Computability and Complexity

Instituto de Matemáticas

Exercises

The Church-Turing Thesis

Exercise 1 : ***Understanding the definitions.***
Based on the formal definition of a Turing machine, answer the following questions and justify your answers:

1. Can a Turing machine write the blank symbol ␣ on its tape?
2. Can the tape alphabet Γ and the input alphabet Σ be equal?
3. Can a Turing machine's head be in the same location in two successive configurations?
4. Can a Turing machine contain a single state?

□

Exercise 2 : ***A weird algorithm.***
Explain why the following is not a description of a Turing machine:

$M_{bad} =$ The input is a polynomial p over variables $x_1, \ldots, x_n$.

1. Try all possible settings of $x_1, \ldots, x_n$ to integer values.
2. Evaluate p on all those settings.
3. If any of these settings evaluate to 0, *accept*; otherwise, *reject*.

□

Exercise 3 : ***Shifting the blame.***
Let w be a word over an alphabet Σ such that $\# \notin \Sigma$. Construct a formal-level Turing machine that takes input w and enters the accept state once its tape contains $\#w$. Explain why this machine is useful.

□

Exercise 4 : ***Reusing machines.***
Let w be a word over an alphabet Σ such that $\# \notin \Sigma$ and such that w is of even length and not empty. Give the implementation-level description of a Turing machine that takes input w and enters the accept state once its tape contains the word where # is inserted in the middle of w.
Hint: You may use several tapes, and reuse the machine of Exercise 3.

□

Exercise 5 : ***Languages.***
Give implementation-level descriptions of Turing machines that decide the following languages:

1. $\{w \in \{a, b\}^* \mid w \text{ contains as many } a \text{ as } b\}$;
2. $\{a^n b^n c^n \mid n \geq 0\}$;
3. $\{a^n b a^{2n} b a^{3n} \mid n \geq 0\}$.

Draw a formal-level implementation of one of those machines.

□

Exercise 6 : ***Turing's elementary school.***
Give implementation-level descriptions of Turing machines that compute the following functions (in every case, we assume the numbers are not empty):

1. A function that takes a binary number, and deletes every useless 0 (so every 0 before the first 1);
2. The increment function on binary numbers (the input is a binary number w, and we want to compute $w+1$);
3. The decrement function on binary numbers (the input is a binary number w, and we want to compute $w-1$) (assume $w \neq 0$);
4. The binary-to-unary conversion function (the input is a binary number w, and we want to compute the unary number equal to w);
5. The binary addition function (the input is $w_1\#w_2$ where w_1 and w_2 are binary numbers, and we want to compute $w_1 + w_2$);
6. The binary multiplication function (the input is $w_1\#w_2$ where w_1 and w_2 are binary numbers, and we want to compute $w_1 w_2$).

You may use several tapes, and reuse machines that you already described or constructed.

□

Exercise 7 : ***Several stacks.***
For this exercise, I assume that you know everything that we saw in the class and exercises on context-free languages.

1. Prove that a pushdown automata with two stacks is more powerful than a pushdown automata with one stack.
2. Prove that you can simulate a Turing machine with a pushdown automata with two stacks.
3. What does that imply for pushdown automatas with more than two stacks?

□

Exercise 8 : ***The Double Infinity Gauntlet.***
A Turing machine with a doubly-infinite tape is a Turing machine where the tape does not have a left end. If you imagine the tape of a Turing machine as a table with indices in the set of natural numbers $\mathbb{N}$, then the doubly-infinite tape is a table with indices in the set of integers $\mathbb{Z}$. The computation is exactly the same, but the head will never encounter the leftmost end of the tape.

Prove that the Turing machines with a doubly-infinite tape recognizes the class of Turing-recognizable languages.

□