COMS 4236: Introduction to Computational Complexity, Spring 2018

Problem Set 1, due Thursday February 8, 11:59pm on Courseworks

Please follow the homework submission guidelines posted on Courseworks.

*All the Turing machines in this problem set are deterministic multitape Turing machines. In your constructions, give clear precise descriptions in English of your Turing machines and explain how they operate; you do not need to give full formal specifications. Questions that may be more challenging are marked with a *.*

Problem 1. [10 points] Consider a Turing machine M that computes the length of its input in binary. Specifically the TM M has an input tape and a work tape. Initially the input tape contains (after the left endmarker) an input string x over some input alphabet A, and the work tape is initially blank after the left endmarker. At the end, the TM halts with the work tape containing the length $n=|x|$ of its input string x, in binary.
The TM M first initializes the work tape to 0, and then repeatedly it moves the input head one cell to the right and increments the number on the work tape by 1, until the input head reaches a blank symbol, at which point M halts. In each iteration, the work tape head starts at the right end and travels only as far left as it needs to perform the increment, and then returns to the right end.
Show that the time complexity of the TM M is O(n).
(Hint: Although some iterations may take time close to log$n$, the total time is O(n), not $n$log$n$. Observe that roughly ½ of the numbers less than n are even, i.e. their binary representation ends in 0; ¼ of the numbers have in their binary representation only one trailing 1, etc. Note also that $\sum_{i=1}^{\infty} i \cdot 2^{-i} = 2$.)

Problem 2. [15 points] Show that the class of languages in TIME($f(n)$) (for any function $f(n)$) is closed under the operations of (a) complementation, (b) union, and (c) intersection. That is, show: (a) If L is a language over alphabet A that can be decided in time O($f(n)$), then its complement $\bar{L} = A^* - L$ is also decidable in time O($f(n)$); (b) if $L_1, L_2 \in \text{TIME}(f(n))$ then $L_1 \cup L_2 \in \text{TIME}(f(n))$; (c) if $L_1, L_2 \in \text{TIME}(f(n))$ then $L_1 \cap L_2 \in \text{TIME}(f(n))$.

Problem 3. [25 points] In a standard Turing machine, every tape has one head. A *multihead* Turing machine is allowed to have any constant number of heads (one or more) for each tape; i.e. such a TM has a constant number k of tapes and a constant number $l \geq k$ of heads, numbered 1,…,l, with each head assigned to one tape. The input tape and the work tapes may be assigned more than one heads; the output tape (if the TM computes a function) is only allowed one head. Initially every head is at the left endmarker of its tape. The transition function of the TM maps each state and *l*-tuple of symbols read by the heads to a new state and an *l*-tuple of new symbols written and directions (Right, Left, Stationary) of movement for the heads. If two or more heads happen to be on the same tape cell, then the symbol specified by the lowest numbered head is the one that is actually written there. For the computation of functions, the output tape is still restricted to have only one head that never moves left and writes only once on each cell. Time and space complexity are defined for multihead Turing machines in the same way as with standard machines; in particular, for sublinear space bounds, the input tape is restricted to be read-only and we count only the space of the work tapes.

a. The multihead feature makes some tasks much simpler.
Describe carefully in English a multihead Turing machine that recognizes palindromes in linear time and 0 space, i.e. without using any work tapes, just a read-only input tape.

b*. Show that every multihead Turing machine with space complexity S(*n*) can be simulated by a standard Turing machine with space complexity O(S(*n*)+log*n*).
How does the time complexity change in your simulation? Express the time complexity of your standard TM as a function of the time and space complexity of the multihead TM. (Hint: Add tapes that keep track of the positions of the heads on the input and the work tapes of the multihead TM.)

Problem 4. [30 points] We want to design a (standard) input-output Turing machine that adds two given binary numbers. Specifically, your TM must have an input tape, an output tape and any number of work tapes. Initially the input tape contains (between a left and a right endmarker) an input string *a*#*b* where *a*,*b* are nonempty binary strings that represent two nonnegative numbers, from most significant to least significant bit (the strings do not have any leading zeroes, i.e. each starts with 1 unless it is equal to 0); thus the input alphabet is {0,1,#}. At the end of the computation, the TM must halt with a binary string *c* on its output tape that represents *a*+*b*. If the input string does not have the right format, then the TM should print # on its output tape. As usual in an i-o TM, the output head cannot move left and cannot overwrite a previously written symbol; the input head can move both ways but cannot overwrite any symbol.

a. Describe carefully in English a Turing machine that runs in O(*n*) time, where *n* is the size of the input, i.e. the number of bits of the input numbers +1. The TM should print in the output tape the sum *a*+*b* from most significant to least significant bit. What is the (asymptotic) space complexity of your TM?

b. Describe a Turing machine with space complexity O(log*n*) that outputs the sum from least significant to most significant bit.

c*. Repeat part b, but now output the sum from most significant to least significant bit.
Estimate in both cases b and c the (asymptotic) time complexity of your TMs.
(You may use Problem 3 in parts b and c if it helps you, or you can give direct constructions.)

Problem 5. [20 points] An *online* Turing machine is a multitape Turing machine with a read-only input tape (with left and right endmarkers) whose head is restricted to never move left, i.e. stays in the same place or moves right. This model is useful for studying problems in a “streaming data” context (massive data streaming through a processor). Variants are also useful for studying the complexity of computations with large data stored in external memory.
In this problem you will show that the recognition of palindromes (over any alphabet with two or more symbols) requires $\Omega(n)$ workspace in the online Turing machine model, i.e. every online TM that recognizes palindromes must use (at least) $\Omega(n)$ work space (whereas we saw in class that $O(\log n)$ space is sufficient without the online restriction). To this purpose, consider an online Turing Machine M. Define the *working configuration* of the TM M at any time to consist of the state, the contents of the worktapes and the positions of the worktape heads only (not the input tape).

a. Argue that if there are two different strings $x \neq y$ of length $n/2$ such that the online TM M is in the same working configuration after reading prefix x of an input as it is after reading prefix y of another input, then the language accepted by M is not the set of palindromes.
(Hint: Consider the behavior of M with input $x\,x^R$, where x^R is the reverse of string x, and the behavior of M with input $y\,x^R$.)

b. Bound the number of different working configurations of a Turing Machine with (work) space $S(n)$ in terms of the space and the number of states, tapes, and tape symbols. (Note: the number of states, tapes, and tape symbols is constant).

c. Use parts a and b to show that any online Turing machine that recognizes the set of palindromes over an alphabet with two or more symbols must use (work) space $\Omega(n)$.

Note: Clearly, for space bounds $\geq n$, the online restriction has no real effect because we can copy the input to a work tape and apply any standard Turing machine algorithm.