cs579: Computational Complexity Assigned: *Thu., Jan. 17, 2019*

Problem Set #1

Prof. Michael A. Forbes Due: *Thu., Jan. 31, 2019 (3:30pm)*

Some details from the syllabus.

- **Submission Policy:** Problem sets will be posted by 3:30pm on the day of release, and are due 3:30pm on the due date. An electronic (pdf) copy must be submitted by email to the course staff (`miforbes` and `awk2`). Each problem should be on a separate page. The subject line of the email should be "[cs579] ps$N$ submission" (N = pset number) and the filename must be "*NETID*_ps$N$.pdf" (N = pset number, *NETID* = your netid).
- **Collaboration Policy:** Students are forbidden from directly searching for solutions on the internet, but may consult the exercise-hints in the textbooks for the course. That said, students are highly encouraged to collaborate in small groups. However, this must be a two-way collaboration. Students should not dictate complete solutions to other students, either verbally or written. Solutions must be written independently regardless of collaboration, and psets must list the collaborators you worked with.
- **Late Policy:** Students are highly encouraged to turn-in assignments on-time to avoid falling behind on the material, and to incentivize this any late homework will automatically lose 10%. However, to be flexible, students have a total of 6 late days (24 hours each, rounded up to an integral number of days), no more than 3 of which can be used for any particular homework. Problem sets will not be accepted for credit once the late days are exhausted.
- **Solutions:** Hard-copy sample solutions will be distributed to students when the problem sets are returned (please keep the internet free of easily-found solutions). These sample solutions will be selected from student submissions (with names omitted). Please inform the course staff if you wish to *opt-out* of ever being selected.

Each problem is worth 10 points.

1. (Sipser #7.12) Let $\mathsf{MODEXP} = \{(a, b, c, p) : a, b, c, \text{ and } p \text{ are integers written in binary, such that } a^b \equiv c \mod p\}$. Show that $\mathsf{MODEXP} \in \mathsf{P}$.
2. (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 $3\mathsf{SAT}$ 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.

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

4. (Sipser #7.32) Let $U = \{\langle M, x, \#^t\rangle : \text{non-deterministic TM } M \text{ accepts input } x \text{ within } t \text{ steps on at least one branch}\}$. Show that U 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.

3. 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.

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.