Lachlan Sinclair

Homework 7

8/9/2019

Problem 1)

a) False. Here X just has to have a time complexity that is the same or less than Y. Since Y is NP complete X could technically be P therefor the statement is false.
b) False. This statement doesn't say exactly what Y is, it could be NP hard, all we know is that its complexity is equal to or greater than X's.
c) False. X can reduce to Y which is NP-complete, however we don't know if X is NP-hard so we cant say its NP-complete.
d) True. Since X can reduce to Y and X is NP-Complete, Y is NP-hard and since Y is also NP it is NP-complete.
e) False. Since X can reduce to Y, we know Y is at least P, but it could also be NP-hard we don't know.
f) True. Since X can reduce to Y, we can infer that X is easier to solve than Y, and Y is P therefor X is P.

Problem 2)

a) We know that SUBSET-SUM is NP-complete and therefor can be reduced to NP-Complete problems, however COMPOSITE is just NP. Since we don't know if COMPOSITE is NP-complete (we don't know if its NP-hard) we cannot be sure that SUBSET-SUM can be reduced to COMPOSITE therefor the **given statement does not follow**.
b) If there was a polynomial algorithm $O(n^3)$ for SUBSET-SUM which is a NP-complete problem, that would then mean NP=P, so all NP problems will also be able to use a polynomial time algorithm. Since COMPOSITE is a NP problem it too would have a polynomial time algorithm, **therefor the given statement follows.**
c) Since COMPOSITE isn't a NP-complete problem as far as we are aware, finding a polynomial runtime algorithm for it will not mean P=NP. Therefor the **given statement does not follow.**
d) If P does not equal NP it follows that no NP-complete problem can be solved in polynomial time, however this does not mean NP problems will not be able to be solved in polynomial time. Therefor the **given statement does not follow.**

Problem 3)

First, we will show that the certificate of HAM-PATH can be validate in polynomial time. Given a certificate it clear that one can validate the solution by making sure every vertex is visited exactly once in polynomial time therefor HAM-PATH is in NP.

Since we know HAM-CYCLE is NP complete we will reduce it to HAM-PATH proving that HAM-PATH is also NP-hard. Let there be some graph G=<V,E> that contains a Hamiltonian cycle and we will make a graph G' such that G contains a Hamiltonian cycle if and only if G' contains as Hamiltonian path. In order create G' will need to duplicate a vertex an arbitrary vertex s making

s'. s will act at the starting/end point of the cycle. s' will be linked to all the same edges as s. To get the Hamiltonian path in G' start at vertex s then follow the Hamiltonian cycle until it would return to s, instead of returning to s travel to s' making the route traveled a Hamiltonian path.

Conversely, if you have graph G' that contains a Hamiltonian path it must then have a starting vertex s and an end point s'. Create graph G by removing the vertex s', then follow the route traveled by the path starting at s, then rather than ending at s' end at s, since s and s' share all the same connections this will possible. Therefor graph G contains a Hamiltonian cycle. Thus G contains a Hamiltonian cycle if and only if G' contains a Hamiltonian path. Therefor HAM-PATH is both NP-hard and NP, which makes its NP-complete.

Problem 4)

a) I would use a BFS to implement 2 coloring. There will be two colors, Yellow and Blue. I will also be using the white, grey, black vertex tracking method discussed in the text book.

   Loop while every vertex is not set to black, add an arbitrary white vertex to the queue. Then in a new loop while the queue isn't empty remove the vertex from the front of the queue and set that vertex to the color Yellow, and set all adjacent vertices to Blue, and make the adjacent vertices grey then add them to the stack. Set the current node to black. If at any point if the algorithm tries set an adjacent vertex to a color that doesn't match the color it is already set to the graph cannot have 2-coloring and the algorithm can end returning this result. This process will continue until the queue loop searches every vertex connected to the first selected vertex, this then repeats for every non connected subgraph. If the algorithm processes every vertex to black, then the graph has 2-coloring. The runtime of my algorithm is O(V+E) when implemented with an edge list. Which is polynomial runtime.

b) First, we will show that the certificate for 4-Color can be validated in polynomial time. Given the certificate make sure there are 4 or fewer colors used, then color each node as described in the certificate. Then loop through each edge and make sure the vertices it connects don't share a color, if no vertices share a color with an adjacent vertex it is Yellow, otherwise it is Blue. This E comparisons where E is the number of edges, this can clearly be done in polynomial time therefor 4-color is NP.

   Next, we will reduce 3-COLOR to 4-COLOR in order to show 4-color is at least NP-hard. Suppose we have some graph G that is a 3-COLOR graph, we will reduce this graph into a 4-COLOR graph G', such that G is 3-COLOR if and only if G' is 4-COLOR. To create G' take the G and add on a vertex v, then connect vertex v to all other nodes. Color v a 4^{th} color that doesn't match the existing three colors in G, this make G' 4-colored. Conversely, since G' is 4-colored and v is the only vertex containing the 4^{th} color, removing vertex v produces a 3-colored graph, which is equivalent to G. Therefor 4-COLOR is in NP-hard and we already know it is NP which makes it NP-complete.