CS 325 - Homework 5

Tyler Cope

CS325 Homework 5

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

   a. If Y is NP-complete then so is X.

      We can't infer this. X could be NP and not NP-complete.

   b. If X is NP-complete then so is Y.

      Can't be inferred. It's possible Y could be stricter than NP.

   c. If Y is NP-complete and X is in NP then X is NP-complete.

      X could just be NP. Therefore, we can't infer this.

   d. If X is NP-complete and Y is in NP then Y is NP-complete.

      This can be inferred. Y is in NP and is at least as hard as X. Therefore, Y is NP complete.

   e. X and Y can't both be NP-complete.

      We can't infer this and part d explains a scenario that lays out the opposite.

   f. If X is in P, then Y is in P.

      Can't be inferred. Y could be in NP and still satisfy the condition.

   g. If Y is in P, then X is in P.

      This can also be inferred. P is the least strict so if Y is in it, X must be as well.

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:

   a. SUBSET-SUM $\leq_p$ COMPOSITE.
      This statement does not follow. SUBSET-SUM is NP complete so we know it can be reduced. However, we only know that COMPOSITE is in NP, not NP-complete. Without knowing that, we don't know if SUBSET-SUM can be reduced to COMPOSITE.

   b. If there is an $O(n^3)$ algorithm for SUBSET-SUM, then there is a polynomial time algorithm for COMPOSITE.
      This statement holds. SUBSET-SUM is NP-complete and if it's solvable in $O(n^3)$ then all NP problems are solvable in that time.

   c. If there is a polynomial algorithm for COMPOSITE, then P = NP.
      This statement does not hold because we don't know if COMPOSITE is in NP-complete.

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

This statement holds because we know that if a problem in NP-complete can be solved in a certain time then all NP problems are solvable in that time. If you negate that then this is the statement that you get.

3. Two well-known NP-complete problems are 3-SAT and TSP, the traveling salesman problem. The 2-SAT problem is a SAT variant in which each clause contains at most two literals. 2-SAT is known to have a polynomial-time algorithm. Is each of the following statements true or false? Justify your answer.

a. 3-SAT $\leq_p$ TSP.
From lecture, we know that TSP can be reduced. So this is true.

b. If P $\neq$ NP, then 3-SAT $\leq_p$ 2-SAT.
This is not true. 3-SAT is in NP-complete and 2-SAT is polynomial. Because of this, 2-SAT is within P and if 3-SAT could be reduced to 2-SAT it would be in P and NP-complete, which can't happen based on the condition.

c. If P $\neq$ NP, then no NP-complete problem can be solved in polynomial time.
Similar to question 2d. This statement is true because by definition if one NP-complete problem can be solved in polynomial time then all others can as well.

4. LONG-PATH is the problem of, given (G, u, v, k) where G is a graph, u and v vertices and k an integer, determining if there is a simple path in G from u to v of length at least k. Show that LONG-PATH is NP-complete.

First, we need to verify that LONG-PATH is within NP. If we look at the vertices that make up a path, we can traverse them and check if the next vertex is in the list (using an adjacency matrix) in O(n) time. This shows that it's in NP. To get to NP-complete, solve it using a Hamiltonian path. We know that finding a Hamiltonian path is NP-complete so we just need to solve G using a Hamiltonian path. Then you just need to check each vertex to make sure there is an edge that connects it to the next vertex. This is O(V + E) time.

5. Graph-Coloring. Mapmakers try to use as few colors as possible when coloring countries on a map, as long as no two countries that share a border have the same color. We can model this problem with an undirected graph G = (V,E) in which each vertex represents a country and vertices whose respective countries share a border are adjacent. 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 graph-coloring problem is to determine the minimum number of colors needed to color a given graph.

a. State the graph-coloring problem as a decision problem K-COLOR. Show that your decision problem is solvable in polynomial time if and only of the graph-coloring problem is solvable in polynomial time.
A decision problem is: Given a graph, G, with Vertices, V and Edges, E, and some positive integer, k, can the vertices be colored such that each vertex has a neighbor with a different

color? We can check the decision in O(E) time (we just need to look at the edges to see the color of each vertex associated with that edge). So now we know that if a graph is polynomial then it's solvable in polynomial time.

b. It has been proven that 3-COLOR 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.
   We can prove this by adding in a new node to a graph. This new node will be connected. Then, we solve the first part of the graph without the new node with 3-COLOR. If it is possible, we know that we can use 4-COLOR because there will only be one additional node that doesn't have a color.