CSE 531 Winter 2016
Computational Complexity I
Homework #2

Due: Monday, February 8, 2016

Problems:

1. Show that TQBF restricted to formulas where the part following the quantifiers is in conjunctive normal form is still PSPACE-complete.

2. Let $STRONGCONN$ denote the problem of deciding whether a directed graph has a path from u to v and v to u for every pair of vertices u, v. Show that $STRONGCONN$ is NL-complete.

3. Let $s_0, s_1, s_2, \ldots$ be an enumeration of the binary strings in lexicographic order, $0, 1, 00, 01, 10, 11, 000, 001, \ldots$. Define $L \subset \{0, 1, \#\}^*$ by
$$L = \{s_0\#s_1\#\cdots\#s_k \mid k \geq 0\}.$$
Show that $L \in DSPACE(\log\log n)$.

4. Show that if a Turing Machine uses $o(\log\log n)$ space, then it must use $O(1)$ space.
HINTS: Consider the shortest input $x_1 \cdots x_n$ that requires space $S > 0$ (where S is chosen to be large enough). Let $\mathcal{C}_i$ denote the set of all *partial* configurations that are possible when the input head is over location i, where a partial configuration consists of everything in the configuation except for the location of the input head. Then prove

   Lemma For $i < j \leq n$, $\mathcal{C}_i \neq \mathcal{C}_j$.

   To do this, assume that it is not the case, and consider the run of the machine on input $x_1 \cdots x_i x_{j+1} \cdots x_n$, and show that this run also uses space S, which contradicts the choice of $x_1 \cdots x_n$.

   Finally, count the number of possible sets $\mathcal{C}_i$, and use the pigeonhole principle to argue that if S is much smaller than $\log\log n$, then some $i < j$ must give the same sets $\mathcal{C}_i = \mathcal{C}_j$ and the same value $x_i = x_j$, which is a contradiction.

5. (Extra credit) A language L is called *unary* iff $L \subseteq 1^*$; in particular it includes at most one string of each length. Prove that if any unary language is NP-complete then $\mathsf{P} = \mathsf{NP}$.
HINT: Use the fact that there are only a polynomial number of different values required to specify the portion of the unary language that any mapping reduction from SAT can map to. Derive a polynomial-time SAT algorithm using the mapping reduction on the formulas appearing in the search-to-decision reduction for SAT.