COMS 4236: Introduction to Computational Complexity, Spring 2012

Problem Set 1, due Thursday February 15, in class

All the Turing machines in this problem set are deterministic multitape Turing machines.

Problem 1. Construct a Turing machine that counts (in binary) the length of its input. Specifically the TM M should have an input tape and a work tape. Initially the input tape contains (after the left endmarker) an input string x over the input alphabet A; you can assume here for simplicity that A contains only one character, eg. $A=\{1\}$, i.e. $x=1^n$ is a string of n 1's for some integer $n\geq 0$. The work tape is initially blank after the left endmarker. The requirement is that the TM must enter at the end a special halting state h with the work tape containing the number n in binary.
This can be accomplished by a TM that operates as follows: First initialize the work tape to 0, and then repeatedly move the input head one cell to the right and increment the number on the work tape, until the input head reaches a blank symbol. The work tape head should be at the right end between iterations and should travel in each iteration only as far left as it needs to perform the increment.

a. Give a complete detailed description of the TM, i.e. specify its set of states, the tape alphabet and the transition function, eg. via a transition table. (This will be the only problem where you are asked to give a detailed TM definition.) You do not have to prove the correctness of your construction; just include a brief explanation of what the states do.

b. Show that the time complexity of the TM is $O(n)$.
(Hint: Although some increments 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, 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. 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 $1,\ldots,l$, 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.

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? (Just give a rough estimate as a function of the time and space complexity of the multihead TM.)

Problem 3. 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 numbers, from most significant to least significant bit (the strings are allowed to have leading zeroes); 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 2 in parts b and c if it helps you, or you can give direct constructions.)

Problem 4. We saw in class that the language of palindromes is in TIME(n) and also in SPACE(log$n$). In this problem you will show that these time and space bounds cannot be achieved simultaneously. You will show in fact that if a Turing machine M decides the language of palindromes in time *T(n)* and space *S(n)* then $T(n) \cdot S(n) = \Omega(n^2)$.

Consider any input-output Turing machine M that decides a language L(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).

For any input *w*, define the *crossing sequence* at input cell *i,* denoted σ(M,*w*,*i*), to be the sequence of working configurations at the times when the input head crosses the boundary between cell *i* and *i*+1; note that the input head may cross several times this boundary, the first time going from left to right, then from right to left, and so forth.

a. Show that for any i-o Turing machine M and any strings u_1, v_1, u_2, v_2, if $u_1 v_1 \in L(M)$, $u_2 v_2 \in L(M)$, and $\sigma(M, u_1 v_1, |u_1|) = \sigma(M, u_2 v_2, |u_2|)$ then also $u_1 v_2 \in L(M)$ and $u_2 v_1 \in L(M)$.

Suppose now that M is a i-o Turing machine with time complexity T(*n*) and space complexity S(*n*) that decides the set of all palindromes over alphabet {0,1}.

b. Consider inputs of the form $w=x0^m y$, where $|x|=|y|=m=n/3$. Define the *reduced crossing sequence* σ'(M, $x0^m y$, $|x|$) to be the subsequence of σ(M, $x0^m y$, $|x|$) that consists only of the times in which the input head travels to the *y* portion of the input between any two consecutive elements of the subsequence.
Show that every reduced crossing sequence σ'(M, $x0^m y$, $|x|$) has length at most 3T(*n*)/*n*. Give an upper bound on the number of possible distinct reduced crossing sequences of this length in terms of T(*n*), S(*n*) and the constants of the Turing machine M (constant number of states, tapes, and tape symbols – you do not have to be too exact with the constants).

c. Show that any two different palindromes $x_1 0^m x_1^R$ and $x_2 0^m x_2^R$, where $x_1 \neq x_2$, have different reduced crossing sequences $\sigma'(M, x_1 0^m x_1^R, |x_1|) \neq \sigma'(M, x_2 0^m x_2^R, |x_2|)$.

d. Combine parts b and c to show that $T(n) \cdot S(n) = \Omega(n^2)$.