Let $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ be a finite alphabet, and suppose that the symbols have a total ordering $\sigma_1 < \sigma_2 < \cdots < \sigma_n$. We say a string $w_1 \cdots w_m$ is sorted if $w_i \preceq w_{i+1}$ for all $1 \leq i < m$. We say a language $L \subseteq \Sigma^*$ is sorted if every string in $L$ is sorted.

**Question 1a**

Let $\text{SORTED}_{\text{NFA}} = \{(M) : M \text{ is an NFA for which } L(M) \text{ is sorted}\}$. Prove or disprove: $\text{SORTED}_{\text{NFA}}$ is Turing decidable.

**Solution:** $\text{SORTED}_{\text{NFA}}$ is decidable. If a language $L$ is sorted, and the alphabet is $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ with total ordering $\sigma_1 < \sigma_2 < \cdots < \sigma_n$, then all strings $w \in L$ have the form $\sigma_1^n \cdots \sigma_n^n$. The idea here is to design a DFA $D$ such that it recognizes $\sigma_1^n \cdots \sigma_n^n$, and then remove all strings from the input DFA (which has been converted from an NFA) that are in $L(D)$. Then, we can have any finite number of strings. By the Pigeonhole Principle and the Pumping Lemma, if a DFA $D$ has a string with length $\geq |Q|$, then the language is infinite. Therefore, for deciding if the language is finite, we just need to check that the DFA accepts only strings of length $\leq |Q|$. We construct the language $\text{FINITE}_{\text{DFA}} = \{(M) | M \text{ is a DFA and } L(M) \text{ is finite}\}$ and a decider $D_{\text{finite}}$ for it:

$$D_{\text{finite}} = \text{“On input } \langle M \rangle \text{ where } M = (Q, \Sigma, \delta, q_0, F) \text{ is an NFA:}$$

1. Suppose $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ with total ordering $\sigma_1 < \sigma_2 < \cdots < \sigma_n$. Construct a DFA $D_{\text{sorted}}$ with $L(D_{\text{sorted}}) = \sigma_1^n \cdots \sigma_n^n$.
2. Construct an equivalent DFA $D$ from $M$.
3. Construct a DFA $C$ such that $L(C) = L(D) \setminus L(D_{\text{sorted}}) = L(D) \cap L(D_{\text{sorted}})$.
4. Run the decider $E_{\text{DFA}}$ on input $\langle C \rangle$.
5. If $E_{\text{DFA}}$ accepts $\langle C \rangle$, then accept; otherwise, reject.”

**Question 1b**

Let $\text{MOSTSORT}_{\text{NFA}} = \{(M) : M \text{ is an NFA for which } L(M) \text{ contains a finite number of unsorted strings}\}$. Prove or disprove: $\text{MOSTSORT}_{\text{NFA}}$ is Turing decidable.

**Solution:** This is equivalent to the (a) part, but after removing all the sorted strings, we can have any finite number of strings (instead of just 0). We just need to check if the resulting DFA’s language is finite. By the Pigeonhole Principle and the Pumping Lemma, if a DFA with states $Q$ accepts a string with length $\geq |Q|$, then the language is infinite. Therefore, for deciding if the language is finite, we just need to check that the DFA accepts only strings of length $< |Q|$. We construct the language $\text{FINITE}_{\text{DFA}} = \{(M) | M \text{ is a DFA and } L(M) \text{ is finite}\}$ and a decider $D_{\text{finite}}$ for it:

$$D_{\text{finite}} = \text{“On input } \langle M \rangle \text{ where } M \text{ is a DFA:}$$

1. Let $n$ be the number of states in $M$.
2. Construct a DFA $M_{\geq n}$ that accepts all strings of length $n$ or more.
3. Construct a DFA $N$ with $L(N) = L(M) \cap L(M_{\geq n})$.
4. Run $E_{\text{DFA}}$ on input $\langle N \rangle$.
5. If $E_{\text{DFA}}$ accepts, accept; otherwise, reject.”

Now we need to proceed similarly as in the (a) part, but using the decider $D_{\text{finite}}$ for $\text{FINITE}_{\text{DFA}}$. We construct a TM $T$:

$$T = \text{“On input } \langle M \rangle \text{ where } M \text{ is an NFA:}$$

1. Suppose $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ with total ordering $\sigma_1 < \sigma_2 < \cdots < \sigma_n$. Construct a DFA $D_{\text{sorted}}$ with $L(D) = \sigma_1^n \cdots \sigma_n^n$.
2. Construct an equivalent DFA $D$ from $M$.
3. Construct a DFA $C$ such that $L(C) = L(D) \setminus L(D_{\text{sorted}}) = L(D) \cap L(D_{\text{sorted}})$.
4. Run the decider $\text{FINITE}_{\text{DFA}}$ on input $\langle C \rangle$.
5. If $\text{FINITE}_{\text{DFA}}$ accepts $\langle C \rangle$, then accept; otherwise, reject.”

**Question 2a**

Let $\text{SORTED}_{\text{PDA}} = \{(M) : M \text{ is a PDA for which } L(M) \text{ is sorted}\}$. Prove or disprove: $\text{SORTED}_{\text{PDA}}$ is Turing decidable.

**Solution:** $\text{SORTED}_{\text{PDA}}$ is decidable. We use the same idea as in Question 1, by constructing a DFA that recognizes $\sigma_1^n \cdots \sigma_n^n$. We also use the answer to Exercise 2.18a, given on page 161 of the text, that shows that for any CFL $C$ and regular language $R$, then $C \cap R$ is a CFL (with the construction of the PDA also). As in Question 1a, we just need to check whether the resulting PDA’s language is empty. Since $E_{\text{CFG}}$ is decidable, we produce a decider $T$:
Therefore, by the Pumping Lemma for CFLs, we can partition $m$ such that $|m| > p$. By the Pumping Lemma for CFLs, we can partition $m_1 = uvwxy$ with $|vwx| \leq p$, $|vx| \geq 1$. Also, $m_2 = uwy \in L(G)$. We can see that $|m_1| - p \leq |m_2| < 2|m_1|$. We can use this argument repeatedly by producing a sequence of strings $\{m_1\}_{i \geq 1}$ such that $m_i \in L(G)$ for all $i$ and if $|m_i| > p$, we have that $|m_i| - p \leq |m_{i+1}| < 2|m_i|$. Therefore, we will eventually find a string $m_k$ such that $p < |m_k| \leq 2p$. For the other direction, suppose there is a string $m \in L(G)$ with $|m| > p$. Therefore, by the Pumping Lemma for CFLs, we can partition $m = uvwxy$ such that $|vxy| \leq p$, $|vx| \geq 1$, and $uv^iwv^jy \in L(G)$ for all $i \geq 0$. Since $|vx| \geq 1$, we have that $uv^iwv^jy \neq uv^iwv^jy$ for $i \neq j$. Therefore, $L(G)$ is infinite. All we need to decide $\text{FINITE}_{CFG}$ (and hence, $\text{FINITE}_{CFG}$) is to test each string of length $m$ for $p < m \leq 2p$. We create a decider $D_{finite}$:

$D_{finite} = \text{“On input } \langle G \rangle \text{ where } G = (V, \Sigma, R, S) \text{ is a CFG:}$$1. \text{ Let } b \text{ be the length of the longest right-hand side of any rule in } R, \text{ and } p = b|V|.$
$2. \text{ For } i \text{ from } p+1 \text{ to } 2p:$
   $\text{ (a) Run the decider } A_{CFG} \text{ on } (G, s) \text{ for all strings } s \text{ of length } i.$
   $\text{ (b) If } A_{CFG} \text{ accepts any string } s, \text{ accept. Otherwise, continue with the next string.}$
$3. \text{ If } A_{CFG} \text{ has rejected all strings } s, \text{ then reject.”}$

Therefore, $\text{FINITE}_{CFG}$ is the same as $\text{FINITE}_{CFG}$, except for swapping accept and reject. We now use the ideas from Question 1b here by constructing a decider $D$:

$D = \text{“On input } \langle M \rangle \text{ where } M \text{ is a PDA:}$$1. \text{ Suppose } \Sigma = \{\sigma_1, \ldots, \sigma_n\} \text{ with total ordering } \sigma_1 < \sigma_2 < \cdots < \sigma_n. \text{ Construct a DFA } D_{sorted} \text{ with } L(D) = \sigma_1^* \cdots \sigma_n^*.$
$2. \text{ Construct an equivalent DFA } D \text{ from } M.$
$3. \text{ Using the procedure of Exercise 2.18a, construct a PDA } P \text{ such that } L(P) = L(M) \cap L(D_{sorted}).$ 
$4. \text{ Construct an equivalent CFG } G \text{ from } P.$
$5. \text{ Run } \text{FINITE}_{CFG} \text{ on input } \langle G \rangle.$
$6. \text{ If } \text{FINITE}_{CFG} \text{ accepts } \langle G \rangle, \text{ accept; otherwise, reject.”}$

However, there is another way that $\text{FINITE}_{CFG}$ can be constructed:

$D_{finite} = \text{“On input } \langle G \rangle \text{ where } G = (V, \Sigma, R, S) \text{ is a CFG:}$$1. \text{ Convert } G \text{ into CNF.}$
$2. \text{ Remove any variables from } V \text{ that cannot generate a string entirely of terminals, and any rules in } R \text{ that contain them.}$
$3. \text{ Remove any variables from } V \text{ that cannot be reached.}$
$4. \text{ Create a digraph (directed graph) } D \text{ with the variables in } V \text{ as vertices, and for any rule } A \rightarrow BC, \text{ create edges } \{A, B\}, \{A, C\}.$
$5. \text{ If there is a directed cycle in } D, \text{ accept; otherwise reject.”}$

Question 3a

Suppose that $L$ is sorted. Prove or disprove: $L$ must be Turing decidable.

Solution: $L$ need not be decidable, as is implied by Question 3b below. However, a direct proof can be done in a similar fashion as Question 3b, but with $A_{TM}$ and $A_{TM, unary}$ (instead of the complements).
Question 3b

Suppose that $L$ is sorted. Prove or disprove: $L$ must be Turing recognizable.

**Solution:** if $L$ is sorted, then $L$ is not necessarily recognizable. Consider the language $\overline{\text{A}_\text{TM, unary}}$, which is the same as $\overline{\text{A}_\text{TM}}$ except the strings $\langle M, w \rangle$ are encoded in unary. Note that this language is sorted. Also note that for any $\langle M, w \rangle \neq \langle M', w' \rangle$ written in a fixed base, then non-equality still holds when converted to unary. We show that $\overline{\text{A}_\text{TM, unary}}$ is not recognizable by showing a reduction from $\overline{\text{A}_\text{TM}}$, which is not recognizable. Suppose (to the contrary) that this language were recognizable. Therefore, there is a TM, $M_{\text{unary}}$, that recognizes $\overline{\text{A}_\text{TM, unary}}$. There is also a TM, $M_{\text{convert}}$, that takes a pair $\langle M, w \rangle$ written in any base and converts it to unary. We design a TM $T$:

$T =$ “On input $\langle M, w \rangle$:
1. Run $M_{\text{convert}}$ on input $\langle M, w \rangle$ to construct $\langle M', w' \rangle$ (written in unary).
2. Run $M_{\text{unary}}$ on $\langle M', w' \rangle$.
3. If $M_{\text{unary}}$ accepts $\langle M', w' \rangle$, accept; otherwise, reject.”

Therefore, if $L$ is sorted, $L$ is not necessarily recognizable.

Question 4

Let $\text{SORTED}_{\text{TM}} = \{ \langle M \rangle : M$ is an TM for which $L(M)$ is sorted $\}$. Prove or disprove: $\text{SORTED}_{\text{TM}}$ is Turing decidable.

**Solution:** $\text{SORTED}_{\text{TM}}$ is undecidable through an application of Rice’s Theorem. To use this, we need to show two things: that “$L(M)$ is sorted” is a non-trivial property, and that for any TMs $M_1, M_2$ that have $L(M_1) = L(M_2)$ then either $\langle M_1 \rangle, \langle M_2 \rangle \in \text{SORTED}_{\text{TM}}$ or $\langle M_1 \rangle, \langle M_2 \rangle \notin \text{SORTED}_{\text{TM}}$. For the first, we just need to show a TM with the property “$L(M)$ is sorted,” and one without it. Let $\Sigma = \{a, b\}$, and let $M_1, M_2$ be TMs such that $L(M_1) = \{a\}$, and $L(M_2) = \{ab, ba\}$. No matter how the ordering between $a, b$ is defined, we have that $M_1 \in \text{SORTED}_{\text{TM}}$, and $M_2 \notin \text{SORTED}_{\text{TM}}$ (one of the strings $ab, ba$ must be unsorted). For the second, if $M_1, M_2$ are 2 TMs with $L(M_1) = L(M_2)$, then both languages are sorted or neither are. Therefore, by Rice’s Theorem, $\text{SORTED}_{\text{TM}}$ is undecidable.

Question 5

Let $\text{REVERSIBLE}_{\text{CFG}} = \{ \langle G \rangle : G$ is a context free grammar for which there exists a string $w \in L(G)$ for which $w = w^{\text{rev}} \}$. Prove or disprove that $\text{REVERSIBLE}_{\text{CFG}}$ is Turing decidable.

**Solution:** $\text{REVERSIBLE}_{\text{CFG}}$ is undecidable. Let $G_1, G_2$ be CFGs over alphabet $\Sigma$, and consider their languages: $L_1 = L(G_1), L_2 = L(G_2)$. Observe $L_1cL_2^{\text{rev}}$, where $c \notin \Sigma$, and $L^{\text{rev}} = \{w^{\text{rev}} \mid w \in L\}$. Note that $L_1cL_2^{\text{rev}}$ contains a palindrome iff $L_1 \cap L_2 \neq \emptyset$. Consider the language $\text{INTERSECT}_{\text{CFG}} = \{ \langle G, H \rangle \mid G, H$ are CFGs and $L(G) \cap L(H) \neq \emptyset \}$. Suppose $\text{REVERSIBLE}_{\text{CFG}}$ is decidable by a decider $R$. We construct a decider $D$ for $\text{INTERSECT}_{\text{CFG}}$:

$D =$ “On input $\langle G, H \rangle$ where $G, H$ are CFGs over alphabet $\Sigma$:
1. Construct CFG $G'$ such that $L(G') = L(G)cL(H)^{\text{rev}}$, where $c \notin \Sigma$.
2. Run $R$ on input $\langle G' \rangle$.
3. If $R$ accepts $\langle G' \rangle$, accept; otherwise, reject.”

Note that $L(G')$ is a CFL because CFLs are closed under reversal and concatenation. The reasoning for reversal is that we can convert $G'$ into CNF, and for every rule $A \rightarrow BC$, replace it with $A \rightarrow CB$. This shows that $L(G')$ as defined in $D$ is a CFL. Therefore, we have that $\text{INTERSECT}_{\text{CFG}}$ is decidable iff $\text{REVERSIBLE}_{\text{CFG}}$ is decidable. We show that $\text{INTERSECT}_{\text{CFG}}$ is undecidable, thereby proving that $\text{REVERSIBLE}_{\text{CFG}}$ is undecidable. To show this, we use computation histories of TMs. We introduce two lemmas:

- **Lemma 1:** let $M$ be a TM. Consider the language:
  \[
  L_{M,2} = \{C_1\#C_2^{\text{rev}} \mid C_1, C_2 \text{ are valid configurations of } M \text{ and } C_2 \text{ is implied by } C_1 \text{ using } M's \text{ rules}\}.
  \]

  Then $L_{M,2}$ is a CFL.

  **Proof:** assume $M$ has tape alphabet $\Gamma$ and states $Q$. The following CFG recognizes $L_{M,2}$:

  \[
  S \rightarrow xSx, \text{ for all } x \in \Gamma \\
  S \rightarrow B \\
  B \rightarrow qaIq'b, \text{ for all } a, b \in \Gamma, q, q' \in Q \text{ with } \delta(q, a) = (q', b, R) \\
  B \rightarrow aqbIacc', \text{ for all } a, b, c \in \Gamma, q, q' \in Q \text{ with } \delta(q, b) = (q', c, L) \\
  I \rightarrow xIx, \text{ for all } x \in \Gamma \\
  I \rightarrow \#
  \]
Note that the derived part of the string corresponding to $C_2$ is reversed of what the TM is doing. The intuition is the following. Between $C_1$ and $C_2$, only a few cells are modified, and the tape head can only move right or left (and only once). The $S$ variable tracks all characters at the ends of the configurations that are the same, $B$ handles the changes between the 2 configurations, and $I$ handles the characters in between.

**Lemma 2**: let $M$ be a TM, and $w$ a string. Consider the language:

$$L_{M,w} = \{ \#C_1\#C_2^{\text{rev}}\#C_3\#\cdots\#C_{\ell}\# | \#C_1\#C_2^{\text{rev}}\#C_3\#\cdots\#C_{\ell}\# \text{ is an accepting computation of } M \text{ on } w \}.$$

Then, we can write $L_{M,w}$ as the intersection of 2 CFLs.

**Proof**: set $L_1 = \#q_0w_1\cdots w_n\#(\Gamma \cup \{\#\})^*$ (i.e., the set of strings with a valid starting configuration); $L_1$ is regular. Let $L_2$ be the set of computation histories, defined as:

- $C_{2i-1}$ in odd position $2i - 1$, and
- $C_{2i}^{\text{rev}}$ in even position $2i$, as implied by $C_{2i-1}$ by the rules of $M$’s transitions.

This is a CFL, as we can just make the CFG in the proof of Lemma 1 longer to handle more configurations. Let $L_3$ be the same as $L_2$, but the configuration $C_{2i+1}$ is implied by $C_{2i}^{\text{rev}}$. Let $L_4 = (\Gamma \cup \{\#\})^*\#q_{\text{accept}}\Gamma^*\#$ (i.e., the set of strings with an accepting configuration at the end); this language is regular. Since $L_1, L_4$ are regular, and since CFLs are closed under intersection with regular languages, $L' = L_1 \cap L_2 \cap L_4$ is a CFL. The language we want is $L' \cap L_3$, which is the intersection of 2 CFLs.

We show that Lemmas 1 and 2 together prove that $\text{INTERSECT}_{\text{CFG}}$ is undecidable. Suppose (to the contrary) that it were decidable. Given $\langle M, w \rangle$, we can decide whether $L_{M,w}$ is empty or not, since it is the intersection of 2 CFLs: $L_1, L_2$. If $L_1 \cap L_2 \neq \emptyset$, then $M$ accepts $w$; if $L_1 \cap L_2 = \emptyset$, then $M$ does not accept $w$. Therefore, we have a decider for $A_{TM}$, which is a contradiction. Therefore, $\text{INTERSECT}_{\text{CFG}}$ is undecidable.