COS487 Theory of Computation	**Fall 2006**

Assignment #7

Due: Thursday November 30 — *Sean Hallgren*

This HomeWork is due Thursday, Nov 30 in class. So you have two weeks to complete this task.

1. A *permutation* on the set $\{1, \ldots, k\}$ is a one-to-one, onto function on this set. When p is a permutation, p^t means the composition of p with itself t times. Let

   $PERM\text{-}POWER = \{\langle p, q, t\rangle \mid p = q^t$ where p and q are permutations on $\{1, \ldots, k\}$ and t is a binary integer$\}$

   Show that $PERM\text{-}POWER \in \mathbf{P}$. (Note that the most obvious algorithm doesn't run within polynomial time. Hint: First try it where t is a power of 2).

2. Show that $\mathbf{P}$ is closed under the star operation. (Hint: Use dynamic programming. On input $y = y_l \cdots y_n$ for $y_i \in \Sigma$, build a table indicating for each $i < j$ whether the substring $y_i \cdots y_j \in A^*$ for any $A \in \mathbf{P}$.)

3. Let *UNARY-SSUM* be the subset sum problem in which all numbers are represented in unary. Why does the **NP**-completeness proof for *SUBSET-SUM* fail to show *UNARY-SSUM* is **NP**-complete? Show that *UNARY-SSUM* $\in \mathbf{P}$.

4. Show that, if $\mathbf{P} = \mathbf{NP}$, then every language $A \in \mathbf{P}$, except $A = \varnothing$ and $A = \Sigma^*$, is **NP**-complete.

5. Let ϕ be a 3cnf-formula. An $\neq$-*assignment* to the variables of ϕ is one where each clause contains two literals with unequal truth values. In other words, an $\neq$-assignment satisfies ϕ without assigning three true literals in any clause.

   (a) Show that the negation of any $\neq$-assignment to ϕ is also an $\neq$-assignment.

   (b) Let $\neq$SAT be the collection of 3cnf-formulas that have an $\neq$-assignment. Show that we obtain a polynomial time reduction from 3SAT to $\neq$SAT by replacing each clause c_i

   $$(y_1 \vee y_2 \vee y_3)$$

   with the two clauses

   $$(y_1 \vee y_2 \vee z_i) \text{ and } (\bar{z}_i \vee y_3 \vee b)$$

   where z_i is a new variable for each clause c_i and b is a single additional new variable.

   (c) Conclude that $\neq$SAT is **NP**-complete.

6. A *cut* in an undirected graph is a separation of the vertices V into two disjoint subsets S and T. The size of a cut is the number of edges that have one endpoint in S and the other in T. Let

   $$MAX\text{-}CUT = \{\langle G, k\rangle \mid G \text{ has a cut of size } k \text{ or more}\}$$

Show that *MAX-CUT* is **NP**-complete. You may assume the result of Problem 7.24. (Hint: Show that $\neq SAT \leq_p$ *MAX-CUT*. The variable gadget for variable x is a collection of $3c$ nodes labeled with x and another $3c$ nodes labeled with $\bar{x}$, where c is the number of clauses. All nodes labeled x are connected with all nodes labeled x . The clause gadget is a triangle of three edges connecting three nodes labeled with the literals appearing in the clause. Do not use the same node in more than one clause gadget. Prove that this reduction works.)

7. Show that if $\mathbf{P} = \mathbf{NP}$, a polynomial time algorithm exists that produces a satisfying assignment when given a satisfiable Boolean formula. (Note: The algorithm you are asked to provide computes a function, but **NP** contains languages, not functions. The $\mathbf{P} = \mathbf{NP}$ assumption implies that *SAT* is in **P**, so testing satisfiability is solvable in polynomial time. But the assumption doesn't say how this test is done, and the test may not reveal satisfying assignments. You must show that you can find them anyway. Hint: Use the satisfiability tester repeatedly to find the assignment bit-by-bit.)

8. A *2cnf-formula* is an AND of clauses, where each clause is an OR of at most two literals. Let *2SAT* $= \{\langle\phi\rangle \mid \phi$ is a satisfiable 2cnf-formula$\}$. Show that *2SAT* $\in$ **P**.

9. (Optional)The *difference hierarchy* $\mathbf{D_iP}$ is defined recursively as

   (a) $\mathbf{D_1P} = \mathbf{NP}$ and

   (b) $\mathbf{D_iP} = \{A \mid A = B \setminus C$ for $B \in \mathbf{NP}$ and $C \in \mathbf{D_{i-1}P}\}$
   (Here $B \setminus C = B \cap \bar{C}$.)
   For example, a language in $\mathbf{D_2P}$ is the difference of two **NP** languages. Sometimes $\mathbf{D_2P}$ is called **DP** (and may be written $\mathbf{D^P}$). Let

   $$Z = \{\langle G_1, k_1, G_2, k_2\rangle \mid G_1 \text{ has a } k_1\text{-clique and } G_2 \text{ doesn't have a } k_2\text{-clique}\}$$

   Show that Z is complete for **DP**. In other words, show that every language in **DP** is polynomial time reducible to Z.

10. (Optional)Call a regular expression *star-free* if it does not contain any star operations. Let

$$EQ_{SF-REX} = \{\langle R, S\rangle \mid R \text{ and } S \text{ are equivalent star-free regular expressions}\}$$

Show that EQ_{SF-REX} is in **coNP**. Why does your argument fail for general regular expressions?