**Design and Analysis of Algorithms** **out:** March 26, 2015

# Problem Set 6

*Lecturer: Daniel A. Spielman* **due: 2:30** PM, April 2, 2015

## Instructions. Read Carefully

1. Make sure that your yale **netid** appears on **every** page of your problem set, and that it is written clearly.

2. The problem sets are due at the beginning of class. Solution sets will be provided in class. For this reason, late problem sets are not accepted.

3. You will **hand in each problem separately**, so make sure that they are not stapled together and that no problem appears on the back of another. There will be 4 boxes in the front of the class, one for each problem. Put each problem in the correct box.

4. Alternatively, you may submit the problem set electronically via the Classes server. If you do so, submit it as a **pdf** and make sure that your **netid** appears on **every** page. **Each problem must be in its own document.**

5. Cite any resources that you use, other than the textbook, TA and instructor.

6. You **must not** search the web for solutions to similar problems given out in other classes.

## Collaboration Policy

You must write the solutions to the problems independently and in your own words.

However, you are allowed to discuss the problem sets in small groups of no more than 4 students. To ensure that you understand the solutions you submit, you are forbidden from taking written notes during your discussions. You must list at the top of the problem set everyone (other than course staff) with whom you have discussed the problem set. Failure to list people with whom you have discussed a problem set is considered a violation of academic honesty.

# Problem 1: Fuzzy Sat

In this problem I will introduce a variant of the SAT problem that I will call FuzzySAT. Your job is to prove that FuzzySAT is NP-Complete. Please remember that this involves both proving that it is in NP and that it is NP-Hard.

Like SAT, the input to FuzzySAT is a list of clauses. However, the variables in FuzzySAT are able to take one of three values, true (1), false (0), or fuzzy (?). The negation of fuzzy is fuzzy. A clause that contains a term that is fuzzy evaluates to fuzzy. An instance of FuzzySAT is satisfiable if there is a setting of the variables in which at most half of the variables are fuzzy that makes every clause true or fuzzy. The answer to the FuzzySAT problem is "yes" if its input instance is satisfiable.

Note: FuzzySAT would be easy if you were allowed to make all of the variables fuzzy. I have made it a difficult problem by only allowing at most half of the variables to be fuzzy.

This problem was inspired by the fuzzy people I've met who have told me that computers will never be able to think because they reduce everything to 0s and 1s. I don't know if computers will ever be able to think, but that's not why.

# Problem 2: Majorities

A majority gate is a logical gate like an OR or an AND gate. It can have any number of inputs, and is true if a *strict* majority of its inputs are true. For example $\mathsf{maj}(0,1,1,0) = 0$ but $\mathsf{maj}(1,0,1) = 1$. They were inspired by neurons, which fire if enough of their inputs fire. We do allow majority gates to have repeated terms in their inputs. For example, $\mathsf{maj}(x_1, x_1, x_1, x_2, x_2, \overline{x_3}, x_4)$ is completely reasonable.

An instance of the Maj-SAT problem consists of boolean variables, $x_1, \ldots, x_n$ and a list of majority gates, each of whose inputs is a variable or its negation. In this way, the instance looks just like a SAT instance (each clause is a list of terms), but the meaning is different. The instance is said to be satisfiable if there is a setting of the variables that makes a strict majority of the majority gates true. You can think of the instance as being a circuit in which there is one big majority gate that has an input all the majorities in the list. In the Maj-SAT problem, you are given a Maj-SAT instance as input and must return "yes" if the instance is satisfiable. Prove that Maj-SAT is NP-complete.

# Problem 3: Finding the Solution

Imagine someone gives you a magic computer that can solve SAT. Maybe its a "quantum" computer. Maybe its an enchanted toaster. You just know that when it is given a list of clauses, it tells you whether or not they have a satisfying assignment. Can you use this device to actually find a satisfying assignment?

Formally, suppose that you have access to a black box $M$ that on input any list of clauses, $C_1, \ldots, C_k$, will tell you if there exists a truth assignment that satisfies all the clauses. Design a polynomial-time algorithm that uses $M$ to find satisfying assignments. Your algorithm will take as input a list of clauses. If the list of clauses is satisfiable, your algorithm should return a satisfying assignment. Otherwise, your algorithm should return "not satisfiable". Your algorithm should perform a polynomial number of ordinary computations, and make at most polynomial number of calls to the black box $M$.

# Problem 4: Sensor Placement

Determining the optimal placement for sensors is a big problem. We will study a special case of this problem in graphs. The input is an undirected graph $G = (V, E)$. The problem is to choose a subset of the vertices $S$ that can monitor every other vertex. A vertex $a$ can monitor itself and every other vertex $b$ for which $(a, b) \in E$. So, a subset of the vertices $S$ can monitor all the vertices if for every $b \in V$,

$$b \in S \quad \text{or} \quad \text{there is an } a \in S \text{ so that } (a, b) \in E.$$

a. In the problem *Restricted Sensor Placement Problem* (RSP), you are given as input a graph $G = (V, E)$, a set $W \subseteq V$ and an integer $k$. The answer is "yes" if and only if there is an $S \subseteq W$ of size at most $k$ that can monitor all of $V$. This problem is called "Restricted" because the set $S$ is restricted to lie inside $W$.

   Prove that RSP is NP-Complete.

   *Hint:* You should first show that it is in NP. You should then show that if you can solve this problem, then you can solve some other NP-complete problem.

b. In the general *Sensor Placement Problem*, you are given a graph $G = (V, E)$ as input and an integer $k$. The answer to the problem is "yes" if $V$ contains a set $S$ of size at most $k$ that can monitor every vertex. Prove that the *Sensor Placement Problem* is NP-Complete.

   *Hint:* You can reduce *Restricted Sensor Placement* to *Sensor Placement*.