Problem 1

Solution. (a) HALT is NP-hard. We will prove that, for any language \( L \in \text{NP} \), \( L \leq_p \text{HALT} \).

If \( L \in \text{NP} \), there exists a polynomial time TM \( M \) and a polynomial \( p : \mathbb{N} \to \mathbb{N} \) such that for every \( x \in \{0, 1\}^* \),
\[
  x \in L \iff \exists u \in \{0, 1\}^{p(|x|)} \text{ s.t. } M(x, u) = 1.
\]

We claim that, for every \( x \), another string \( x' \) can be constructed in polynomial time such that \( x \in L \iff x' \in \text{HALT} \). The construction follows.

On input \( x \), we construct \( x' = \langle S \rangle \) where \( S \) is a TM that runs \( M \) on every \( u \in \{0, 1\}^{p(|x|)} \) to check whether there is a \( u \in \{0, 1\}^{p(|x|)} \) such that \( M(x, u) = 1 \). If yes, then \( S \) will halt. Otherwise, \( S \) will loop forever. We can see that \( S \) can be constructed in polynomial time.

Here is the analysis. If \( x \in L \), then there exists \( u \in \{0, 1\}^{p(|x|)} \) such that \( M(x, u) = 1 \). So \( x' \in \text{HALT} \). If \( x \not\in L \), \( \forall u \in \{0, 1\}^{p(|x|)} \), \( M(x, u) = 0 \), so \( S \) will loop forever. Thus \( x' \not\in \text{HALT} \). This proves that \( L \leq_p \text{HALT} \).

As a result, HALT is NP-hard.

(b) No, because HALT is not in NP. Suppose HALT is in NP. As NP \( \subseteq \text{EXP} \), HALT can be decided by a DTM in exponential time. However, HALT is not decidable. This is a contradiction. So HALT is not in NP. \( \square \)

Problem 2

Solution. We show that there exists a polynomial time TM \( M \) and a polynomial \( p \), such that given the input \( \langle G, H \rangle \),
\[
  \langle G, H \rangle \in \text{ISO} \iff \exists u \in \{0, 1\}^{p(s)}, M(\langle G, H \rangle, u) = 1,
\]
where \( s = |\langle G, H \rangle| \).

For \( \langle G, H \rangle \), the certificate is \( n \) pairs of vertices where \( n \) is the number of vertices of \( G \), such that the \( i \)th pair \( v_i, w_i \) indicates that the vertex \( v_i \) in \( G \) corresponds to the vertex \( w_i \) in \( H \). The TM \( M \) checks the following assertions.

- The two graphs have the same number of vertices and edges;
- For every edge \((v_i, v_j)\) of \( H \), \((v_i, v_j)\) is also an edge in \( G \).
- For every edge \((v_i, v_j)\) of \( G \), \((v_i, v_j)\) is also an edge in \( H \).

\( M \) accepts if and only if all three assertions hold.

We can see the running time of \( M \) is \( O(\tilde{n} + \tilde{m}) \) where \( \tilde{n} \) is the total number of vertices of \( G \) and \( H \), and \( \tilde{m} \) is the total number of edges of \( G \) and \( H \). This is because \( M \) only enumerates the vertices and edges of \( G \) and \( H \) for a constant number of times.

If \( \langle G, H \rangle \in \text{ISO} \), according to the definition of isomorphic, we know that there is a one to one correspondence between the vertices of \( G \) and the vertices of \( H \). As a result, we have \( n \) pair of vertices (assume it is the string \( u \)) which indicates the correspondence. So the three assertions are all true when the input \( \langle G, H \rangle, u \). So \( M(\langle G, H \rangle, u) = 1 \). On the other hand, if \( \exists u \in \{0, 1\}^{p(s)}, M(\langle G, H \rangle, u) = 1 \), we can get a correspondence between the vertices of \( G \) and \( H \) according to \( u \). The three assertions guarantees that the two graphs are isomorphic according to the definition of isomorphic. \( \square \)
Problem 3

Solution. As \( A \in \mathbb{P} \) and \( \mathbb{P} = \mathbb{NP} \), \( A \in \mathbb{NP} \).

Also as \( \mathbb{P} = \mathbb{NP} \), for every \( B \in \mathbb{NP} \), there exists a polynomial time \( \mathcal{T} \mathcal{M} M' \) such that

\[
x \in B \iff M'(x) = 1.
\]

As \( A \) is not \( \emptyset \) or \( \Sigma^* \), there exists a fixed string \( a \in \Sigma^* \) such that \( a \in A \) and a fixed string \( b \in \Sigma^* \) such that \( b \notin A \). (Note that we can find \( a, b \) in a fixed amount of time in advance, given the language \( A \), just by enumerating every string and stop whenever we find an \( a \in A \) and \( b \notin A \).) We construct a mapping function \( f \) which maps an input \( x \) of \( \mathcal{B} \) to an input of \( A \). The function first checks whether \( M'(x) = 1 \). If yes, it outputs \( f(x) = a \); otherwise it output \( f(x) = b \). This function is polynomial time computable. In addition, if \( x \in B \), then \( M'(x) = 1 \) so \( f(x) = a \in A \); otherwise \( M'(x) = 0 \), so \( f(x) = b \notin A \). This shows that \( A \) is \( \mathbb{NP} \)-hard.

This shows that \( A \) is \( \mathbb{NP} \)-complete.

\( \square \)

Problem 4

Solution. (a) For any clause \( C \) of \( \phi \), the \( \neq \)-assignment will assign two literals in \( C \) with different truth values. This means that one of them is 1. So \( C \) is satisfied. So \( \phi \) is satisfied.

For the negation of a \( \neq \)-assignment, every literal takes its negation. As a result, for each clause, the two literals that have different truth values are both flipped. So they are still not equal. So the assignment is still a \( \neq \)-assignment.

(b) Let \( \phi \) be an arbitrary input for 3SAT. We transform \( \phi \) to \( \phi' \) according to the transformation given in the problem.

Assume \( \phi \) is satisfied by some assignment. Then for any clause \( c_i = (y_1 \lor y_2 \lor y_3) \), at least one of \( y_1, y_2, y_3 \) is 1 under this assignment. In order to satisfy \( (y_1 \lor y_2 \lor z_i) \) and \( (\bar{z}_i \lor y_3 \lor b) \), we can let \( z_i = \neg(y_1 \lor y_2) \) and \( b = 0 \). We can see that this is a \( \neq \)-assignment because \( z_i = \neg(y_1 \lor y_2) \) and \( b = \neg(\bar{z}_i \lor y_3) \). So \( \phi' \) has a \( \neq \)-assignment.

On the other hand, assume \( \phi' \) has a \( \neq \)-assignment. If \( b = 1 \) in this assignment, by (a) we can take the negation of the assignment to get another \( \neq \)-assignment which has \( b = 0 \). If \( b = 0 \) in the assignment, then for each clause \( c_i \), the literals \( y_1, y_2, y_3 \) cannot be all 0 (otherwise no \( z_i \) can simultaneously satisfy \( (y_1 \lor y_2 \lor z_i) \) and \( (\bar{z}_i \lor y_3 \lor b) \)). So \( c_i \) is satisfied and thus \( \phi \) is satisfied.

This shows that \( \neq \text{SAT} \) is \( \mathbb{NP} \)-hard.

(c) We need to show that \( \neq \text{SAT} \) is in \( \mathbb{NP} \). For any input string \( \phi \), the certificate is an assignment \( x \) to the variables. We know that \( \phi \in \neq \text{SAT} \) if and only if there exists a \( \neq \)-assignment. We can check that \( x \) is a \( \neq \)-assignment by checking that each clause has two literals that have different values. This can be done in polynomial time. So \( \neq \text{SAT} \) is in \( \mathbb{NP} \).

As a result, \( \neq \text{SAT} \) is \( \mathbb{NP} \)-complete.

\( \square \)