CSCI 570 - HW 10

Graded Problems

1. State True/False

   (a) Assume $P \neq NP$. Let A and B be decision problems. If $A \in NPC$ and $A \leq_p B$, then $B \in P$.

   False. If B were in P, then $A \leq_p B$ would imply $A \in P$. Since $A \in NPC$, $\forall D \in NP, D \leq_p A$. Since $A \leq_p B$, this implies $\forall D \in NP, D \in P$ which contradicts $P \neq NP$.

   (b) If someone proves **P=NP**, then it would imply that every decision problem can be solved in polynomial time.

   False. Halting is a counter example.

   (c) If $A \leq_p B$ and $B \in \mathbf{NP}$, then $A \in \mathbf{NP}$.

   True. Proof : $B \in NP$ implies that there exists a polynomial time verifier V_B and a constant k_B such that $x \in B$ if and only if there exists a c_x such that $||c_x|| \leq ||x||^k$ and $V_B(x, c_x) = 1$. Here, $||y||$ denotes the number of bits used to write y.

   Let $f : \mathbb{N} \longrightarrow \mathbb{N}$ be a polynomial time reduction from A to B with running time bounded by a polynomial of degree d. Compose the algorithms V_B and f to obtain the polynomial time algorithm V_A, that is

   $$V_A(w) := V_B(f(w)), \forall w \in \mathbb{N}$$

   We claim that V_A is a verifier for A with certificate size bounded by kd bits.

   If $w \in A$, then $f(w) \in B$ and $f(w)$ is at most $||w||^d$ bits long. Hence $\exists c_{f(w)}$ such that $||c_{f(w)}|| \leq ||f(w)||^k \leq ||w||^{dk}$ and $V_B(f(w), c_{f(w)}) = 1$.

Conversely, if there exists $\bar{c}_w$ such that $||\bar{c}_w|| \leq ||w||^{kd}$ and $V_B(f(w), \bar{c}_w) = 1$, then $f(w) \in B \rightarrow w \in A$. Hence our claim is true and $A \in NP$.

2. Given an n bit positive integer, the problem is to decide if it is composite. Here the problem size is n. Is this decision problem in **NP** ?

   Yes. For every "yes" instance (the number is composite), a factor of the number is a certificate. Verification proceeds by dividing the number by the factor and making sure that the reminder is zero. The factor is at most n nits and verification can be done in time polynomial in n. Thus deciding if a number is composite is in **NP**.

3. State True/False. Assume you have an 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. Pick the first literal in the first clause (call it x_1). Replace x_1 with 1 (or 0) and $\neg x_1$ with 0 (or 1) in every mention of x_1 and $\neg x_1$ in the satisfying problem. Run the deciding algorithm. If the problem is still satisfiable we go to the next literal (if needed in the next clause) and we do the same process until all the literals are assigned value. If the problem is not satisfiable, this implies that we have set the literal with a wrong value. Then we change the value from 1 to 0 (or 0 to 1) and again we continue with rest of the literals.

4. Show that vertex cover remains **NP**-Complete even if the instances are restricted to graphs with only even degree vertices.

   Let $< G; K >$ be an input instance of VERTEX-COVER, where $G = (V; E)$ is the input graph.

   Because each edge in E contributes a count of 1 to the degree of each of the vertices with which it is incident, the sum of the degrees of the vertices is exactly $2|E|$, an even number. Hence, there is an even number of vertices in G that have odd degrees.

   Let U be the subset of vertices with odd degrees in G.

   Construct a new instance $< \bar{G}; k + 2 >$of VERTEX-COVER, where $\bar{G} = (V_0; E_0)$ with $V_0 = V \cup \{x, y, z\}$ and $E_0 = E \cup \{(x, y), (y, z), (z, x)\} \cup \{(x, v) | v \in U\}$. In words, we make a triangle with the three new vertices, and then connect one of them (say x) to all the vertices in U.

   The degree of every vertex in V_0 is even. Since a vertex cover for a triangle is of (minimum) size 2, it is clear that $\bar{G}$ has a vertex cover of size $k + 2$

if and only if G has a vertex cover of size k.

Practice Problems

5. Given an integer $m \times n$ matrix A and an integer $m - vectorb$, the **0-1integer programming problem** asks whether there exists an integer $n - vectorx$ with elements in the set $\{0; 1\}$ such that $Ax = b$. Prove that 0-1integer programming is NP Complete. (*Hint*: Reduce from 3-CNF-SAT.)

   Given an x, one can easily verify in polynomial time whether $Ax \leq b$, hence, the 0-1 integer-programming problem is in NP. 3-CNF-SAT $\leq_p$ 0-1 INTEGER-PROGRAMMING, that is we reduce the 3-CNF-SAT problem to the 0-1 integer-programming problem to show NP-completeness. Let the 3-CNF-SAT problem have n variables. A variable x in integer-programming corresponds to a $(0, 1)$-integer variable x, the negation of x corresponds to $(1 - x)$. Now each clause would be converted into a row of A. So for example, the clause x or (not y) or z is going to be converted to $x + (1 - y) + z \geq 1$ which is equal to $-x + y - z \leq 0$.

6. Assume that you are given a polynomial time algorithm that decides if a directed graph contains a Hamiltonian cycle. Describe a polynomial time algorithm that given a directed graph that contains a Hamiltonian cycle, lists a sequence of vertices (in order) that forms a Hamiltonian cycle.

   Let $G = (V, E)$ be the input graph. Let A be an algorithm that decides if a given directed graph has a Hamiltonian cycle. Hence $A(G) = 1$.

   Pick an edge $e \in E$ and remove it from G to get a new graph $\bar{G}$. If $A(\bar{G}) = 1$, then there exists a Hamiltonian cycle in $\bar{G}$ which is a subgraph of G, set $G = \bar{G}$. If $A(\bar{G}) = 0$, then every Hamiltonian cycle in G contains e. Put e back into G.

   Iterate the above three lines until we are left with exactly $|V|$ edges. Since after each step we are left with a subgraph that contains a Hamiltonian cycle, at termination we are left with the set of edges that forms a Hamiltonian cycle. Starting from an edge, do a BFS to enumerate the edges of the Hamiltonian cycle in order.