Problem 1. Draw a Turing machine that decides the language

\[ A = \{0^k1^m0^k \mid 0 \leq k < m\}. \]

You may assume \( \Sigma = \{0,1\} \).

Solution 1. In the below Turing machine, we mark off 0’s and 1’s as trios. Then when we run out of 0’s, we accept iff we still have a 1 left. Note that missing transitions cause the machine to reject.

Problem 2. Recall that a computable function \( f : \Sigma^* \rightarrow \Sigma^* \) is one for which there is a Turing machine \( M \) such that on all inputs \( \omega \in \Sigma^* \), \( M(\omega) \) halts with just \( f(\omega) \) on its tape.

Show that the function \( f(x) = 2x \) is computable by drawing a Turing machine for it. Here, \( x \) is a binary number (with its most significant bit given first) and \( f \)’s output in binary should be left on the tape (in the same format). You may assume that the input alphabet is \( \Sigma = \{0,1\} \).
**Solution 2.** For this problem, multiplication by 2 in binary is just adding a 0 to the end of it. To keep the answer simple, we will also interpret $2 \epsilon$ as 0.

![](image1.png)

**Problem 3.** Prove that if a language is decidable, then there is an enumerator which enumerates that language in lexicographical order.

**Solution 3.** Let $L \in DEC$. Then there is a decider $D$ for which $L(D) = L$. We give an enumerator $E$ with $L(E) = L$ that prints $L$ in lexicographical order as follows.

$E = \text{Ignore input,}$
1. For $i = 0$ to $\infty$
   (a) Run $D(\omega_i)$, where $\omega_i$ is the $i^{th}$ string of $\Sigma^*$ in lexicographical order
   (b) If $D$ accepts, print $\omega_i$

Since $D$ is a decider, it will never loop forever. Thus $E$ will never get stuck on any string. Moreover, $E$ clearly prints out each string $D$ accepts. Thus $L(E) = L$. Moreover, $E$ clearly prints its language in lexicographical order.

**Problem 4.** A **linear bounded automaton** (LBA) is a restricted Turing machine. Instead of having a tape of unlimited length, it has a tape of length linear in its input. In many ways, this is a more realistic model of computation than Turing machines as our physical computers work with limited memory. The class of languages which LBAs recognize are the context-sensitive languages. They fall between the context-free and the decidable languages in terms of computational complexity.

An equivalent definition of an LBA is to impose a further restriction. Instead of having a linear length tape, the input is given between a left end marker and a right end marker. The Turing machine must not move past these end markers and it must not overwrite them. It is otherwise allowed to modify the tape where the input is given.

Define the analogous set to $A_{TM}$ for linear bounded automata as

$$A_{LBA} = \{ \langle M, \omega \rangle \mid M \text{ is an LBA and } M(\omega) \text{ accepts} \}.$$ 

Prove that $A_{LBA}$ is decidable.

**Solution 4.** The key realization here is that an LBA has a finite number of configurations it can take. If it ever enters a configuration it has been in before, it has entered an infinite loop. We can thus decide $A_{LBA}$ as follows.

$D = \text{On input } \langle M, \omega \rangle,$
1. Run $M(\omega)$
2. For each step of the simulation, do the following
   (a) Record the configuration of $M$
(b) If $M$ is in an accepting configuration, accept
(c) If $M$’s current configuration is equal to any of its previous configurations, reject

Clearly $D$ accepts $\langle M, \omega \rangle$ iff $M$ accepts $\omega$, so $L(D) = A_{LBA}$. Moreover, $D$ is guaranteed to halt since $M$ can only enter a finite number of distinct configurations, so $D$ rejects all other inputs. Thus $D$ decides $L_{LBA}$. □

**Problem 5.** For both of the following languages, prove it is undecidable using Rice’s Theorem or show that Rice’s Theorem does not apply. You may assume that $\Sigma = \{0, 1\}$ for each problem.

a) $B = \{\langle M \rangle \mid M$ is a TM and $10 \in L(M)\}$

b) $C = \{\langle M \rangle \mid M$ is a TM, $L(M) = 0^*$, and $M$ visits at most 3 distinct states\}$

**Solution 5.**

a) Rice’s Theorem applies to this problem (the property is $10 \in L(M)$). To see why, notice that $C$ does not contain any TM which accepts no strings but it does contain any TM which accepts the string 10. As such, $C$ is nontrivial. Moreover, if $M_1$ and $M_2$ are Turing machines with $L(M_1) = L(M_2)$, then $\langle M_1 \rangle \in C$ iff $10 \in L(M_1)$ iff $10 \in L(M_2)$ iff $\langle M_2 \rangle \in C$. So $C$ is a property of the TM’s language. Together, this shows that Rice’s Theorem applies, so $C \notin DEC$. □

b) Rice’s Theorem does not apply to this problem. To see why, consider the TM $M$ given below.

```
\begin{tikzpicture}

\node (q) at (0,0) [circle, draw] {$q$};
\node (qaccept) at (1,-1) [circle, draw] {$q_{accept}$};
\node (qreject) at (-1,-1) [circle, draw] {$q_{reject}$};
\node (square) at (0,-2) [circle, draw] {$\square$};

\draw[->] (q) edge [loop above] node {$0 \rightarrow R$} (q);
\draw[->] (q) edge [loop below left] node {$\square \rightarrow R$} (square);
\draw[->] (q) edge [loop below right] node {$1 \rightarrow R$} (square);
\draw[->] (q) edge node {$R$} (qaccept);
\draw[->] (q) edge node {$R$} (qreject);
\end{tikzpicture}
```

Clearly, $L(M) = 0^*$ and visits at most 3 states. So $\langle M \rangle \in D$.

We could now add 42 chained useless transitions through 42 new states from a new start state. These states do nothing, only attempting to move the head left off the tape and failing. They end in the state $q$. This TM, which we will call $N$, clearly visits more than 3 states but still decides the language 0*. Then $\langle N \rangle \notin D$.

This means that $D$ is not a property of the TM’s language, so Rice’s Theorem does not apply. □

**Problem 6.** Prove that $A_{TM} \leq_m E_{TM}$.
Solution 6. We know that $A_{TM} \notin \text{co-RE}$. If $A_{TM} \leq_m E_{TM}$, then this would imply that $E_{TM} \notin \text{co-RE}$. But we also know that $E_{TM} \in \text{co-RE}$, so it must be the case that $A_{TM} \not\leq_m E_{TM}$.

Problem 7. Show that if $L \in \text{RE}$ and $L \leq_m \overline{L}$, then $L \in \text{DEC}$.

Solution 7. Suppose that $L \in \text{RE}$ and that $L \leq_m \overline{L}$. Then we also know that $\overline{L} \leq_m L$. Since $L \in \text{RE}$, it follows that $\overline{L} \in \text{RE}$. But if $\overline{L} \in \text{RE}$, by definition, $L \in \text{co-RE}$. Then since $L \in \text{RE}$ and $L \in \text{co-RE}$, $L \in \text{DEC}$.

Problem 8. Show that if $A \leq_m B$ and $B \leq_m C$, then $A \leq_m C$.

Solution 8. Suppose $A \leq_m B$ and $B \leq_m C$.

By definition, there are computable functions $f_1$ and $f_2$ such that for all $\omega$, both
\[ \omega \in A \iff f_1(\omega) \in B, \]
\[ \omega \in B \iff f_2(\omega) \in C. \]

But then this gives us the following for all $\omega$.
\[ \omega \in A \rightarrow f_1(\omega) \in B \rightarrow f_2(f_1(\omega)) \in C \]
\[ \omega \notin A \rightarrow f_1(\omega) \notin B \rightarrow f_2(f_1(\omega)) \notin C \]

In other words,
\[ \omega \in A \iff f_2 \circ f_1(\omega) \in C. \]

Moreover, the function $f_2 \circ f_1$ is clearly computable. We need only successively apply the Turing machines which compute $f_1$ and then $f_2$ to compute their composition. Thus $A \leq_m C$.

Problem 9. Consider the following language.

$ODD_{TM} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) \text{ contains no even length strings} \}$

Prove that $ODD_{TM} \notin \text{RE}$ via a mapping reduction and $ODD_{TM} \in \text{co-RE}$.

Solution 9. To show that $ODD_{TM} \in \text{co-RE}$, it suffices to give a Turing machine $O$ which recognizes $\overline{ODD_{TM}}$.

$O = \text{On input } \langle M \rangle$,

1. For $i = 1$ to $\infty$
   (a) Run $M$ on the first $i$ even length strings of $\Sigma^*$ for $i$ steps
   (b) If $M$ accepts any such string, accept

Clearly, $O$ accepts no Turing machines which do not accept an even length string. Moreover, if $M$ accepts an even length string, it is the $k^{th}$ such string of $\Sigma^*$. Then when $i \geq k$, we will run $M$ on that string. Eventually, we will run $M$ on that string long enough for $M$ to halt and accept it in some finite number of steps. So $O$ will accept $M$. Thus $L(O) = \overline{ODD_{TM}}$, and $\overline{ODD_{TM}} \in \text{RE}$. Consequentially, $ODD_{TM} \in \text{co-RE}$.

To show that $ODD_{TM} \notin \text{RE}$, we give a mapping reduction to it from $E_{TM}$. We will need a utility Turing machine $E$ to do so.

$E_M = \text{On input } \omega,$
1. If $|\omega|$ is odd, reject
2. Let $\nu$ be the first half of $\omega$
3. Run $M(\nu)$
4. If $M$ accepts, accept; otherwise, reject

Notice that $\nu$ will take on every value of $\Sigma^*$ for at least one input $\omega$. To get $\nu \in \Sigma^*$, pick $\omega = \nu \nu$. Then if $M$ accepts any string $\nu$, $E_M$ will accept $\nu \nu$ (an even length string). Moreover, if $M$ accepts no strings, then $E_M$ will reject every even length string. This means that $E_M \in \text{ODD} \iff M \in E_{TM}$.

From this if and only if statement, we arrive in the usual way at the reduction $f(\langle M \rangle) = \langle E_M \rangle$. Thus $E_{TM} \leq_m \text{ODD}$. And since $E_{TM} \notin \text{RE}$, it follows that $\text{ODD} \notin \text{RE}$. □

**Problem 10.** Recall that for a function to be computable, there must be a Turing machine which computes it. This applies to functions whose inputs and outputs are natural numbers as well since there is a (computable) bijection between $\Sigma^*$ and $\mathbb{N}$. This allows us to state some theorems more naturally.

The set of all computable functions $f : \mathbb{N} \rightarrow \mathbb{N}$ is countable However, the set of all functions $f : \mathbb{N} \rightarrow \mathbb{N}$ is uncountable. Consequentially, almost all such functions are not computable. This is the same argument we made to show that almost all languages are not in $\text{RE}$ or $\text{co-RE}$. Beyond this, there are many other strange properties of uncomputable functions that may surprise you.

Prove that there exists a function which grows faster than any computable function. In other words, show that there is a function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that for any computable function $g : \mathbb{N} \rightarrow \mathbb{N}$ there exists an $N \in \mathbb{N}$ such that for all $n \geq N$, $f(n) > g(n)$.

More succinctly, show $\exists f : \mathbb{N} \rightarrow \mathbb{N} : \forall g : \mathbb{N} \rightarrow \mathbb{N}$ computable $\forall^\infty n, f(n) > g(n)$.

(Hint: Approach this similarly to dovetailing by overcoming one additional computable function at a time)

**Solution 10.** Since the computable functions are countable, let $g_1, \ldots, g_n, \ldots$ be an ordering of them. Define the function $f : \mathbb{N} \rightarrow \mathbb{N}$ to be

$$f(n) = 1 + \max_{1 \leq i \leq n} (g_i(n)).$$

Then for each $n \in \mathbb{N}$, it follows that $f(n) > g_i(n)$ for all $1 \leq i \leq n$. It follows that for each computable function $g_i$, there exists an $N = i$ such that for all $n \geq N$, $f(n) > g_i(n)$. Thus $f$ grows faster than any computable function. □