COMS 3261, Computer Science Theory (Fall 2023): Assignment 3
Due on Gradescope 11:59pm, Monday Nov 20, 2023

Instructions

- The total number of points is 55 and there are two pages. Submit your solutions in pdf format. Late homeworks will **not** be accepted.
- You can discuss with TAs, the prof, and other students, but please acknowledge them at the beginning of each problem. All solutions must be written in your own words.
- You should be able to solve questions 1-4a already; the material required to solve questions 4b and 5 will be covered Nov 8 and Nov 13.

Problems

1. (10 points) Give a **formal description** of a one or two tape input-output Turing machine that takes a string $w \in \{0,1\}^*$ as input and halts with w^R (the reverse of the string w) on the first tape. For example, if the tape initially contains the input "11001" (followed by blanks), then after halting the tape should be contain '10011' (followed by blanks). Include a brief high level description of your TM.

2. (10 points) Prove that for every infinite set S, the following are equivalent:

   (i) There exists a function $g : \mathbb{N} \to S$ that is onto (i.e., g is surjective).

   (ii) There exists a function $f : S \to \mathbb{N}$ that is one-to-one (e.g., f is injective).

3. (10 points) Consider the language L_{DFA} which accepts an input w if w is an encoding of a DFA A, where A accepts at least one input $x \in \{0,1\}^*$.

   a. Prove that L_{DFA} is decidable by giving a **high-level description** of a TM that always halts and that accepts exactly the strings in L_{DFA}.

   b. In one or two sentences, explain what goes wrong with your algorithm if you were to apply similar ideas to try to prove that L_{TM} is decidable (where L_{TM} accepts the set of encoding of TMs that accept at least one input).

4. (15 points) Let L_{pair} be the set of all encodings of Turing machines $<M>$ such that there is a pair of consecutive binary numbers that are both accepted by M. (For example, if a Turing machine M accepts both 100 and 101 then $<M>$ is in L_{pair}.)

   a. Prove that L_{pair} is recognizable (r.e.) by giving a **high-level description** of a Turing Machine that accepts L_{pair}.

   b. Prove that L_{pair} is not decidable (not recursive).

5. (10 points) Let $L_{\text{Prime}} = \{<M> \mid \text{the number of strings accepted by } M \text{ is prime}\}$. Classify this language as either (i) decidable, (ii) recognizable but not decidable, or (iii) not recognizable. Prove your answer. You may give a **high-level description** of any TM programs used in your proof. (You should not use Rice's theorem.)

Instructions on level of detail required for TM constructions

Below is a description of the level of detail expected for describing TMs and programs. Please read the section in Sipser book on "Terminology for Describing TMs" for more information.

- **Formal Level Description:** Unless otherwise mentioned, you should use the standard definition of a TM (with a single tape, and a single head that can move either left or right in each step). Provide all transitions in detail. (Note in the first question you are asked to give a formal level description of an input-output TM, but you are allowed to use either a single-tape TM or a two-tape TM. If you give a two-tape TM, it should again be the standard one, with two tapes, and two heads, each of which can move either left or right at each step. And again you should provide all transitions in detail.)
- **Implementation Level Description:** You may use any of the equivalent models of TMs discussed in class or in the book (e.g., multitape TMs, nondeterministic TMs). You can use low-level subroutines like the ones you saw in class (e.g., insert a blank symbol and shift rest of the tape content to the right; head can move left or right or stay in place at each step).
- **High Level Description:** You do not need to discuss the mechanics of an actual TM (tapes, states, the transition function). Instead you can present your algorithm in pseudocode. Pseudocode is a simpler version of a programming code that is given in plain English which uses short phrases to write code for a program before it is implemented in a specific programming language. Make sure you algorithm is clear and well defined (e.g., what is the input to the program, what are the main subroutines/loops/steps; specify the data structure(s) being used (variables, arrays, etc).