Class: CS-325

Term: Fall 2017

Author: Jon-Eric Cook

Date: November 19, 2017

Homework: #7

Problem 1: (7 points)

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 in 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.

Answer:

A) A **cannot** be inferred because it could be possible for X to only be in NP.

B) B **cannot** be inferred because it could be possible for Y to be in NP-hard.

C) C **cannot** be inferred because it could be possible for X to be in P.

D) D **can** be inferred because Y can be in NP and also in NP-complete.

E) E **cannot** be inferred because both X and Y could be NP-complete because reduced X could be NP-complete and reduce to Y, which is also NP-complete.

F) F **cannot** be inferred because it could be possible for Y to be harder than X.

G) G **can** be inferred because X is no harder than Y.

Problem 2: (4 points)

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 the fact that COMPOSITE is in NP and SUBSET-SUM is NP-complete:

A) SUBSET-SUM <=p COMPOSITE

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

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

D) If P != NP, then no problem in NP can be solved in polynomial time.

Answer:

A) With knowing that COMPOSITE is in NP and SUBSET-SUM is NP-complete, the statement saying that SUBSET-SUM can reduce to COMPOSITE is **false**. It does not make sense for a a problem to be reduced to an easier problem. Because of this, SUBSET-SUM may be reduced to any other NP-complete problem and not to a NP problem, of the likes of COMPOSITE. It is not known if COMPOSITE is NP-complete, only that it is NP.

B) It is **true** that if there is an O(n^3) algorithm for SUBSET-SUM, which is in NP-complete, then there is a polynomial time algorithm for COMPOSITE, which is in NP. This is so because NP-complete is in NP, thus a solution for an NP-complete problem can solve an NP problem. Essentially, because an NP-complete problem can be solved in polynomial time, all can be solved, thus, P = NP.

C) The statement that if there is a polynomial algorithm for COMPOSITE, the P = NP is **false**. The is so because even though COMPOSITE is in NP, it is not known if it is also NP-complete. Because of this, it cannot be known for sure if P = NP.

D) The statement that if P != NP, then no problem in NP can be solved in polynomial time is **false**. This statement is false because if in fact P != NP then it shows that there is not an algorithm that solve all problems in NP in polynomial time. There can still be solutions to NP problems that can be solved in polynomial time, there just isn't one solution for all problems in NP.

Problem 3: (3 points)

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.

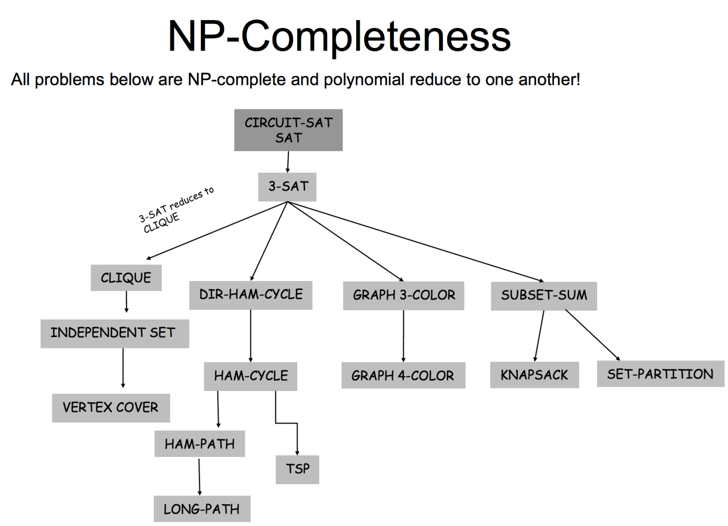
A) 3-SAT <=p TSP

B) If P != NP, then 3-SAT <=p 2-SAT

C) If P != NP, then no NP-complete problem can be solved in polynomial time.

Answer:

A) It is **true** that 3-SAT can be reduced to TSP because for one, both 3-SAT and TSP are both NP-complete problems. Also, based on the lecture slide below, 3-SAT gets reduced to TSP but first being reduced to DIR-HAM-CYCLE, then to HAM-CYCLE and then finally to TSP.



B) It is **false** that if P != NP, then 3-SAT can be reduced to 2-SAT. This is so because P != NP means that there would be no polynomial time algorithm that can solve 3-SAT. Conversely, we know 2-SAT is in P so if 3-SAT can be reduced to 2-SAT then 3-SAT would also be in P. This is a contradiction to the first statement.

C) It is **true** that if P != NP then no NP-complete problem can be solved in polynomial time. This is so because if an NP complete problem can be solved in polynomial time, then all NP-complete problems can be solved in polynomial time. So if P != NP, then by definition, no NP-complete problems can be solved in polynomial time.

Problem 4: (6 points)

A Hamiltonian path in a graph is a simple path that visits every vertex exactly once. Show 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.

Answer:

First, if we are given a solution to HAM-PATH, we are able to verify it in polynomial time. This would be done by taking the answer (the path through the graph) and confirming that there are no repeats and that the first node is u and the last node is v. Because of this fact, HAM-PATH is in NP. Second, referencing the screen shot above, it is known that HAM-CYCLE is NP-complete and reduced to HAM-PATH. For HAM-CYCLE to be reduced to HAM-PATH, we can assume we have the same graph with the same number of edges. As a side note, a HAM-CYCLE in a graph is a simple path that visits every vertex exactly once and starts and ends at the same node. From this knowledge, reducing a HAM-CYCLE to a HAM-PATH is trivial as if there is a HAM-CYCLE in a graph then there must be a HAM-PATH as all nodes are connected and there is a path through the graph where each vertex is visited exactly once. Thus, reducing a HAM-CYCLE to a HAM-PATH is trivial and therefore the HAM-PATH is NP-complete.

Problem 5: (5 points)

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

Answer:

First, if we are given a solution to LONG-PATH, we are able to verify it in polynomial time. This would be done by making sure the edges given are in G and also the length of the path is less at least k. Because of this fact, LONG-PATH is in NP. Second, referencing the screen shot above, it is known that HAM-PATH is NP-complete and reduces to LONG-PATH. For HAM-PATH to be reduced to LONG-PATH, we can assume we have the same graph with the same number of edges. From here, we can find the longest path it takes to traverse through all the vertices without repeating an edge. This is a lot like LONG-PATH except we are trying to find a path from u to v that is at least k. We can simply make all the edges have a weight of 1 and then try and find a path from u to v that crosses at least k number of edges. Thus, proving that LONG-PATH is NP-complete.