**Exercise 1.**  
*Warming up.*

Let $a, b$ be integers in their binary representation.

**Definition (recall) Turing Machine (one definition, there are many others).** A $k$-tape Turing Machine (TM) is described by a triple $M = (\Gamma, Q, \delta)$ containing:

- A finite set $\Gamma$, called the tape alphabet, that contains symbols that the TM uses in its tapes. In particular, $\Gamma$ contains a blank symbol "□", and "⊿" that denotes the beginning of a tape.

- A finite set $Q$ called the states of the TM. It contains special states $q_{\text{start}}$, $q_{\text{halt}}$, called respectively the initial state and the halt state.

- A function $\delta : (Q \setminus \{q_{\text{halt}}\}) \times \Gamma^{k-1} \rightarrow Q \times \Gamma^{k-1} \times \{←, ↓, →\}^k$, called the transition function, that describes the behaviour of the internal state of the machine and the TM heads. Namely, $\delta(q, a_1, \ldots, a_{k-1}) = (r, b_2, \ldots, b_k, m_1, \ldots, m_k)$ means that upon reading symbols $(a_1, \ldots, a_{k-1})$ on tapes 1 to $k - 1$ (where the first tape is the input tape, and the $k$-th tape is the output tape) on state $q$, the TM will move to state $r$, write $b_2, \ldots, b_k$ on tapes 2 to $k$ and move its heads according to $m_1, \ldots, m_k$.

A TM $M$ is said to compute a function $f : \Sigma^* \rightarrow \Gamma^*$, if for any finite input $x \in \Sigma^*$ on tape $T_1$, blank tapes $T_2, \ldots, T_k$ with a beginning symbol ⊿ and initial state $q_{\text{start}}$, $M$ halts in a finite number of steps with $f(x)$ written on its output tape $T_k$.

Similarly to finite automata, a TM can be written as a graph where the vertices represent the states and edges represent the transitions.

1. Write the full description of a Turing machine that performs addition on input $a\#b$.

2. multiplication on input $a\#b$.

**Exercise 2.**  
*Change the Model.*

**Definition.** A Post Machine (PM, or queue automaton) $M$ is a finite state machine with a single tape of unbounded length with FIFO access: in a single transition, $M$ reads and deletes the symbol at the head of the FIFO queue and may append symbols to the tail of the queue.

1. Write the description of a PM $M$ that translates a text in $\Sigma_1 = \{a, \ldots, z\}$ into a Morse text in $\Sigma_2 = \{-, \cdot, _\}$, where _ denotes the long wait separating two characters. (If you don’t have enough space on your sheet, you can restrict to the subset $\{a \rightarrow \cdot-; d \rightarrow -\cdot\cdot; e \rightarrow \cdot\}$).

2. Describe a PM that performs string duplication. That is, show a PM that transforms an input string, for instance 001, into this string written twice: 001001.

3. Show that a Post Machine can be simulated by a Turing Machine.
4. Show that a Turing Machine can be simulated by a Post Machine.
What is (somewhat) surprising with this result?

We will show the following crucial result:

**Theorem.** There exists a TM $U$ such that for every $x, \alpha \in \{0, 1\}^*$, $U(x, \alpha) = M_\alpha(x)$ where $M_\alpha$ denotes the TM represented by $\alpha$.
Moreover, if $M_\alpha$ halts on input $x$ within $T$ steps, then $U(x, \alpha)$ halts within $O(T \log T)$ steps. Remark that the complexity of $U$ only depends on properties on $M$ and not on $|x|$.

1. Give a simple way of simulating a two-tapes machine with a one-tape machine. Why is it an issue in our case?
2. What is the cost of simulating a machine on alphabet $\Gamma_k$ with a machine on alphabet $\Gamma$?
3. Now, to simulate a $k$-tape machine on alphabet $\Gamma$, we won’t simulate the move of each head sequentially, but we will work on $\Gamma_k$ to simulate the transition of the tapes simultaneously. Show how to simulate a $k$-tape TM on alphabet $\Gamma$ by a 1-tape TM on alphabet $\Gamma_k$ in time $O(T(n) \log T(n))$.
4. Describe the universal Turing machine $U$ corresponding to the theorem.

Exercise 4. Aibohphobia.
For $x \in \Sigma^*$, let us denote by $\bar{x}$ the mirror of $x$ (e.g. $\text{flow} = \text{wolf}$). The palindrome language over $\Sigma$ is defined as

$$\text{PAL} = \{x \in \Sigma^* : x = \bar{x}\}.$$

1. Describe a TM with two tapes that recognizes $\text{PAL}$ in linear time.
2. Describe a TM with one tape that recognizes $\text{PAL}$ in quadratic time.

All the following TMs are assumed to have one half-infinite tape, where cells are indexed starting from 0.

**Definition.** The crossing sequence $C_i(x)$ at boundary $i$ of a $M$ on input $x$ is defined as the sequence of states of $M$ when the head crosses between cells $i$ and $i + 1$.

Let $M$ be a TM recognizing $\text{PAL}$. Let $\text{PAL}_n$ the sublanguage of $\text{PAL}$ defined as

$$\text{PAL}_n = \{x_i^{2n} \bar{x} : x \in \Sigma^n\},$$
assuming that $0 \in \Sigma$. Let us define

$$C(x) = \{C_i(x) : n \leq i \leq 3n\}.$$

1. Show that for every $x, y \in \text{PAL}_n$ such that $x \neq y$, $C(x) \cap C(y) = \emptyset$.
2. For $x \in \text{PAL}_n$, let $m_x$ be the shortest crossing sequence of $C(x)$, and let $m = \max\{|m_x| : x \in \text{PAL}_n\}$. Let $Q$ be the set of sets of $M$. Show that

$$\frac{|Q|^{m+1} - 1}{|Q| - 1} \geq 2^n.$$

3. Conclude that the complexity of $M$ is $\Omega(n^2)$.
Exercise 5.

Definition. A Markov algorithm is an ordered string rewriting system \( P \) over the alphabet \( \Sigma \), written \( P = [ \alpha_1 \to \beta_1; \ldots ; \alpha_k \to \beta_k ] \) along with a subset \( F \subseteq P \) of terminal rules.

On input \( u \in \Sigma^* \), a potentially infinite unique string sequence \( u \to u_1 \to u_2 \to \ldots \to u_n \to u_{n+1} \to \ldots \) is defined for \( u \to u_{j+1} \) as the application on the leftmost instance of an \( \alpha_i \) on the current string \( u_j \) of the first rule that can apply in \( P \).

The algorithm stops on step \( n-1 \) and returns \( u_n \) if the last step \( u_{n-1} \in F \) and no rules can apply on \( u_n \). Otherwise the algorithm does not terminate.

1. What does the following Markov algorithm do?

\[
P = [10 \to 011; \ 1 \to 01; \ 0 \to \varepsilon] \quad F = [0 \to \varepsilon]
\]

2. Write a Markov algorithm computing the successor of a binary integer.

3. Show that a Markov algorithm can be simulated by a Turing machine.

4. Show that a Turing Machine can be simulated by a Markov Algorithm.

Exercise 6.

Let \( \text{PRIME} = \{ n : n \text{ is prime} \} \). Show that \( \text{PRIME} \in \text{NP} \).

We recall that if there exists \( a \in \{2, \ldots, n-1\} \) such that \( a^{n-1} = 1 \mod n \) and for each prime factor \( q \) of \( n-1 \), \( a^{(n-1)/q} \neq 1 \mod n \), then \( n \) is prime.

---

1You are not allowed to invoke [AKS02] unless you are able to prove it.