CS 325 Homework Assignment 7
Peter Moldenhauer
11/13/17

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 not infer this because it is possible for X to only be NP. We can only infer that Y is at least as hard as X, so it is possible for Y to be "harder" than X.

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

We can not infer this because Y only needs to be at least as hard as X. It is possible for Y to be NP-hard and not NP-complete because it is still at least as hard as X.

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

We can not infer this because it is possible for X to only be NP. Y only has to be at least as hard as X, so it is possible for Y to be "harder" than X.

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

We can infer this because Y has to be at least as hard as X while still being in NP. Therefore, Y has to also be NP-complete to meet these requirements.

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

We can not infer this because it is possible for X and Y to both be NP-complete. Since X reduces to Y and if X is NP-complete, Y has to be at least as hard as X. So if it is of the same "hardness" (NP-complete) this still meets the requirement.

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

We can not infer this because it is possible for Y to be NP. Y only has to be at least as hard as X, so it is possible for Y to be "harder" than X and thus not in P.

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

We can infer this because Y has to be at least as hard as X. It is not possible for X to be "harder" than Y and thus both X and Y have to be in P.

2 Consider the problem COMPOSITE: given and 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 <=p COMPOSITE

No, SUBSET-SUM can not be reduced to COMPOSITE. This is because SUBSET-SUM is NP-complete which means that the problem it is being reduced to must also be at least as hard as NP-complete. However, this is not the case because COMPOSITE might only be NP and might not be NP-complete. We only know that COMPOSITE is in NP (not if it is NP or if it is NP-complete). COMPOSITE would have to be at least NP-complete for this statement to be true – but this is not known for sure.

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

Yes, if so, this would be a true statement. Since SUBSET-SUM is NP-complete, this would mean that P = NP. Therefore, since COMPOSITE is in NP (either NP or NP-complete), it is at most as hard as SUBSET-SUM. This would mean that there would be a polynomial time algorithm for COMPOSITE as well. If there is a polynomial time algorithm for a NP-complete problem then there is indeed a polynomial time algorithm for other NP-complete problems as well and most certainly a polynomial time for NP problems (which are not as "hard" as NP-complete problems).

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

No, we can not say this to be true. This is because we know that COMPOSITE is in NP, but we don't know if it is only NP or NP-complete. COMPOSITE would have to be NP-complete for P = NP and we do not know this for sure.

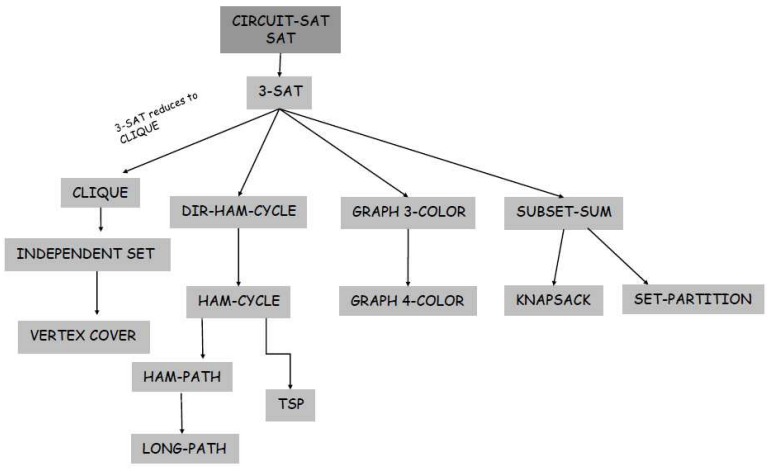
d. If P != NP, then no problem in NP can be solved in polynomial time.

No, this would not be true. This is because if P != NP, this only means that NP-complete problems can not be solved in polynomial time. It may still be the case that NP problems can be solved in polynomial time.

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 <=p TSP

True. This is because both 3-SAT and TSP are NP-complete problems. The graph below is from the lectures and all of the problems in the graph are NP-complete problems. 3-SAT and TSP are both contained in this graph. We know that all NP-complete problems polynomial reduce to one another. Therefore 3-SAT can indeed be reduced to TSP.



b. If P != NP, then 3-SAT <=p 2-SAT

False. This is because if P != NP, this means that NP-complete problems can not be solved in polynomial time. Since 3-SAT is an NP-complete problem, it would not be able to be solved in polynomial time. However, we know that 2-SAT is a polynomial time algorithm. Therefore, 3-SAT <=p 2-SAT would mean that a non-polynomial time algorithm is reduced to a polynomial time algorithm. This would not be possible. For it to be possible, 2-SAT would have to be at least as hard as 3-SAT, but it is not.

c. If P != NP, then no NP complete problem can be solve in polynomial time

True. This is because if P does equal NP, then this means that NP-complete problems can be solved in polynomial time. However, by definition, P = NP is NP-complete. Therefore, if P != NP, then by definition, no NP-complete problems can be solved in polynomial time.

4 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.

To prove that HAM-PATH is NP-complete we have to show 1) that HAM-PATH is NP (the "answer" can be verified in polynomial time) and 2) that R reduces to HAM-PATH in polynomial time (for some R that is an NP-complete problem).

We can prove 1) based on the following: HAM-PATH is in NP because if we are given a solution, we can verify the solution in polynomial time. With a given solution for HAM-PATH, we can first check to make sure that all vertices in the path are all contained in the original graph. We can then check to see if all of the vertices are visited exactly once – no vertices are repeated more than once. Lastly, we can check to make sure that each edge in the patch is indeed contained in the original graph. Through these steps we can verify that the given solution is correct or not and we can do this verification in polynomial time.

We can then prove 2) based on the following: We know that the HAM-CYCLE problem is NP-complete so we need to show that HAM-CYCLE can be reduced to HAM-PATH in polynomial time. To do this we can take a given instance of HAM-CYCLE (G) and manipulate G so that it works with HAM-PATH. We can manipulate G by choosing an arbitrary node v in G and split it into two nodes to get graph G'. Now, any HAM-PATH must start at v' and end at v". If G' has a HAM-PATH then the same ordering of nodes (after we put v' and v" back together) is a HAM-CYCLE in G. Similarly, if G has a HAM-CYCLE, then the same ordering of nodes is a HAM-PATH in G' (if we split v into v' and v"). Therefore, this shows that HAM-CYCLE can be reduced to HAM-PATH in polynomial time.

Since we proved both 1) and 2) we have shown that HAM-PATH is indeed NP-complete.

5 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. Prove that LONG-PATH is NP complete.

To prove that LONG-PATH is NP-complete we have to show 1) that LONG-PATH is NP (the "answer" can be verified in polynomial time) and 2) that R reduces to LONG-PATH in polynomial time (for some R that is an NP-complete problem).

We can prove 1) based on the following: LONG-PATH is in NP because if we are given a solution, we can verify the solution in polynomial time. We can

verify this solution by first checking to make sure that the given edges are in G and we can traverse from u to v. Then we can also add up all of the edges to check to make sure that the edges are at least k. The running time of verifying the solution to LONG-PATH would be O(E) which is based on the number of edges. This is indeed polynomial time.

We can then prove 2) based on the following: We know that the HAM-PATH problem is NP-complete and it reduces to the LONG-PATH problem (see graph from lectures in problem #3). We can see that with HAM-PATH, if we have the same graph G with a set of vertexes and edges, we can find the longest path it takes to traverse through all the vertices without repeating an edge. This is very similar to LONG-PATH except that we are finding the longest path from u to v that is at least k. If we assume that the graph is connected, and each edge is 1, we can set k to be the number of vertices – 1. This will "transform" HAM-PATH to LONG-PATH. Therefore, this shows that HAM-PATH can be reduced to LONG-PATH in polynomial time and thus proving that LONG-PATH is NP-complete.

Since we proved both 1) and 2), we have shown that LONG-PATH is indeed NP-complete.