CS 325 Spring 2019, Homework 7

May 24, 2019

1. Let X and Y be two decision problems. Suppose we know that X reduces to Y in polynomial time. Which of the following can we infer? Explain.

   - **If Y is NP-complete then so is X.**
     False. X can either be NP or NP-complete. X is NP-complete if and only if all other problems in NP reduces to X.
   - **If X is NP-complete then so is X.**
     False. If X is not verifiable in polynomial time then X is NP-hard.
   - **If Y is NP-complete and X is in NP then X is NP-complete.**
     False. X can still be in NP and not NP-complete if not all problems in NP reduces to X.
   - **If X is NP-complete and Y is in NP then Y is NP-complete.**
     True. This is the definition of NP-complete.
   - **If X is in P, then Y is in P.**
     False. If Y is in NP then X still reduces to Y.
   - **If Y is in P, then X is in P.**
     True. If X reduces to Y, then X is no more difficult than Y.

2. Consider the problem COMPOSITE:

   "Given an integer y, does y have any factors other than one and itself?"

   For this exercise, you may assume that COMPOSITE is in NP, and you will be comparing it to the well-known NP-complete problem SUBSET-SUM:

   "Given a set S of n integers and an integer target t, is there a subset of S whose sum is exactly t?"

   Clearly explain whether or not each of the following statements follows from that fact that COMPOSITE is in NP and SUBSET-SUM is NP-complete:

   - **SUBSET-SUM $\leq_p$ COMPOSITE**

     This statement is False because COMPOSITE is not necessarily NP-complete just because it is in NP. If we can show that SUBSET-SUM reduces to COMPOSITE in polynomial time then this statement is True.

   - **If there is an $O(n^3)$ algorithm for SUBSET-SUM then there is a polynomial time algorithm for COMPOSITE**

     Because SUBSET-SUM is NP-complete and COMPOSITE is in NP, then COMPOSITE reduces to SUBSET-SUM in polynomial time. Since there is a polynomial algorithm for SUBSET-SUM then there must be a polynomial time algorithm for COMPOSITE.

   - **If there is a polynomial algorithm for COMPOSITE, then P = NP.**

     This statement is only True if COMPOSITE is NP-complete. So far it has not been proven that COMPOSITE is NP-complete.

   - **If P $\neq$ NP, then no problem in NP can be solved in polynomial time.**

     If P $\neq$ NP, some problems in NP may still be solvable in polynomial time. The only conclusion you can draw is that no NP-complete problems can be solved in polynomial time.

3. A Hamiltonian path in a graph is a simple path that visits every vertex exactly once. Prove that HAM-PATH = {(G, u, v): there is a Hamiltonian path from u to v in G} is NP-complete. You may use the fact that HAM-CYCLE is NP-complete.

   In order to prove that HAM-PATH is NP-complete, we need to show that,

   - **HAM-PATH $\in$ NP**
     Given a list of vertices, we can verify that it is a hamiltonian path in G by
     – Checking that we only visit each vertices only once.
     – Checking that there is an edge going from one vertex to the next vertex.

     This can be done in polynomial time.
   - **HAM-PATH $\in$ NP-hard**
     We can show that,

     $$\text{HAM-CYCLE} \leq_p \text{HAM-PATH}$$

     where HAM-CYCLE $\in$ NP-complete.
     A Hamiltonian cycle is simply a Hamiltonian path that ends in one of the adjacent vertex of the source vertex u. Let $G = (V, E)$ be an instance of HAM-CYCLE. We construct an instance of HAM-PATH as follows,
     – Let u be an arbitrary vertex in graph G.
     – Let v be an adjacent vertex of u.

     The instance of HAM-PATH is then <G, u, $adj(u)$> which is easily formed in polynomial time.
     We now show that graph G has a Hamiltonian cycle if and only if graph G has a Hamiltonian path from u to $adj(u)$.
     Suppose graph G has a Hamiltonian cycle h,
     – Then, by removing the final path from $adj(u)$ to u, we have a Hamiltonian path from u that ends in $adj(u)$.

     Suppose graph G has a Hamiltonian path from u to $adj(u)$,
     – Then, by adding a path from $adj(u)$ to u, we get a Hamiltonian cycle.

     Thus, HAM-PATH is NP-complete.

4. K-COLOR. Given a graph $G = (V, E)$, a k-coloring is a function $c : V- > 1, 2, ..., k$ such that $c(u) \neq c(v)$ for every edge $(u, v) \in E$. In other words the number $1, 2, .., k$ represent the k colors and adjacent vertices must have different colors. The decision problem K-COLOR asks if a graph can be colored with at most K colors.

   - The 2-COLOR decision problem is in P. Describe an efficient algorithm to determine if a graph has a 2-coloring. What is the running time of your algorithm?

     For the 2-COLOR decision problem, we can use a Breadth-First Search.

```
BIPARTITE(G, V, src)
    let color[1..V] be a new array
    color[1..V] = -1
    color[src] = 1
    q = queue()
    q.append(src)
    while q is not empty:
        u = q.pop()
        for v in adj(u):
            if color[v] = -1:
                color[v] = 1 - color[u]
                q.append(v)
            elif color[v] = color[u]:
                return False
    return color
```

     The running time of the algorithm is $O(V + E)$.

- It is known that the 3-COLOR decision problem is NP-complete by using a reduction from SAT. Use the fact that 3-COLOR is NP-complete to prove that 4-COLOR is NP-complete.

  In order to prove that 4-COLOR is NP-complete, we need to show that,

  - **4-COLOR $\in$ NP**
    Given a function $c : V- > 1, 2, 3, 4$, we can verify that it is a solution to 4-COLOR by:
    * Checking that c has ≤ 4 colors.
    * For each node u in graph G, check that $c(u)$ has a different color than its neighbors $c(adj(u))$.

    This can be done in polynomial time.
  - **HAM-PATH $\in$ NP-hard**
    We can show that,

    $$\text{3-COLOR} \leq_p \text{4-COLOR}$$

    where 3-COLOR $\in$ NP-COMPLETE
    The reduction maps a graph G into a new graph G' such that $G \in$ 3-COLOR if and only if $G' \in$ 4-COLOR. We do so by adding a new node y and connecting y to each node in G to form G'.
    If G is 3-colorable, then G' can be 4-colored exactly as G with y being the only node colored with the additional color. Similarly, if G' is 4-colorable, then we know that node y must be the only node of its colors - this is because it is connected to every other node in G'. Thus, we know that G must be 3-colorable.
    This reduction takes linear time to add a single node and $|V|$ edges where $|V|$ is the number of vertices in G.
    Since 4-COLOR is in NP and NP-hard, we know it is NP-complete.