CSE 431 Spring 2007
Assignment #5

Due: Friday, May 11, 2006

Reading assignment: Read Sections 7.1-7.4 of Sipser's text.

Problems:

1. Sipser's text: Problem 7.6 (both editions).
2. Sipser's text: 1st edition Problem 7.10; 2nd Edition Problem 7.9.
3. Sipser's text: Problem 7.7 (both editions).
4. Sipser's text: Problem 7.11 (both editions).
5. All the computational problems we have described are defined as languages, i.e. yes/no questions. This problem gives an idea as to why that gives us enough information.
   Given a function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ we say that f is *computable in polynomial time* iff there is some TM computing f whose running time is $O(n^k)$ for some k. We say that f is *length-preserving* if $|f(x)| = |x|$ for every input x. Define the language $L_f = \{\langle x, i\rangle \mid$ the i-th bit of $f(x)$ is $1\}$.
   (a) Show that if f is polynomial-time computable then $L_f \in P$.
   (b) Show that if f is length-preserving and $L_f \in P$ then f is polynomial-time computable. (This direction holds more generally but this case gives the basic idea of the argument.)
6. (Bonus*) In this question you will show that if an ordinary 1-tape TM M has running time $o(n \log n)$ then $L(M)$ must be regular.

   A *crossing-sequence* is the sequence of states on which, and directions from which, a boundary between two cells is crossed during the course of a computation.

   (a) Show that if the lengths of all the crossing sequences for a TM are bounded by some constant k (independent of the input length) then $L(M)$ is regular. Do this by building an NFA N to recognize $L(M)$.
   (b) Use a pigeonhole argument to argue that for any TM running in $o(n \log n)$ time on any sufficiently long input, there must exist two different cell boundaries for cells that originally contained the input that have precisely the same crossing sequence in the computation on that input.
   (c) Show that if a 1-tape TM M has crossing sequences of arbitrarily large size then it cannot run in $o(n \log n)$ time. To do this, consider a minimal-length string that produces a long crossing sequence when M is run on it and use part (b) to derive a contradiction by splicing out a piece of the input string using the repeated crossing sequence.
   (d) Finally, put the pieces together to produce the claimed result.