Problem 1. For each of the following, determine if the statement is true or false. No explanation is necessary.

a) \(2n^2 + 4n - 50^{1000000} \in O(n^2)\)

b) \(n \in O(\sqrt{n^2})\)

c) If \(f \in O(g)\), then \(\sin f \in O(\sin g)\)

d) \(TIME(n) \subseteq NTIME(n)\)

e) \(TIME(n) = NTIME(n)\)

Problem 2. Prove that if \(f \in O(g)\) and \(g \in O(h)\), then \(f \in O(h)\).

Problem 3. Define the language

\[
PATH = \{ \langle G, s, t \rangle \mid G \text{ is a directed graph with an } s-t \text{ path} \}
\]

Remember that a \(s-t\) path in a graph visits each vertex at most once, starts at vertex \(s\), and ends at vertex \(t\).

Prove that \(PATH \in P\).

Problem 4. The longest path problem involves finding the longest path from a source vertex to a destination vertex in a weighted, directed graph. For convenience, we’ll assume that the weights on each edge of the graph are integers specified by weight function \(\omega\). Then we can encode this as a decision problem as follows.

\[
LP = \{ \langle G, \omega, s, t, k \rangle \mid G \text{ is a weighted, directed graph and has an } s-t \text{ path of length at least } k \}
\]

Remember that a \(s-t\) path in a graph visits each vertex at most once, starts at vertex \(s\), and ends at vertex \(t\).

Prove \(LP \in NP-COMPLETE\).

Problem 5. Let

\[
DOUBLE-SAT = \{ \langle \phi \rangle \mid \phi \text{ is a Boolean formula with at least two satisfying assignments} \}
\]

Prove that \(DOUBLE-SAT \in NP-COMPLETE\).
Problem 6. There is a distinction to be made between decision problems, those that accept/reject strings for languages, and function problems which, compute a function. For instance, \( SAT \) is a decision problem that determines if a given Boolean formula is satisfiable. It does not, however, produce an assignment which satisfies it. The analogous class to \( P \) (\( NP \)) is \( FP \) (\( FNP \)), which is the set of function problems that can be solved by a deterministic (nondeterministic) Turing machine in polynomial time.

Show that if \( P = NP \), then there is a polynomial time (deterministic) algorithm which produces a satisfying assignment when given a Boolean formula.

(Hint: there exists a deterministic polynomial time algorithm which decides \( SAT \) which you can query any polynomial number of times you like)

Problem 7. Prove that if \( P \neq NP \), then every nontrivial (not \( \emptyset \) or \( \Sigma^* \)) language in \( P \) is in \( NP-COMPLETE \).

Problem 8. Give a language \( L \) that is not in \( NP \) and prove that \( L \notin NP \).

Problem 9. Prove that \( P \subseteq NP \cap co-NP \).

Problem 10. Recall that for a language \( L \) to be complete for a language class \( C \), then every language in \( C \) must mapping reduce to \( L \) (possibly subject to some resource bound) and \( L \) must be in \( C \). For \( P \) and \( NP \), the resource bound is polynomial time. This is also the case for \( co-NP \). A language \( L \) is in \( co-NP-COMPLETE \) if \( L \in co-NP \) and every language in \( co-NP \) mapping reduces to \( L \) in polynomial time.

Prove that if \( NP \neq co-NP \), then no \( NP-COMPLETE \) language can be in \( co-NP \) nor can any \( co-NP-COMPLETE \) language be in \( NP \).