Arthur Liou
CS325

HW 7

1. (6 pts) 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.
a. If Y is NP-complete then so is X.
b. If X is NP-complete then so is Y.
c. If Y is NP-complete and X is in NP then X is NP-complete.
d. If X is NP-complete and Y is in NP then Y is NP-complete.
e. If X is in P, then Y is in P.
f. If Y is in P, then X is in P.

Answer: d and f only
Explain: X reduces to Y means that if you had a black box to solve Y efficiently, you could use it to solve X efficiently. X is no harder than Y.

2. (4 pts) 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:

a. SUBSET-SUM $\leq$p COMPOSITE.
No. SUBSET-SUM is NP-complete and so may be reduced to any other NP-complete problem. However, we don’t know that COMPOSITE is NP-complete, only that it is in NP. Hence, we cannot say for sure that SUBSET-SUM reduces to COMPOSITE.

b. If there is an O(n^3) algorithm for SUBSET-SUM, then there is a polynomial time algorithm for COMPOSITE.
Yes. The given running time is polynomial in n. Since SUBSET-SUM is NP-complete, this implies P = NP. Hence, every algorithm in NP, including COMPOSITE, would have a polynomial-time algorithm.

c. If there is a polynomial algorithm for COMPOSITE, then P = NP.
No. COMPOSITE is in NP, but it is not known to be NP-complete. Hence, a polynomial-time algorithm for COMPOSITE does not imply P = NP.

d. If P $\neq$ NP, then no problem in NP can be solved in polynomial time.
No. The class P is a subset of NP, and it is clearly not empty! Proving P $\neq$ NP would show only that NP-complete problems cannot be solved in polynomial time.

3. (8 pts) A Hamiltonian path in a graph is a simple path that visits every vertex exactly once. P 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.

Proof: We need to show that HAM-PATH can be verified in polynomial time. Let the input x be (G, u, v) and let the certificate y be a sequence of vertices {v1, v2, · · · , vn}. An algorithm A(x, y) verifies HAM-PATH by executing the following steps:
1) Check if |G.V | = n;
2) Check if v1 = u and vn = v;
3) Check if ∀i ∈ {1, 2, · · · , n}, vi ∈ G.V ;
4) Check if ∀i, j ∈ {1, 2, · · · , n}, vi 6= vj;
5) Check if ∀i ∈ {1, 2, · · · , n − 1}, (vi , vi+1) ∈ G.E;

If any of the above steps fail, return false. Else return True;
Time Complexity:
Steps 1 and 2 takes O(1) time;
step 3 takes O(V) time;
step 4 runs in O(V^2) time, and
step 5 runs in O(E) time.
Therefore the verification algorithm runs in O(V^2) time. Hence **HAM-PATH ∈ NP**

4. (12 pts) K-COLOR. Given a graph G = (V,E), a k-coloring is a function c: V -> {1, 2, … , k} such that c(u) ≠ c(v) for every edge (u,v) ∈ 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.
a. 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?
1) Do a breadth-first search
2) assigning Color1 to the first layer, Color2 to the second layer, Color1 to the third layer, etc.
3) Then go over all the edges and check whether the two endpoints of this edge have different colors.
Time Complexity: This algorithm is $O(|V|+|E|)$, where v are vertices and e are edges
Resource: https://www.cs.cornell.edu/courses/cs3110/2009fa/recitations/rec22.html

b. 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
Part 1. 4-COLOR is in NP.
The coloring is the certificate (i.e., a list of nodes and colors). The following is a verifier G for 4-COLOR. V = "On input <G, c>"
1) Check that c includes =< 4 colors
2) Color each node of G tas specificed by c
3) For each node, check it has a unique color compared to its neighbors
4) If all checks pass, accept. Otherwise reject
Part 2. 4-Color is NP-hard
We can give a polynomial-time reduction from 3-COLOR to 4-COLOR. This reduction maps a graph into a new Graph such that graph ∈ 3-Color IFF newGraph 4-Color. If the graph is 3-

colorable, then newGraph can be 4-colored exactly as the 3-colored graph but with an a new node Y and connecting Y to each node in newGraph. Y would then be colored with the new / additional color. This reduction takes linear time to add a single node and G edges.
Since 4-COLOR is in NP and NP-hard, **proof that 4-COLOR is NP-complete.**