1 The class P

1. How do we prove a language is in P?

Solution: We can prove a language L is in P by proving the following statements about L:

(a) L is decidable

(b) There exists a decider for L that runs in polynomial time with respect to $|x|$ where x is the input to the decider and $|x|$ is the size (length) of x's bitstring representation.

We can prove the first statement in the same way that we've shown how to prove languages are decidable in the past, by writing a decider for L.

To prove the second statement, we must analyze each step of the decider we created for L and show that each step runs in polynomial time with respect to the size of the input to the decider.

2. Show that the following language is in P.

$$L_{<376376} = \{(\langle M \rangle, x) : M \text{ is a Turing Machine and halts on } x \text{ in less than } |x|^{376376} \text{ steps}\}$$

Solution: We can decide $L_{<376376}$ by constructing the following decider D:

D = "On input $(\langle M \rangle, x)$:
1. Simulate execution of M on input x for at most $|x|^{376376} - 1$ steps
(break if M halts)
2. If M has halted, **accept**
3. Else, **reject**"

$$(\langle M \rangle, x) \in L_{<376376} \implies M \text{ halts on } x \text{ in no more than } |x|^{376376} - 1 \text{ steps} \implies D \text{ accepts}$$
$$(\langle M \rangle, x) \notin L_{<376376} \implies M \text{ halts on } x \text{ in at least } |x|^{376376} \text{ steps or loops} \implies D \text{ rejects}$$

Furthermore, D runs in polynomial time. Simulating execution of a Turing machine for a polynomial number of steps takes polynomial time. We simulate M for $|x|^{376376}$ steps, which is polynomial with respect to the size of x.

We have shown that we can decide $L_{<376376}$ in polynomial time. Therefore, $L_{<376376} \in$ P.

3. Suppose L_1 and L_2 are decidable languages such that $L_1 \in$ P. Then, $L_1 \cap L_2 \in$ P for (All/ some/ no) such L_1, L_2.

Solution: Some.

If L_1 and L_2 are both in P, then we can use efficient deciders for both to construct an efficient decider for $L_1 \cap L_2$.

However, suppose $L_1 = \Sigma^*$ (trivially in P) and $L_2 \notin \mathsf{P}$. Then, $L_1 \cap L_2 = L_2 \notin \mathsf{P}$.

4. $\textsc{SequenceSum} = \{(x \in \mathbb{N}, y \in \mathbb{N}) : (\sum_{i=1}^{x} i) = y\}$. Does the following decider for $\textsc{SequenceSum}$ show that $\textsc{SequenceSum} \in \mathsf{P}$? Explain why or why not.

$D =$ "On input (x, y):
1. $s \leftarrow 0$
2. **for** i from 1 to x **do**
$s \leftarrow s + i$
3. If $s = y$, **accept**
4. Else, **reject**"

Solution: No. Step 2 of D does not run in polynomial time with respect to $|(x, y)|$. Step 2 does x iterations, each of which takes $\Omega(1)$ time, so it runs in $\Omega(x)$ time, which is about $\Omega(2^{|x|})$. Since one of the steps of D runs in exponential time, D is not efficient. Therefore, D does not show that $\textsc{SequenceSum} \in \mathsf{P}$.

2 The class NP

1. How do we prove a language is in NP?

Solution: In order to show a language $L \in \mathsf{NP}$ we need to construct an efficient verifier for L; that is, it should run in polynomial time with respect to the size of the inputs from L. A verifier is a Turing Machine (algorithm) V such that if $x \in L$ then there is a certificate c such that $V(x, c)$ accepts, and if $x \notin L$ then $V(x, c)$ rejects for all certificates c.

2. Show that the following languages are in NP:

(a) $\textsc{PairSum} = \{(S, k) : \exists\, a, b \in S \text{ such that } a + b = k\}$

Solution: We'll write a verifier that accepts input $(x = (S, k), c = (a, b))$ only if certificate c contains a pair of numbers $a, b \in S$ that sum to k:

$V =$ "On input $(x = (S, k), c = (a, b))$:
1. Check that a, b are in S, reject if not
2. Check if $a + b = k$, reject if not
3. accept"

This verifier runs in polynomial time since we do one pass over S and do one computation $a + b = k$. Further,

- $x = (S, k) \in \textsc{PairSum} \implies$ there is $a, b \in S$ s.t $a+b = k \implies V(x, c)$ accepts for $c = (a, b)$
- $x = (S, k) \notin \textsc{PairSum} \implies$ there is no pair $a, b \in S$ s.t $a+b = k \implies$ for any pair of a, b either line 1, line 2, or line 4 rejects $\implies V(x, c)$ rejects for all c

(b) $\textsc{TripleFactor} = \{n \in \mathbb{Z} : n = pqr \text{ for some } p, q, r \in \mathbb{Z}\}$

Solution: We'll write a verifier on input $(x = n, c = (p, q, r))$ that accepts only if certificate c contains three integers whose product is n:

V = "On input $(n, (p, q, r))$:
1. Check that $-n \leq p, q, r \leq n$
2. Check that $pqr = n$
3. accept if so, reject otherwise"

This verifier runs in polynomial time since we are simply doing two multiplications of numbers that are at most size $|n|$ - which we know is a polynomial-time operation. Further,

- $n \in \textsc{TripleFactor} \implies n = pqr \implies V(n, c = pqr)$ accepts
- $n \notin \textsc{TripleFactor} \implies n$ cannot be factored into three integers $\implies V(n, c)$ rejects for all certificates c.

3. (True/False/Unknown) There are languages that are not efficiently decidable but efficiently verifiable.

Solution: Unknown.
If there were such a language, then $\mathsf{NP} \not\subseteq \mathsf{P}$ which would mean $\mathsf{P} \neq \mathsf{NP}$.