COMS 4236: Introduction to Computational Complexity, Spring 2012

Problem Set 6, due Monday April 30, in class

Problem 1. Do Problem 17.3.12 in the book. The integer B is given in unary.

Problem 2. Consider the following *DFA Intersection Problem*. We are given a collection of deterministic finite automata (DFA) $A_1, \cdots, A_n$, with the same alphabet Σ. The question is whether there exists a string that is accepted by all the DFA.

1. Show that the DFA Intersection problem is in PSPACE.
(*Hint:* Remember that NPSPACE = PSPACE.)

2. Show that the DFA Intersection problem is PSPACE-complete.
(*Hint:* One way is to use a reduction from the LBA problem, similar to the Reachable Deadlock problem. Use for the common alphabet Σ of the DFA the set of all tuples (q,a,i) consisting of a state q of the LBA, a tape symbol a and an index i=1,…,n of a tape cell. Given an input x to the LBA, construct your DFAs, one for each tape cell, so that they keep track of the symbol in the cell, and additionally the state if the tape head is in that cell. The DFAs should have the property that a string $w=w_1w_2\ldots w_m$ is accepted by all the DFAs iff the t-th symbol w_t of w for each t=1,2,…,m, consists of the state of the LBA, the tape symbol under the head and the location of the head of the LBA at time t in the computation of the LBA on input x, and the state in the last symbol w_m is accepting.)

Problem 3. Recall that a *matching* in a graph is a set of disjoint edges, i.e. a set of edges that do not share any nodes. (As a special case, the empty set of edges is considered a matching, as well as any set that contains only one edge).
The *#Matchings* problem is the following: Given an undirected graph G, output the total number of matchings in G (all matchings, not only perfect). The *#Matchings* problem is #P-complete, even when restricted to bipartite graphs. (You do not have to show this.)

Prove that the following problems are #P-complete.
1. #2SAT for monotone formulas: Given a Boolean formula ϕ in conjunctive normal form, where each clause contains two variables (no negative literals), compute the number of satisfying assignments of ϕ.
(*Hint*: Reduce from the #Matchings problem. Use a Boolean variable for each edge denoting that the edge is *not* in the matching.)

2. #Node Covers: Given an undirected graph G, output the number of node covers of G.