College of Computer & Information Science — Spring 2009
Northeastern University — Handout 19
CSG714: Theory of Computation — 6 April 2009

Problems of the Week 12-15

12. Random walk on directed graphs

We saw that a random walk from a source node s in a given connected directed graph reaches any destination t within a polynomial number of rounds with high probability. This helped us to show that undirected connectivity is in RL. Show that the same approach will fail for directed graphs. That is, give an example of a directed graph G, in which the expected number of steps needed for a random walk to go from a given source s to a given destination t is exponential in the number of nodes, even though there is a path from s to t in G.

13. NL-completeness

Show that it is NL-complete to determine whether a given NFA accepts infinitely many strings.

(*Hint:* For NL-hardness, give a logspace reduction from the reachability problem in directed graphs.)

14. Linear inequalities for circuits

A few weeks ago, we showed that finding a value of a circuit is complete for the class P with respect to log-space reductions. Show how to reduce a boolean circuit to a set S of linear inequalities with all coefficients chosen from $\{-1, 0, 1\}$ such that the circuit evaluates to true if and only if the set S is feasible. That is, each linear inequality should be of the form $\sum_i a_i x_i \leq b$, where x_is form the variables and each a_i and b are in $\{-1, 0, 1\}$. Show that your reduction can be performed in log-space.

15. SATand unary languages

Show that if SAT reduces in polynomial-time to a unary language L, then P = NP.

(*Hint:* Consider the reduction, say f, from SATto L. This is a polynomial-time reduction; so for a given formula ϕ of length n, $f(\phi)$ is a string 1^ℓ, where ℓ is at most some polynomial $p(n)$. And ϕ is satisfiable if and only if $f(\phi)$ is in L. Now, consider the formula ϕ with variable x_1 set to 0 (and x_1 set to 1). These two new formulae also map to two unary strings of length no more than $p(n)$, according to f. Keep constructing these formulae and mapping them using f; if two formulae map to the same string, then we need not continue constructing the subformulae for both formulae. Since there are only a polynomial number of strings to map to, argue that this process will terminate in polynomial time.)