NP

For some problems, it is much easier to check a solution than to find one from scratch. For example, there is no known fast way to find a Hamiltonian path in a graph (i.e. a path that visits each vertex once), but if someone just *gives you* a path in a graph, it is easy to *check* that it is Hamiltonian.

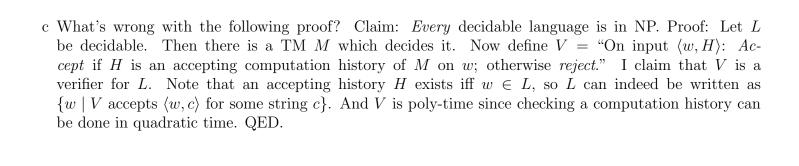
(Sipser Definition 7.18) A verifier for a language A is an algorithm V, where

$$A = \{w \mid V \text{ accepts } \langle w, c \rangle \text{ for some string } c\}.$$

(so you can think of c as the "evidence that w should be accepted", e.g. w could encode a graph and c could encode a Hamiltonian path in that graph.) We measure the time of a verifier only in terms of the length of w, so a polynomial time verifier runs in polynomial time in the length of w. A language A is polynomially verifiable if it has a polynomial time verifier.

a Prove that $COMPOSITES = \{\langle n \rangle \mid n \text{ is a composite integer} \}$ is polynomially verifiable (by constructing an explicit verifier).

b (Sipser Definition 7.19) NP is the class of languages that have polynomial time verifiers. Prove that $P \subseteq NP$.



d (Sipser Definition 7.9) Let N be a nondeterministic Turing machine that is a decider. The running time of N is the function $f: \mathbb{N} \to \mathbb{N}$, where f(n) is the maximum number of steps that N uses on any branch of its computation on any input of length n. (i.e. every branch must halt, and we score the machine based on its slowest branch, regardless of whether that branch is accepting or rejecting.)

Prove that if a language is in NP, then it is decided by some poly-time NTM.

e Prove that for any poly-time NTM $N, L(N) \in NP$.
f Come up with 2 or 3 problems which are definitely in NP but feel like they're <i>probably</i> not in P. Explain how you came up with your answers. (Try to make your problems feel significantly different from each other, e.g. don't come up with 3 examples that are all about graphs.)

g	Recall:	(Sipser	Definition	5.20)	Language	A is	mapping	${\it reducible}$	to	language	B,	written	A	\leq_m	B,	if
	there is	a comp	outable fund	ction .	$f: \Sigma^* \to \Sigma^*$, w	here for e	very w, w	€ .	$A \leftrightarrow f(w)$	\in .	B.				

Mapping reductions were a useful tool for making claims like "if B is decidable, A is also decidable". Formalize a similar concept \leq_P which could be a similarly useful tool for making claims like "if $B \in P$, $A \in P$ ". Prove that your \leq_P is transitive.

h Define a language $L \in NP$ to be NP-complete if we have that $L \in P$ iff every language in NP is also in P. (Informally, L is NP-complete if L is one of the "hardest" languages in NP.) Come up with two overall proof templates using \leq_P that we might be able to use to prove a language L is NP-complete. (Hint: what were some of the ways we could show a language was undecidable?)