

**15-251 : Great Theoretical Ideas In Computer Science****Fall 2014****Assignment 11**Due: **Thursday**, December 4, 2014 11:59 PM

Name: \_\_\_\_\_

Andrew ID: \_\_\_\_\_

Question:	1	2	3	4	Total
Points:	25	25	25	25	100
Score:					

## 0. Warmup

- (a) Recall that for a language  $L$  to be in NP, we must have a polynomial time “verifier” that, when given a “witness” (proof), can verify that a given  $x \in L$ . In the case of *CIRCUIT\_SAT*, the proof can simply be a setting of the variables  $x_1 \cdots x_n$  such that the circuit evaluates to True, and the verifier can be an algorithm that plugs these values into the circuit and computes the output.

What proof and verifier would you provide for the the following languages in NP?

*3\_SAT*?

*3\_COLOR*?

*CLIQUE*?

*IS\_COMPOSITE*? Checks if an integer  $n$  is composite. (This is also in  $P$ .)

*TRAVELING\_SALESMAN*? Answers whether a directed graph with weighted edges has a path of cost at most  $k$  which visits every vertex exactly once.

- (b) Let  $A$  be an NP-Complete problem.

Let  $B$  be a problem that is poly-time reducible to  $A$ . That is, we can use an oracle for  $A$  to efficiently solve  $B$ . What can we conclude about  $B$ ?

Let  $C$  be a problem such that  $A$  is poly-time reducible to  $C$ . That is, we can use an oracle for  $C$  to efficiently solve  $A$ . What can we conclude about  $C$ ?

Can  $B$  and  $C$  be the same problem? Are they necessarily the same problem? Are they in  $NP$ ? Can they be in  $P$ ?

## 1. Getting all snacks

- (25) (a) After an exhausting day of grading, the 15-251 TAs visit the cafeteria to grab some food. The cafeteria serves  $m$  different kinds of snacks  $S = \{s_i\}_{i=1}^m$ . The snacks are grouped into  $n$  different types of bags  $B_1, \dots, B_n \subseteq S$  (the same kind of snack might be in multiple bags). That is, each type of bag  $B_i$  contains a non-empty subset of the snacks in  $S$ .

Each bag costs \$1. The 251 TAs want to try all  $m$  different kinds of snacks but Venkat and Victor only give them a budget of  $k$  dollars to spend. The problem is therefore to decide if there are  $k$  bags  $B_{i_1}, \dots, B_{i_k}$  such that  $\bigcup_{j=1}^k B_{i_j} = S$ .

Show that this problem is NP-Complete.

Hint: For this homework you may assume that the following problems, as defined in lecture, are NP-Complete: *CIRCUIT SAT*, *3SAT*, *3COLOR*, *CLIQUE*, *INDSET*.

## 2. Those roots can be hard to find

For a polynomial with integer coefficients in several variables, an integral root is an assignment of integers to the variables so that the polynomial evaluates to 0. For example, the polynomial  $2x_1^2x_2 - x_1^3x_2^2$  has an integral root  $x_1 = 2$  and  $x_2 = 1$ , where as the polynomial  $x_1^2 + x_2^2 + 1$  has no integral root. Consider the computational problem of determining, given a polynomial with integer coefficients, whether it has an integral root. We can capture this problem by the language:

$$ROOT = \{ \langle p \rangle \mid p \text{ is a polynomial in several variables, given as a list of its terms,} \\ \text{that has an integral root} \} .$$

In his famous address at the International Congress of Mathematicians in Paris in 1900, David Hilbert posed 23 mathematical problems as challenges for the 20th century, the 10'th problem of which, re-stated in the language of computability theory, was to give a decider for *ROOT*. We now know, however, that *ROOT* is in fact undecidable, so no algorithm exists to tell if a polynomial in several variables has an integral root.

- (25) (a) In this problem you are to prove a different property of *ROOT*, namely, that *ROOT* is NP-hard.

Hint: How can you enforce a potential integral root to only take values in  $\{0, 1\}$ ? One possible reduction is from *3SAT* (though, again, you may reduce from any of *CIRCUITSAT*, *3SAT*, *3COLOR*, *CLIQUE*, *INDSET*, or even some new intermediate problem that you prove to be NP-complete).

### 3. What's the smallest circuit?

- (25) (a) Let `SMALLESTCIRCUIT` be the language consisting of (suitable encodings in  $\{0, 1\}^*$ ) of all Boolean circuits  $C$  (with many input bits and a single output bit) with the property that there is no smaller circuit  $C'$  that has the same truth table as  $C$ . (Here “smaller” means fewer gates, and the truth table of a circuit with  $n$  inputs has  $2^n$  entries, one for each setting of the  $n$  input bits.)

Prove that if  $P = NP$ , then `SMALLESTCIRCUIT`  $\in P$ .

Hint: If a language is in  $P$ , then its complement is also in  $P$ . Also, think about how you would check whether two circuits, each with  $n$  inputs, compute the same truth table.

#### 4. Balanced gifting

It is the holiday season now, and you have a collection of  $n$  awesome gifts, the  $i$ 'th gift being worth  $v_i$  dollars, to be distributed among  $m$  of your family members (assume  $n > m$ , so you have enough gifts for everyone).

You want to give away all the gifts, each gift going to exactly one person (the gifts are indivisible). However, the value  $v_i$  of the gifts is public knowledge, so everyone will know the total value everyone else receives. So to minimize envy, you would like to allocate the gifts in a balanced way, so as to minimize the maximum value any person receives.

Formally, you would like to find a partition  $\mathcal{P}$  of the  $n$  gifts as  $\{1, 2, \dots, n\} = S_1 \cup S_2 \cup \dots \cup S_m$ , the  $S_j$ 's being disjoint sets with the  $j$ 'th person receiving the gifts in subset  $S_j$ , so that

$$V(\mathcal{P}) \stackrel{\text{def}}{=} \max_{j \in \{1, 2, \dots, m\}} \sum_{i \in S_j} v_i \quad (1)$$

is *minimized*.

Let  $\text{OPT}$  be the minimum value of  $V(\mathcal{P})$  over all possible partitions  $\mathcal{P}$ . Turns out finding the optimal partition  $\mathcal{P}$  with  $V(\mathcal{P}) = \text{OPT}$  is a hard problem. It is in fact one of those pesky NP-hard problems you know about. However, you also know something about how to cope with NP-hard problems by finding approximately optimal solutions.

- (25) (a) Can you show off your 15-251 skills this holiday season by designing an efficient algorithm, running in time polynomial in  $n$ , that finds a partition  $\mathcal{P}$  with  $V(\mathcal{P}) \leq 2 \cdot \text{OPT}$ ?

Hint: One approach is to use an appropriate “greedy” strategy. Also, think about how to lower bound  $\text{OPT}$  in terms of the  $v_i$ 's and  $m$ .