Eddie C. Fox

May 30, 2017

CS 325 – Analysis of Algorithms

Assignment 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.
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. X and Y can't both be NP-complete.
f. If X is in P, then Y is in P.
g. If Y is in P, then X is in P.

A, B, C, E, and F are false, and therefore cannot be inferred.

D and G are true.

Explanations Below

A is false because reducing X to Y might not yield an optimal solution for X.

B is false because we cannot tell if Y is NP-complete because only know that Y is NP-hard. We are not sure if it is also in NP.

C is false because if X reduces to Y in polynomial time and Y is NP-complete, we would expect X to be a member of NP hard.

D is true because X reduces to Y. Meaning if you could solve Y efficiently with a black box, you could use it to solve X as well. If X is NP-complete, then all problems in NP are reducible to X. So Y being in NP would be Y is NP complete as well.

E is false because all NP-complete problems are essentially the same problem. If X is NP complete and Y is in NP, then Y is NP complete as well, which counters the assertion that they can't both be NP complete.

F if false because given that X reduces to Y, X should be at least NP. Even assuming that X is in P, this doesn't change what we know about Y because reducing X to Y might not be an optimal solution for X.

G is true because X reduces to Y. Y is an intermediate step in solving X. Because Y can be solved in polynomial time (being in P), and X reduces to Y, X is also solvable in polynomial time.

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.

False. Because Subset-sum is NP-complete, it can be reduced to any other NP-complete problem, but we don't know if COMPOSITE is NP-complete, we only know that it is in NP, so we can't conclusively say that SUBSET-SUM can reduce to it.

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

True. If there was a polynomial solution for an NP complete problem (SUBSET-SUM) there would be a polynomial time algorithm for any problem in NP, which includes COMPOSITE.

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

False. This would only be true if COMPOSITE happened to be NP-complete, but we only know it to be in NP. The same thing that is wrong with Part A.

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

If P is not equal to NP, that just means there is no NP-COMPLETE problems that can't be solved in polynomial time, not all problems in NP problems which are broader. P is still in NP and P problems have polynomial time solutions.

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.

True. Any NP-complete problem can reduce to any other NP-complete problem.

b. If P $\neq$ NP, then 3-SAT $\leq_p$ 2-SAT.

False. Because 2-SAT is in P, a reduction from 3-SAT to 2-SAT would imply a polynomial algorithm for 3-SAT which is impossible because we are assuming P is not equal to NP, which prevents the possibility of a polynomial algorithm for NP-complete problems.

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

True. If an NP-complete problem had a polynomial time solution, then all NP-complete problems would have polynomial time solutions, and then all NP-complete problems would be in P, and thus P would equal NP, which contradicts the assumption we are starting with that P is not equal to NP.

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.

We know that LONG-PATH is in NP because we can use Dijkstra's algorithm or something else to find the shortest path and then compare it to length integer K. This can easily be done and verified in polynomial time. A Hamiltonian path is a simple path that visits each vertex exactly once. Hamiltonian paths are actually special variations of the LONG-PATH problem where K is equal to the number of vertices in the graph minus 1. You could consider Hamiltonian Path = LONG-PATH(G,u,v, v-1). What exactly K is equal to should not change the complexity of the problem, so LONG-PATH should have whatever complexity Hamiltonian path has. The Hamiltonian path is NP complete, so LONG-PATH is also NP-complete.

Proof that Hamiltonian Path is NP complete:

The Hamiltonian cycle problem is NP complete and reduces to the Hamiltonian path problem.

Proof Hamiltonian cycle reduces to Hamiltonian path:

Given a G = (V,E) instance of Hamiltonian cycle, we could create a new graph G". For every edge between vertices u and v, create two vertices v' and v" that are connected to u and v respectively. For every edge, the following holds: there is a Hamiltonian cycle in G if and only if there is a Hamiltonian path in G". Therefore graph G has a Hamiltonian cycle if and only if some edge in G" has a Hamiltonian path. So we can reduce the cycle problem to the path problem, and Hamiltonian path is NP-complete.

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.

Decision problem K-COLOR: Given a Graph G(V,E) and integer k, can we color the graph with k colors such that adjacent vertices have different colors?

If this decision problem K-COLOR is solvable in polynomial time, then we can use the K-color problem starting with K = the number of verticies in the graph all the way down to 1 color, stopping whenever k reaches a value where there is no longer a sufficient number of colors to color everything. The solution to the graph coloring problem is the smallest number of K from the results above that was still colorable.

On the other hand, if the graph coloring problem was solvable in polynomial time, we could just compute the solution to the graph coloring problem in polynomial time and compare its value (v) to the integer k from the decision problem. If k is less than v, then we can say the answer to the decision problem is no, and if k is greater than v, we can say the answer to the decision problem is yes. For example, if we are trying to find the solution to 3-COLOR, and the graph coloring problem says 4 colors minimum are needed, we know we cannot color a graph with just 3 colors, because it isn't enough, so we know that the answer to the decision problem 3-COLOR for that graph is no.

Regardless of which problem is polynomial, we can deduce a polynomial solution to other one, so they must follow necessarily from each other.

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.

4-COLOR is NP-complete because it is in both NP and NP hard, which satisfies the definition for being NP-complete.

First, we can tell that 4-color is in NP.

We can verify a solution in polynomial time by proposing the following tests.

1. Check that a graph contains 4 or less colors.

2. Color each node as specified by the solution.

3.. For each node, check that it has a unique color from its neighbor.

If these tests pass, the solution is valid, otherwise, it is invalid. This is similar to the breadth first search coloring tests we did earlier, which we determined was also polynomial with O(V+E).

Next, we prove that 4-color is NP-hard.

3-color has been proven to be NP-complete per the assumptions of the problem. Because 3-color is NP complete, it must be NP hard, so if we can prove that the 4 color problem can be reduced from the 3 color problem in polynomial time, we can prove the 4 color is NP-hard.

The reduction takes a graph G and creates a new graph G” such that G is an element in the set of 3-colored graphs if G” is 4 colored. We can do this by assigning G” the graph G and adding a new node Y that is connected to all other nodes. If G is 3-colorable, then G” can be 4-colorable by using the new node with only the new node being colored an additional color. This takes linear time or close to it. Only the time it would take to add the new node, connect it to all other nodes, and color it, which only depends on the number of nodes there are. Adding the node and coloring it have constant complexity, added to the linear O(n) complexity of connecting the node to all others.

Because we can reduce the 4 color problem from the 3 color problem in polynomial time, 4-color is NP hard, and thus NP-complete when combined with our earlier NP proof.