CS 142 & Math 167
Fall 2008
Homework 5
Due Wednesday, October 8

1. **[15 Points] PCP!** In class we stated the PCP Theorem without proof. That is, we asserted that $PCP(\log n, 1) = NP$. Prove that $PCP(\log n, 1) \subseteq NP$. (The other direction, $NP \subseteq PCP(\log n, 1)$, is the basis of Sanjeev Arora's Ph.D. dissertation at Berkeley which won the ACM Best Doctoral Dissertation Award for 1995.)

2. **[15 Points] Clique Revisited!** On the last homework assignment we showed that CLIQUE is NP-complete via a reduction from 3SAT. Now we'll show that this reduction has a "gap" property and then show that this implies that CLIQUE does not have a PTAS unless $P = NP$. *This problem isn't hard, it just requires modest assembly of ideas that we've grown to know and love.*

   Let ϕ denote an expression in 3CNF form (that is, the form in which 3SAT and MAX3SAT take their input). Let $\sigma(\phi)$ denote the fraction of the clauses of ϕ which are satisfiable in the "best" valuation. So, the expression is satisfiable iff $\sigma(\phi) = 1$. Similarly, let G be a graph and let $w(G)$ denote the size of the largest clique in G.

   (a) Prove that there is a polynomial-time reduction from MAX3SAT to CLIQUE such that for all instances ϕ of MAX3SAT with C clauses, the instance G of CLIQUE which is created has the property that

   $$\sigma(\phi) = k \Leftrightarrow w(G) = kC$$

   (b) Now explain why CLIQUE has no PTAS if $P \neq NP$.

   (c) Finally, explain why CLIQUE has no constant factor approximation algorithm unless $P = NP$.

3. **[15 Points] Funkiness!** Let

   $L_{\text{funky}} = \{< M, w > | \text{TM } M \text{ accepts } w \text{ without ever leaving the first } |w| + 1 \text{ tape cells}\}$

   Prove that L_{funky} is PSPACE-complete. *Careful here. Showing that the problem is in PSPACE requires a bit of caution.*

4. **[20 Points] A New Complexity Class!**

   In this problem you will investigate a "new" complexity class called DP. DP is the collection of all languages L such that L can be expressed as $A - B$ where A and B are both in NP. Recall that $A - B$ means the collection of items in A and not in B.

   (a) First, show that DP contains all of NP and also all of co-NP. Aha, so DP includes some pretty hard problems!

   (b) A decision problem L is defined to be *DP-complete* if L is in DP and also every problem in DP is polynomial-time reducible to L. Now, consider the language

   $$L = \{< G_1, k_1, G_2, k_2 > \mid \ G_1 \text{ has a clique of size } k_1 \text{ and } G_2 \text{ doesn't have a clique of size } k_2\}$$

   Prove that L is DP-complete.

   (c) Next, consider the problem MAX-CLIQUE defined as the language:

   $$\{< G, k > \mid \text{the largest clique in } G \text{ is of size exactly } k\}$$

   Prove that MAX-CLIQUE is DP-complete.

5. **[35 Points] Regular expressions and PSPACE.**

   Many problems related to regular expressions are known to be in PSPACE but are not known to be in any "lower" complexity class. For example, consider the problem of determining whether a given regular expression over the alphabet $\Sigma = \{0, 1\}$ is equivalent to Σ^*. For example, $0^*1^* + (0 + 1)1^*$ is not the same same as Σ^* but $1^*((\epsilon + 1)(\epsilon + 0))^*$ is equivalent to Σ^*.

   So, we are considering the following problem: Given an encoding of a regular expression, is that regular expression equivalent to Σ^*? We'll call this problem REGEQ (REGular expression EQuivalence). We'll show that REGEQ is in PSPACE in two steps.

   (a) First, consider the problem of determining whether a DFA accepts Σ^*. Describe a finite-time algorithm (it need not be efficient in any sense) that takes as input a DFA and determines whether that DFA accepts Σ^*. Explain why your algorithm is correct.

(b) Now, prove that REGEQ is in PSPACE. Be careful to show that your solution works correctly and really uses only polynomial space! (*Note:* Take a close look at how we prove the equivalence of regular expressions, NFA's, and DFA's. You will also want to use the result that you proved in part (a), but you will need to use it with caution.)

(c) Finally, consider the more general problem REGPAIR which takes as input the encodings of two regular expressions, R_1 and R_2, and determines whether or not these two regular expressions represent the same language. Prove that REGPAIR is in PSPACE.