Exam 5

CS 3510A, Spring 2014

These solutions are being provided **for your personal use only**. They are not to be shared with, or used by, anyone outside this class (Spring 2014 section of Georgia Tech CS 3510A). Deviating from this policy will be considered a violation of the GT Honor Code. **Instructions:**

- **Do not open this exam packet until you are instructed to.**
- **Write your name** on the line below.
- You will have **50 minutes** (from 11:05 to 11:55) to earn **50 points**. Pace yourself!
- You may use one double-sided 8.5-by-11" study sheet. **You may not use any other resources, including electronic devices.**
- Do your best to fit your final answers into the space provided below each question. You may use the backs of the exam pages, and the final page of the exam, as scrap paper.
- **You may refer to any facts from lectures or the textbook without re-deriving them.**
- Your work will be graded on correctness and clarity. Please write legibly!

Question	Points	Score
1	21	
2	14	
3	16	
Total:	51	

Your name: _____

1. (21 points) *True, False, **or Unknown***. State whether each of the following sentences is known to be true, known to be false, or **not yet known** to be true or false, **and give a convincing justification for your answer**. A correct justification is worth more than a correct true/false answer.

   (a) To prove that a problem B is NP-complete, it is enough to prove that B is in NP and that B reduces to 3SAT.

   > **Solution:** False. The direction of the reduction is wrong—it only shows that $B \leq$ 3SAT—so it does not prove what we need. For example, if B was in P, then B would still reduce to 3SAT, but it would not be NP-complete (unless $P = NP$!).
   >
   > To show that B is NP-complete, it would be enough to reduce 3SAT to B, i.e., to prove that 3SAT $\leq B$.

   (b) For decision problems A, B, C, if $A \leq B$ and $B \leq C$, then $A \leq C$.
   (Here your justification should do more than just quote facts from the book/lectures.)

   > **Solution:** True. By assumption, there are reductions R_{ab}, R_{bc} such that: x is a YES instance of $A \iff y = R_{ab}(x)$ is a YES instance of B, and y is a YES instance of $B \iff z = R_{bc}(y)$ is a YES instance of C. Therefore, there is a reduction R_{ac} from A to C, which does the following: given an instance x of A, compute $y = R_{ab}(x)$, then compute $z = R_{bc}(y)$, and output z.
   >
   > The above reduction is clearly correct by the properties of R_{ab}, R_{bc} stated above. It also runs in time polynomial in $|x|$, because $R_{ab}(x)$ runs in time polynomial in $|x|$, and $R_{bc}(y)$ runs in time polynomial in $|y|$, which itself is polynomial in $|x|$.

   (c) There is a (polynomial time) reduction from 3SAT to 2SAT.

   > **Solution:** Unknown, because this statement is true if and only if $P = NP$. Because 2SAT is in P, a reduction demonstrating 3SAT $\leq$ 2SAT would imply that 3SAT is in P, and because 3SAT is NP-complete, this in turn would imply that $P = NP$. Conversely, if $P = NP$ then there are polynomial-time algorithms for both 2SAT and 3SAT, and therefore there are trivial polynomial-time reductions between them (in both directions).

2. The "longest path" *search* problem is defined as follows: given a weighted directed graph G, find a *simple* path with *largest* total weight. (A simple path never visits the same vertex more than once.)

   (a) (7 points) Define a *decision* problem that captures the essence of the longest-path problem. Clearly describe the form of the input and the yes/no question that the problem asks, and briefly give some intuition why it captures the essence of the search problem. (You do not need to give a detailed proof.)

   Solution: The decision problem is: given a weighted directed graph G and a number k, does there exist a simple path in G having total weight *at least* k? This captures the search problem because by answering the question for different values of k using binary search, we can learn the length k^* of a longest path. It is also possible to find a longest path by removing one edge at a time and asking whether the graph still has a path of length k^*; a good exercise is to show this formally.

   Note that $\geq k$ (not $\leq k$) is the proper test here, because we are concerned with *longest* paths. If we had used $\leq k$, then the yes/no answers for various values of k would not give us any useful information about a longest path.

   (b) (7 points) Prove that the decision problem you defined above is in NP. Be sure to clearly describe the form of a witness and the efficient verification algorithm.

   Solution: For a "yes" instance (G, k), a witness is an ordered list of the vertices $v_1, \ldots, v_\ell$ on some simple path of G that has total weight at least k. The verifier, given the instance (G, k) and a witness, checks that no vertex is repeated, that there is an edge from each v_i to v_{i+1}, and that the sum of the weights of these edges is at least k. It is easy to see that this verification procedure is polynomial time. Moreover, if (G, k) is a "no" instance (i.e., G has no simple path of weight $\geq k$), then no witness will convince the verifier.

3. The 4SAT problem is defined almost identically to the 3SAT problem; the only difference is that the number of literals in each clause is four instead of three.

   (a) (8 points) Let $(\ell_1 \vee \ell_2 \vee \ell_3)$ be an arbitrary 3-clause, where each ℓ_i is some literal (either a variable or its negation), and fix some arbitrary assignment to the variables. Prove that the assignment satisfies the 3-clause *if and only if* the *same* assignment satisfies both of the 4-clauses $(\ell_1 \vee \ell_2 \vee \ell_3 \vee \ell_4)$ and $(\ell_1 \vee \ell_2 \vee \ell_3 \vee \overline{\ell_4})$, where ℓ_4 is some arbitrary literal.

   Solution: If the assignment satisfies $(\ell_1 \vee \ell_2 \vee \ell_3)$, then at least one of those literals is true under the assignment. So both of the 4-clauses in question also have at least one true literal, which means they are both satisfied.

   In the other direction, suppose that the assignment satisfies both 4-clauses. Then because ℓ_4 and $\overline{\ell_4}$ are not both true, at least one of ℓ_1, ℓ_2, ℓ_3 must be true under our assignment (otherwise one of the 4-clauses would not be satisfied). Therefore, the original clause $(\ell_1 \vee \ell_2 \vee \ell_3)$ is satisfied.

   (b) (8 points) Using the previous part, prove that 4SAT is NP-complete. (To save time, you don't have to show that 4SAT is in NP.)

   Solution: Clearly 4SAT is in NP; a witness for a 4SAT formula is a satisfying assignment to all the variables, and the verifier just checks that all the clauses evaluate to "true" under the assignment.

   We now give a reduction from 3SAT to 4SAT, showing 3SAT $\leq$ 4SAT; since 3SAT is NP-complete this will show that 4SAT is NP-complete as well. Given an input 3SAT instance ϕ having m clauses, our reduction constructs a 4SAT formula ϕ' with $2m$ clauses on the same variables. For each 3-clause $(\ell_1 \vee \ell_2 \vee \ell_3)$ of ϕ, we just create two 4-clauses in ϕ' as in the previous part, letting the fourth literal ℓ_4 be an arbitrary one of the variables. This procedure clearly runs in polynomial time.

   To prove that this is a valid reduction, we need to show that ϕ is a "yes" instance of 3SAT (i.e., it has a satisfying assignment) if and only if $\phi' = R(\phi)$ is a "yes" instance of 4SAT. This follows immediately from the previous part, since any assignment that satisfies ϕ also satisfies ϕ', and vice versa.

Scrap paper — no exam questions here.