $A \in NP$.

Since $A \in P$, we know there exists some TM M such that M decides A in polynomial time.

Since $A \neq \emptyset$ and $A \neq \Sigma^*$, we know there exists some $w_1 \in A$ and there exists some $w_2 \notin A$.

Now let $B \in NP$ be arbitrary. Consider the following TM F.

$F =$ "on input w:

1. Run M on w.
2. If M accepts w, then output w_1 on the tape.
3. If M rejects w, then output w_2 on the tape."

Clearly F runs in polynomial time as Step 1 simulates M which runs in polynomial time with respect to the length of the input, and Steps 2 and 3 trivially runs in polynomial time.

So F induces a computable function $f : \Sigma^* \to \Sigma^*$ where F runs in polynomial time.

And clearly $w \in B$ if and only if $f(w) \in A$. Hence, we have $B \leq_p A$.

Since $B \in NP$ was arbitrary, we have that every $B \in NP$ is such that $B \leq_p A$. So A is NP-hard.

Since $A \in P$ and $P = NP$, we have that $A \in NP$.

Since A is NP-hard and $A \in NP$, we have that A is NP-complete, as required. $\square$

Question 2

Required: Show that $TAUT$ is $coNP$-complete (i.e. $TAUT \in coNP$ and there is a reduction $A \leq_p TAUT$ for all $A \in coNP$.

Proof. First we will show that $TAUT \in coNP$. Hence, we must show that $\overline{TAUT} \in NP$.

Consider the following polynomial time verifier V for $\overline{TAUT}$ which considers a falsifying truth assignment as a certificate.

$V =$ "on input $\langle \phi, c \rangle$:

1. Test that ϕ is indeed a well-formed formula and that c is an encoding of an assignment of truth values to all the variables in ϕ.
2. Test that ϕ is false under the truth assignment that c encodes.
3. If both tests pass, then accept. Otherwise, reject."

If $\langle \phi \rangle \in \overline{TAUT}$, then there exists a truth assignment that makes ϕ false. Hence, a certificate c exists which encodes this assignment and V accepts $\langle \phi, c \rangle$. If $\langle \phi \rangle \notin \overline{TAUT}$, then ϕ is a tautology and there does not exist a truth assignment that makes ϕ false, and so no such certificate exists. And clearly V is a decider as it always halts. Hence, V is a verifier.

Clearly V is a poly-time verifier as the number of variables of ϕ is less than $|\langle \phi \rangle|$. And a valid certificate is just an assignment of truth values to these variables that make ϕ false. And checking that this assignment falsifies ϕ also takes polynomial time with respect to $|\langle \phi \rangle|$.

Since we have a poly-time verifier for $\overline{TAUT}$, we have that $\overline{TAUT} \in NP$. Hence, we have that $TAUT \in coNP$.

Now consider the following reduction from SAT to $\overline{TAUT}$ given by the following TM F.

$F =$ "on input $\langle \phi \rangle$:

1. Output $\langle \neg \phi \rangle$"

And if F is given an input that is not an encoding of a well-formed formula ϕ, then output $\langle x_1 \vee \overline{x_1} \rangle$ where $x_1 \vee \overline{x_1}$ is a tautology.

Hence, F induces the the following computable function $f : \Sigma^* \to \Sigma^*$.

$$f(x) = \begin{cases} \langle \neg \phi \rangle & \text{if } x = \langle \phi \rangle \\ \langle x_1 \vee \overline{x_1} \rangle & \text{otherwise} \end{cases}$$

Note that if ϕ is any satisfiable CNF formula, we have that $\neg \phi$ cannot be a tautology since any assignment that makes ϕ true would make $\neg \phi$ false.

Hence, we get $x \in SAT$ if and only if $f(x) \in \overline{TAUT}$. And trivially F runs in polynomial time as we are just adding a negation sign $\neg$ in front of ϕ. Hence, f is a polynomial reduction.

Hence, $SAT \leq_p \overline{TAUT}$.

Now let $A \in coNP$. Hence, $\overline{A} \in NP$. We know that SAT is NP-complete from lecture and the textbook. Hence, $\overline{A} \leq_p SAT$. Since $\overline{A} \leq_p SAT$ and $SAT \leq_p \overline{TAUT}$, by transitivity of $\leq_p$ we get that $\overline{A} \leq_p \overline{TAUT}$.

From $\overline{A} \leq_p \overline{TAUT}$, we want to show $A \leq_p TAUT$. Since $\overline{A} \leq_p \overline{TAUT}$, there exists a polynomial reduction g such that $w \in \overline{A} \Leftrightarrow g(w) \in \overline{TAUT}$. Equivalently, $w \notin A \Leftrightarrow g(w) \notin TAUT$. Equivalently, via contrapositive we get $w \in A \Leftrightarrow g(w) \in TAUT$. Hence, $A \leq_p TAUT$.

Since $A \in coNP$ was arbitrary, we have that every $A \in coNP$ is such that $A \leq_p TAUT$. Since we've shown earlier that $TAUT \in coNP$, we have that $TAUT$ is $coNP$-complete, as required.

$\square$

Question 3

Let CNF_k be the set of all satisfiable CNF formulas where each variable appears at most k times.

Required: Show that $CNF_2 \in P$.

Defining a TM M that decides CNF_2

Consider the following TM M that decides CNF_2 in polynomial time.

$M =$ "on input $\langle \phi \rangle$:

1. Test that ϕ is CNF and each variable in ϕ appears at most 2 times. If this test fails, reject immediately.

2. Let $x_1, x_2, ..., x_k$ be the variables that occur in ϕ. For each $i \in \{1, ..., k\}$, repeat the following sub-steps i), ii), iii), and iv).

i) If x_i occurs exactly once in ϕ in some clause c_i, then modify ϕ so that the clause c_i containing x_i is removed. Note that x_i could appear as x_i or $\overline{x_i}$ in such a clause c_i.

ii) If x_i occurs exactly twice in ϕ, where both occurrences are x_i or both occurrences are $\overline{x_i}$, then modify ϕ so that the clause(s) containing these occurrences are removed.

iii) If ϕ contains an occurence of x_i and an occurrence of $\overline{x_i}$ in the same clause c_i, then remove this clause.

iv) If ϕ contains an occurence of x_i and an occurrence of $\overline{x_i}$ in distinct clauses c_{i_1} and c_{i_2}, then consider the following modification to ϕ. If $c_{i_1} = x_i$ and $c_{i_2} = \overline{x_i}$, then do not modify ϕ. If $c_{i_1} \neq x_i$ or $c_{i_2} \neq \overline{x_i}$, then modify ϕ by replacing the causes c_{i_1} and c_{i_2} with the clause c', where c' is the clause containing all the literals of c_{i_1} and c_{i_2} except for x_i and $\overline{x_i}$.

3. If $\phi = \epsilon$, then accept. Otherwise, reject."

Explanation of M

It is clear that M always halts, and hence is a decider. To see that M decides CNF_2 in polynomial time, first we will clarify what M in fact accomplishes.

Step 1 simply checks that ϕ is indeed CNF where each variable in ϕ appears at most 2 times.

Step 2 then reduces ϕ by removing and modifying clauses of ϕ. Please see the next page.

Variables x_i that occur exactly once in ϕ can have 1 assigned to x_i (or 0 to x_i if $\overline{x_i}$ appears) to make their respective clauses satisfied. Hence, we can reduce ϕ by removing these clauses which is what Step 2 i) does. The satisfiability of the original ϕ is unchanged.

Variables x_i that occur twice (either both x_i or both $\overline{x_i}$) can have (1 or 0 respectively) assigned to x_i to make their respective clause(s) satisfied. Hence, we can reduce ϕ by removing these clause(s) which is what Step 2 ii) does. The satisfiability of the original ϕ is unchanged.

Variables x_i that occur twice (one occurrence of x_i and one occurence of $\overline{x_i}$) within the same clause can have x_i assigned 1 to make this respective clause satisfied. Hence, we can reduce ϕ by removing this clause which is what Step 2 iii) does. The satisfiability of the original ϕ is unchanged.

Variables x_i that occur twice (one occurrence of x_i and one occurence of $\overline{x_i}$) in distinct clauses c_{i_1} and c_{i_2} cause ϕ to be modified as follows. If the occurrences of x_i and $\overline{x_i}$ are the only literals in their respective clauses, then ϕ is unchanged. Otherwise, we replace the causes c_{i_1} and c_{i_2} with the clause c', where c' is the clause containing all the literals of c_{i_1} and c_{i_2} except for x_i and $\overline{x_i}$. Note that this does not change the satisfiability of ϕ since it is clear that c' is satisfiable if and only if $c_{i_1} \wedge c_{i_2}$ is satisfiable. This is the case since any assignment that satisfies c' must satisfy at least one literal that does not contain the variable x_i. Hence, at least one of c_{i_1} or c_{i_2} would be satisfied. WLOG, suppose c_{i_1} is satisfied. Since c_{i_2} contains x_i or $\overline{x_i}$, assign 1 (or 0) respectively to x_i to satisfy c_{i_2}. Hence, we would satisfy $c_{i_1} \wedge c_{i_2}$. Conversely, any assignment that satisfies $c_{i_1} \wedge c_{i_2}$ must satisfy some literal that is not x_i or $\overline{x_i}$. This literal would also be part of c', and hence c' would be satisfied. Hence, the satisfiability of the now reduced ϕ is also unchanged.

When we reach Step 3, our modified ϕ will be in one of two cases. In the first case, ϕ will be an empty formula (no connectives or variables). In the second case, ϕ will contain clauses with 1 literal each, and ϕ will contain pairs of contradictory clauses like $c' = x_i$ and $c'' = \overline{x_i}$ which is unsatisfiable since $x_i \wedge \overline{x_i}$ is unsatisfiable.

Hence, if $\langle \phi \rangle \in CNF_2$, then ϕ will be a satisfiable CNF formula where each variable appears at most twice, and M will reduce ϕ to the empty string and be accepted.

And, if $\langle \phi \rangle \notin CNF_2$, then ϕ is either not a CNF formula where each variable appears at most twice and will be rejected at Step 1, or ϕ is not satisfiable and on Step 3 will be reduced to a formula containing contradictory clauses x_i and $\overline{x_i}$ for some variable x_i, and will be rejected.

Each step in M clearly takes polynomial time with respect to $|\langle \phi \rangle|$ since each step simply scans through the string $\langle \phi \rangle$ and possibly does some deletion of the string $\langle \phi \rangle$ as we modify and delete the clauses of ϕ. Hence, M runs polynomial in $|\langle \phi \rangle|$.

Therefore, $CNF_2 \in P$, as required.

Question 4

Required: Show that CNF_3 is NP-complete.

Show $CNF_3 \in NP$

Consider the following verifier V for CNF_3 which uses a satisfying assignment as a certificate.

$V =$ "on input $\langle \phi, c \rangle$:

1. Test that ϕ is a CNF formula where each variable appears at most 3 times.
2. Test that c is an encoding of an assignment of truth values to all the variables in ϕ.
3. Test that ϕ is true under the truth assignment that c encodes.
4. If all tests pass, then accept. Otherwise, reject."

Clearly V is a decider as it always halts. If $\langle \phi \rangle \in CNF_3$, then ϕ is a satisfiable CNF formula where each variable appears at most 3 times. Hence, a satisfying assignment exists which serves a certificate c, and V will accept $\langle \phi, c \rangle$. If $\langle \phi \rangle \notin CNF_3$, then either ϕ is not CNF, or it is not the case that every variable appears at most 3 times, or ϕ is not satisfiable. Hence, no such certificate exists.

Clearly V is a poly-time verifier as the number of variables of ϕ is less than $|\langle \phi \rangle|$. And a valid certificate is just an assignment of truth values to these variables that make ϕ true. And checking that this assignment satisfies ϕ also takes polynomial time with respect to $|\langle \phi \rangle|$.

Hence, $CNF_3 \in NP$.

Show CNF_3 is NP-Hard

We know from lecture and the textbook that SAT_3 is NP-Complete. We will show that $SAT_3 \leq_p CNF_3$.

Let α be a CNF formula with three literals per clause.

Let $Vars_\alpha = \{x_1, x_2, x_3, ..., x_k\}$ be the set of variables that appear in α. Note that a variable $x_i \in Vars_\alpha$ could appear more than 3 times in α.

Hence, we need to come up with a new formula α', where each variable in α' appears at most 3 times such that α is satisfiable if and only if α' is satisfiable.

First let β be the same formula as α except that all the variables are renamed so that each variable appears exactly once.

We need to add additional conjunctive clauses to β so that variables that were intially the same variable in α are always assigned the same value in any satisfying assignment.

i.e. Let $x_i \in Vars_\alpha$. Assume that there are j occurrences of x_i in α. Then each occurrence of x_i is renamed in β. Let the renamed occurrences of x_i in β be $y_{i_1}, y_{i_2}, ..., y_{i_j}$.

We want to enforce the fact that all the renamed variables $y_{i_1}, y_{i_2}, ..., y_{i_j}$ are all given the same value (true or false) in any satisfying assignment.

To do this, consider the following formula.

$$(y_{i_1} \Rightarrow y_{i_2}) \wedge (y_{i_2} \Rightarrow y_{i_3}) \wedge \cdots \wedge (y_{i_{j-1}} \Rightarrow y_{i_j}) \wedge (y_{i_j} \Rightarrow y_{i_1})$$

The above formula is clearly satisfied only when $y_{i_1}, y_{i_2}, ..., y_{i_j}$ are all assigned True, or are all assigned False since the formula is a closed chain of implications. But we need this formula to be in CNF. But we can easily translate this formula by translating each implication to the disjunction connective using the logical equivalence $\phi \Rightarrow \psi \equiv \neg\phi \vee \psi$. Hence, we get the following equivalent formula.

$$(\overline{y_{i_1}} \vee y_{i_2}) \wedge (\overline{y_{i_2}} \vee y_{i_3}) \wedge \cdots \wedge (\overline{y_{i_{j-1}}} \vee y_{i_j}) \wedge (\overline{y_{i_j}} \vee y_{i_1})$$

So let $\gamma_{x_i} := (\overline{y_{i_1}} \vee y_{i_2}) \wedge (\overline{y_{i_2}} \vee y_{i_3}) \wedge \cdots \wedge (\overline{y_{i_{j-1}}} \vee y_{i_j}) \wedge (\overline{y_{i_j}} \vee y_{i_1})$.

Note that γ_{x_i} is CNF and each variable in γ_{x_i} appears exactly twice.

Since we need to enforce the above for each variable $x_i \in Vars_\alpha$, consider the conjunction $\gamma := \bigwedge\limits_{x_i \in Vars_\alpha} \gamma_{x_i}$

So $\beta \wedge \gamma$ is our desired formula which is in CNF. Since each variable in β appears exactly once, and each variable in γ appears exactly twice, we have that each variable in $\beta \wedge \gamma$ appears exactly 3 times.

Let $\alpha' := \beta \wedge \gamma$.

Want to Show: $\langle\alpha\rangle \in SAT_3$ if and only if $\langle\alpha'\rangle \in CNF_3$

$(\Rightarrow)$: If $\langle\alpha\rangle \in SAT_3$, then there exists a truth assignment v that makes α true. Consider a new assignment v' that assigns the same truth value that v assigns to each $x_i \in Vars_\alpha$ to each of x_i's renamed variables in α'. Hence, v' satisfies β. And since each renamed variable of each occurence of x_i is given the same truth value for each variable x_i in α, we have that γ is also satisfied. Hence, $\alpha' := \beta \wedge \gamma$ is satisfied. Hence, $\langle\alpha'\rangle \in CNF_3$.

$(\Leftarrow)$: If $\langle\alpha'\rangle \in CNF_3$, then there exists a truth assignment v' that makes α' true. Given how γ was constructed, we know that the group of variables $y_{i_1}, y_{i_2}, ..., y_{i_j}$ that were renamings of the same variable x_i from α must have the same truth value (all True or all False). Hence, let v be an assignment that assigns each $x_i \in Vars_\alpha$ the same truth value as v' assigns to each of $y_{i_1}, y_{i_2}, ..., y_{i_j}$. Hence, v makes α true so that α is satisfiable. Hence, $\langle\alpha\rangle \in SAT_3$.

Please see the next page.

Now, consider the following TM F that induces a computable function f.

$F =$ "on input $\langle \alpha \rangle$:

1. Check that α is CNF with at most 3 literals per clause. If not, output $\langle x_1 \wedge \overline{x_1} \rangle$.

2. Otherwise, construct the formula $\alpha' := \beta \wedge \gamma$ as outlined in the above discussion.

3. Output $\langle \alpha' \rangle$."

Notice, if F is given a string that is not a valid input, we simply output $\langle x_1 \wedge \overline{x_1} \rangle$, where $x_1 \wedge \overline{x_1}$ is a CNF formula where each variable appears at most 3 times and is not satisfiable.

Hence, F induces the following function $f : \Sigma^* \to \Sigma^*$.

$$f(w) = \begin{cases} \langle \alpha' \rangle & \text{if } w = \langle \alpha \rangle \text{, where } \alpha \text{ is CNF with 3 literals per clause} \\ \langle x_1 \wedge \overline{x_1} \rangle & \text{otherwise} \end{cases}$$

Recall, we showed earlier that $\langle \alpha \rangle \in SAT_3$ if and only if $\langle \alpha' \rangle \in CNF_3$.

Hence, $w \in SAT_3$ if and only if $f(w) \in CNF_3$.

And clearly F is polynomial as each step only takes polynomial steps with respect to $|\langle \alpha \rangle|$. i.e. We only take polynomial steps with respect to $|\langle \alpha \rangle|$ in constructing α'.

Hence, f is a polynomial reduction. Hence, $SAT_3 \leq_p CNF_3$.

From lecture and the textbook we know that SAT_3 is NP-complete.

Let $A \in NP$. Hence, $A \leq_p SAT_3$. Since $A \leq_p SAT_3$ and $SAT_3 \leq_p CNF_3$, by transitivity of $\leq_p$ we get that $A \leq_p CNF_3$. Since $A \in NP$ was arbitrary, we get that CNF_3 is NP-hard.

Since CNF_3 is NP-hard and since we showed earlier that $CNF_3 \in NP$, we conclude that CNF_3 is NP-complete, as required.

Question 5

Required: Show that TF is NP-complete.

First we will show that $TF \in NP$.

Consider the following verifier V_0 which uses a triangle-free subset as a certificate.

$V_0 =$ "on input $\langle\langle G, k\rangle, c\rangle$:

1. Test that $G = (V, E)$ for some set of vertices V and set of edges E.
2. Test that $c = \langle S\rangle$ is an encoding of some subset $S \subseteq V$ such that $|S| \geq k$.
3. For every 3 element subset $\{x, y, z\} \subseteq S$, test that at least one of the possible edges between any two distinct vertices in $\{x, y, z\}$ is not an element of E.
4. If all tests pass, then accept. Otherwise, reject."

Clearly V_0 is a decider since it always halts. And if $\langle G, k\rangle \in TF$, then G has a triangle-free subset S of size at least k. Hence, all tests will clearly pass and V_0 will accept $\langle\langle G, k\rangle, c\rangle$ where $c = \langle S\rangle$. And if $\langle G, k\rangle \notin TF$, then G does not have a subset of size at least k that is triangle-free. Hence, no such certificate exists.

To show that V_0 runs in polynomial time, notice that Steps 1 and 2 clearly runs in polynomial time with respect to the input length. And Step 3 considers any 3 element subset of S where $S \subseteq V$ and $|S| \geq k$. Hence, the number of subsets we're considering is $\binom{|S|}{3} \leq \binom{|V|}{3} = \frac{|V|!}{3!(|V|-3)!} = \frac{|V|(|V|-1)(|V|-2)}{3!} \in O(|V|^3)$. Since this is polynomial with respect to $|V|$, we have that Step 3 is clearly polynomial with respect to the length of the input. Hence, V_0 is polynomial. Since we have a polynomial verifier V_0 for TF, we have that $TF \in NP$.

Notation: Since we are working with undirected graphs, we will represent an edge between vertices x and y by a set $\{x, y\}$. This is consistent with Professor Saraf's notation from the lecture slides.

As the hint suggests, we will reduce from $INDSET$.

Let $G = (V, E)$ be a graph for some set of vertices V and set of edges E.

Let V_0 be a set of vertices such that $V \cap V_0 = \emptyset$ and $|V_0| = |V| + 1$.

Let $V' = V \cup V_0$. Note, $|V'| = |V| + |V_0| = |V| + |V| + 1 = 2|V| + 1$

Let $E' = E \cup \{\{x, y\} : x \in V \wedge y \in V_0\}$.

Consider the graph $G' = (V', E')$.

Let $k \in \mathbb{N}$.

Want to Show: Given G and G' as defined, we have that $\langle G, k\rangle \in INDSET$ if and only if $\langle G', k + |V| + 1\rangle \in TF$

($\Rightarrow$): Assume $\langle G, k\rangle \in INDSET$. Hence, G has an independent set $S \subseteq V$ such that $|S| \geq k$.

Now consider the set $S \cup V_0$. Since $S \subseteq V$ and $V \cap V_0 = \emptyset$, we have that $S \cap V_0 = \emptyset$. Hence, $|S \cup V_0| = |S| + |V_0| \geq k + |V| + 1$.

And we have $S \cup V_0 \subseteq V'$. We just need to show that $S \cup V_0$ is triangle-free. Let $\{x, y, z\} \subseteq S \cup V_0$.

Case 1: Assume $x, y, z \in V_0$. Looking at the definition of E', since $x, y \in V_0$ and $V_0 \cap V = \emptyset$, we have that $\{x, y\} \notin E'$. Hence, there is no triangle involving x, y, z in G'.

Case 2: Assume exactly two of x, y, z are contained in V_0. WLOG, assume $x, y \in V_0$. Looking at the definition of E', since $x, y \in V_0$ and $V \cap V_0 = \emptyset$, we have that $\{x, y\} \notin E'$. Hence, there is no triangle involving x, y, z in G'.

Case 3: Assume exactly one of x, y, z is contained in V_0. WLOG, assume $x \in V_0$ so that $y, z \in S$. Since $y, z \in S$, and S is an independent set, we know that $\{y, z\} \notin E$. We know that $\{y, z\} \notin \{\{x, y\} : x \in V \wedge y \in V_0\}$ since $y, z \in S \subseteq V$ and $V \cap V_0 = \emptyset$. Hence, $\{y, z\} \notin E'$. Hence, there is no triangle involving x, y, z in G'.

In all three cases we have no triangle involving x, y, z in G'. Therefore, $S \cup V_0$ is a triangle-free set such that $|S \cup V_0| \geq k + |V| + 1$. Hence, $\langle G', k + |V| + 1\rangle \in TF$.

($\Leftarrow$): Assume $\langle G', k + |V| + 1\rangle \in TF$. Hence, G' has a triangle-free set $S' \subseteq V'$ such that $|S'| \geq k + |V| + 1$.

Let $S = S' \setminus V_0$. Notice,

$$\begin{aligned}
|S| &= |S' \setminus V_0| & \\
&\geq |S'| - |V_0| & \text{By Properties of Sets} \\
&\geq (k + |V| + 1|) - |V_0| & \text{Since } |S'| \geq k + |V| + 1 \\
&= (k + |V| + 1) - (|V| + 1) & \text{Since } |V_0| = |V| + 1 \\
&= k &
\end{aligned}$$

Hence, $|S| \geq k$. And $S \subseteq V$ by definition. We want to show that $S \subseteq V$ is an independent set in G.

Let $x, y \in S$ be distinct vertices. We want to show that $\{x, y\} \notin E$.

Assume for contradiction that $\{x, y\} \in E$. Since $E \subseteq E'$, we have that $\{x, y\} \in E'$.

Note that $|S'| \geq k + |V| + 1$. And $S' \subseteq V' = V \cup V_0$, where $|V_0| = |V| + 1$ and $V \cap V_0 = \emptyset$. Hence, there must exist a $z \in S'$ such that $z \in V_0$.

Since $x, y \in S \subseteq V$ and $z \in V_0$, looking at $E' = E \cup \{\{x, y\} : x \in V \land y \in V_0\}$, we get that $\{x, z\} \in E'$ and $\{y, z\} \in E'$.

So we have $\{x, y\}, \{x, z\}, \{y, z\} \in E'$. But we know that $x, y \in S \subseteq S'$. And $z \in S'$. So $x, y, z \in S'$ where x, y, z form a triangle, contradicting the fact that S' is triangle-free.

Therefore, our initial assumption was wrong and $\{x, y\} \notin E$. Hence, we've shown that S is an independent set.

Now consider the following TM F which induces a computable function f.

$F =$ "on input $\langle G, k\rangle$:

1. Check that $G = (V, E)$ for some set of vertices V and set of edges E. If not, output $\langle H, 3\rangle$, where H is a graph with exactly 3 vertices x, y, z with edges connected pairwise as a triangle.

2. Otherwise, construct the graph G' as outlined in the above discussion.

3. Output $\langle G', k + |V| + 1\rangle$."

Note that if F is given an input that is not a valid graph, we output $\langle H, 3\rangle$, where H is a graph with exactly 3 vertices with edges connected as a triangle. Hence, $\langle H, 3\rangle \notin TF$.

Hence, F induces the following function $f : \Sigma^* \to \Sigma^*$.

$$f(w) = \begin{cases} \langle G', k + |V| + 1\rangle & \text{if } G = (V, E) \text{ for some set of vertices } V \text{ and set of edges } E \\ \langle H, 3\rangle & \text{otherwise} \end{cases}$$

And we've shown earlier that $\langle G, k\rangle \in INDSET$ if and only if $\langle G', k + |V| + 1\rangle \in TF$.

Hence, $w \in INDSET$ if and only if $f(w) \in TF$.

We know F is polynomial-time since constructing G' is trivially proportional polynomially to the length $|\langle G\rangle|$. And constructing H is always constant.

So f is a polynomial reduction. Hence, $INDSET \leq_p TF$.

We know from lecture that $INDSET$ is NP-complete. So let $A \in NP$. Hence, $A \leq_p INDSET$. Since $A \leq_p INDSET$ and $INDSET \leq_p TF$, we have that $A \leq_p TF$. Since A

was arbitrary, we have that TF is NP-hard.

Since TF is NP-hard, and since we've shown earlier that $TF \in NP$, we have that TF is NP-complete, as required.