cs579 Computational Complexity Out: *Thu., 2024-08-29*

Problem Set #1

Prof. Michael A. Forbes Due: *Thu., 2024-09-12 12:00*

All problems are of equal value.
Note: there are hints on the last page, for those who want them.

1. (Sipser #7.12) Let $\mathsf{MODEXP} = \{(a,b,c,p)$: a, b, c, and p are integers written in binary, such that $a^b \equiv c \mod p\}$. Show that $\mathsf{MODEXP} \in \mathsf{P}$.

2. (Sipser #7.16)

   Let $\mathsf{UNARYSSUM}$ be the subset sum problem in which all numbers are represented in unary.

   (a) Why does the NP-completeness proof for $\mathsf{SUBSETSUM}$ fail to show $\mathsf{UNARYSSUM}$ is NP-complete?

   (b) Show that $\mathsf{UNARYSSUM} \in \mathsf{P}$.

3. (Sipser #7.24) 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 $\neq \mathsf{SAT}$ be the collection of 3CNF-formulas that have an $\neq$-assignment. Show that we obtain a polynomial time reduction from $\mathsf{3SAT}$ to $\neq \mathsf{SAT}$ by replacing each clause $c_i = (y_1 \vee y_2 \vee y_3)$ with the two clauses $(y_1 \vee y_2 \vee z_i)$ and $(\overline{z_i} \vee y_3 \vee b)$, where z_i is a new variable for each clause c_i and b is a single additional new variable.

   (c) Conclude that $\neq \mathsf{SAT}$ is NP-complete.

4. (Sipser #7.25) 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 $\mathsf{MAXCUT} = \{\langle G, k\rangle : G$ has a cut of size k or more$\}$. Show that MAXCUT is NP-complete.

5. (Arora-Barak #1.15) Define a Turing Machine (TM) M to be *oblivious* if its head movements do not depend on the input but only on the input length. That is, M is oblivious if for every input $x \in \{0,1\}^\star$ and $i \in \mathbb{N}$, the location of M's head at the i-th step of execution on input x is only a function of $|x|$ and i.

   A function $t : \mathbb{N} \to \mathbb{N}$ is *(weakly) time-constructible* if the function that maps the string 1^n to the string $1^{t(n)}$ is computable in time $O(t(n))$.

   Show that for every weakly time-constructible function $t : \mathbb{N} \to \mathbb{N}$, if $L \in \mathsf{TIME}(t(n))$ (on a one-tape TM), then there is an oblivious *two*-tape TM that decides L in time $O(t(n)^2)$.

6. (Sipser #7.36) Show that if $\mathsf{P} = \mathsf{NP}$, then a polynomial time algorithm exists that produces a satisfying assignment when given a satisfiable boolean formula.

Some hints.

1. Note that the most obvious algorithm doesn't run in polynomial time. Try it first where b is a power of 2.

4. Show that $\neq \mathsf{SAT} \leq_p \mathsf{MAXCUT}$. The variable gadget for variable x is a collection of $3c$ notes labeled with x and another $3c$ nodes labeled with $\overline{x}$, where c is the number of clauses. All nodes labeled x are connected with all nodes labeled $\overline{x}$. 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.

5. One can produce an oblivious *one-tape* TM with the same time complexity. We only ask for a two-tape TM to make the problem slightly easier.

6. The algorithm you are asked to provide computes a function, but NP contains languages, not functions. The $\mathsf{P} = \mathsf{NP}$ 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. Use the satisfiability tester repeatedly to find the assignment bit-by-bit.