# Problem Set 4 (due Thursday, March 12)

**1. (10 points) Determining whether a graph is a forest**

Show that it is possible to determine in logspace whether a given undirected graph is a forest (i.e., has no cycles).

(*Hint:* Note that the graph may not be connected. Logarithmic space means that you can only keep track of a constant number of vertices and edges. Start from a vertex $u$ on an edge $(u, v)$, and explore a path, by the following simple approach: when arriving at a node $w$, use the edge that is next on the adjacency list. If you loop back to $u$ along a different edge, then you have found a cycle. Repeat this process with other vertices and edges. Argue that this process always detects a cycle, if and only if one exists, and terminates while using space logarithmic in the size of the graph.)

**2. (10 points) Integer Multiplication is in L**

Problem 8.20 from Sipser's text.

**3. (10 + 10 = 20 points) Regular expressions and space complexity**

The following two parts concern the space complexity of the emptiness or universality of regular languages given in a certain form.

**(a)** Show that it is NL-complete to determine whether the language generated by a given DFA is empty.

(*Hint:* Compare with the directed reachability problem.)

**(b)** You are given a regular expression $r$ over the alphabet $\Sigma$ with the usual operators ($\cdot$, $+$, and $^*$). Show that it is PSPACE-complete to determine whether the language generated by $r$ is $\Sigma^*$.

(*Hint:* You need to show both membership in PSPACE, and PSPACE-hardness. For membership in PSPACE, note that NPSPACE = co-NPSPACE = PSPACE; take advantage of nondeterminism and the fact that deciding the complement of the language is "equivalent" to deciding the language. For PSPACE-hardness, one could try a reduction from a known PSPACE-hard problem such as QBF. Easier is to give a polynomial-time reduction from an arbitrary PSPACE language $L$: given a PSPACE machine $M$ deciding $L$ and a string $w$, generate a regular expression $r$ in polynomial time so that $M$ accepts $w$ if and only if $r$ generates all strings other than the string representing the accepting computation history of $M$ on $w$.)

**4. (10 + 2 + 8 = 20 points) P, NL, and PolyLog**

Define the class PolyLog as $\bigcup_{k \geq 0} \mathrm{SPACE}(\log^k n)$; i.e., the class of languages that require (deterministic) poly-logarithmic space.

**(a)** Prove that PolyLog does not have any language that is complete with respect to logspace reductions. That is, show that there does not exist any language $A \in$ PolyLog such that every language $B$ in PolyLog reduces to $A$ in logarithmic space.

**(b)** Using (a), argue that PolyLog $\neq$ P.

Two natural classes to consider at the "intersection" of PolyLog and P are (i) PolyLog $\cap$ P and (ii) PolyLogSpacePolyTime, the class of languages decided by deterministic TMs that run in *both* polylogarithmic space and polynomial time. (Note the subtle difference between the two definitions.)

**(c)** Show that NL is contained in PolyLog$\cap$P. However, we do not know whether NL is contained in PolyLogSpacePolyTime. Why does this not follow from Savitch's Theorem? In particular, what is the running time of the algorithm for reachability in directed graphs given in Savitch's Theorem?

**5. (10 points) Go-Moku is in PSPACE**

Problem 8.10 of Sipser's text.