CSCI 570 - Fall 2016 - HW 10

Due: *Nov 18th*

1. State True/False. If $A \leq_p B$ and $B \in NP$, then $A \in NP$.

   True. The important observation is that you can construct a certifier for A by composing the polynomial time reduction map and the certifier for B. A proof sketch for Karp reductions follows if you are interested.

   Proof: $B \in NP$ implies there exists a polynomial time certifier C_B. That is, there exists a constant k such that, if x is a "no" instance for B then $\forall c, C_B(x, c) = 0$ and if $x \in B$ then there exists a c_x such that $||cx|| \leq ||x||^k$ and $C_B(x, cx) = 1$. Here, $||y||$ denotes the number of bits used to write y.

   Let f be a polynomial time reduction from A to B with running time bounded by a polynomial of degree d. That is, f is a polynomial time algorithm that maps instances of A to instances of B, such that x is a "yes" instance of A if and only if $f(x)$ is a "yes" instance of B. Compose the algorithms C_B and f to obtain the polynomial time algorithm C_A, that is

   $$\forall x, \forall c, C_A(x, c) := C_B(f(x), c)$$

   We claim that C_A is a polynomial time certifier for A with certificate size bounded by kd bits.

2. State True/False. If $A \leq_p B$ and $A \in$ *NP-complete*, then $B \in$ *NP-complete*.

   False. If $A \leq_p B$ and $A \in$ *NP-complete*, then B is not necessarily in *NP-complete* (since B need not be in *NP*).

3. State True/False. Assume you have a polynomial time algorithm that given a *3-SAT* instance, decides in polynomial time if it has a satisfying assignment. Then you can build a polynomial time algorithm that finds a satisfying assignment (if it exists) to a given *3-SAT* instance.

   True. Let A be a polynomial time algorithm that decides *3-SAT*.

   Let $\phi(x_1, x_2, ..., x_n) = c_1 \wedge c_2 \wedge ... \wedge c_m$ be the boolean formula corresponding to a *3-SAT* instance where $c_1, c_2, ..., c_m$ are the clauses and $x_1, x_2, ..., x_n$

are the variables.

If $A(\phi((x_1, x_2, ..., x_n))) = 0$, then return that the instance is non-satisfiable.

If $A(\phi((x_1, x_2, ..., x_n))) = 1$, then we can find an assignment for x_1 as follows. If $A(\phi((1, x_2, ..., x_n))) = 1$, then we can set $x_1 = 1$ and be guaranteed that $\phi_1 := \phi(1, x_2, ..., x_n)$ is satisfiable. Otherwise, we can set $x_1 = 0$ and be guaranteed that $\phi_1 := \phi(0, x_2, ..., x_n)$ is satisfiable.

In either case, we know ϕ_1 has a satisfying assignment. That is, there exists a satisfying assignment for ϕ that assigns to x_1 the same assignment that our algorithm made.

In the second iteration, we find an assignment for x_2 given that we have already made a choice for x_1. If $A(\phi_1(1, x_3, ..., x_n)) = 1$, we can set $x_2 = 1$ and $\phi_2(x_3, ..., x_n) = \phi_1(1, x_3, ..., x_n)$. Else we can set $x_2 = 0$ and $\phi_2(x_3, ..., x_n) = \phi_1(0, x_3, ..., x_n)$.

By iterating, we find an assignment for the variables such that ϕ_n (which is nothing but the evaluation of ϕ at this assignment) is 1.

4. State True/False. If someone proves $P = NP$, then it would imply that every decision problem can be solved in polynomial time.

   False. If $P = NP$, then we can conclude that every problem in NP can be solved in polynomial time. However, there are decision problems (that are not in NP) that are known to not have polynomial time algorithms. One such example is the Halting problem which cannot be solved even if there were no restriction on the resources (like time or space).

5. State True/False. Assume $P \neq NP$. Let A and B be decision problems. If $A \in$ *NP-complete* and $A \leq_p B$, then $B \notin P$.

   True. If B were in P, then $A \leq_p B$ would imply $A \in P$. Since $A \in$ *NP-complete*, $\forall D \in NP, D \leq_p A$. Since $A \leq_p B$, this implies $\forall D \in NP, D \in P$ which contradicts $P \neq NP$.

6. State True/False. Assume $P = NP$. Let A and B be decision problems. If $A \in P$ and $B \in$ *NP-complete*, then $A \leq_p B$.

   True. $B \in$ *NP-complete* implies $\forall D \in NP, D \leq_p B$. Since $P \subseteq NP$, $A \leq_p B$. (Note that we don't use the assumption $P = NP$)