Casey Levy - CS 325 – HW 7

Problem 1

a) No. X can be NP-complete, NP, or P since we can't infer it is NP-complete if Y is
b) No. Y can't be NP-complete since Y is not reducible to X, but X is to Y
c) No. X can be solved by Y but Y cannot be solved by X. This is because X can be reduced to Y
d) Yes. Y is NP-complete
e) No. If Y is P, then X is in P as well since X can be reduced to Y
f) Yes. X must be in P since X reduces to Y.
We can infer d and f

Problem 2

a) Invalid. An NP-complete problem isn't guaranteed to reduce to a problem in NP.
b) Valid. SUBSET-SUM is in NP since it is NP-complete. All problems in NP reduce to SUBSET-SUM, therefore, all problems in NP bound to poly algorithm. There is a poly-time algorithm for COMPOSITE.
c) Invalid. COMPOSITE is one instance of problems in NP. Even though COMPOSITE can be solved in poly-time, this doesn't guarantee all other problems in NP can also be solved in poly-time. This results in P = NP being untrue if COMPOSITE has polynomial algorithm.
d) Invalid. Problems in NP-complete can have polynomial time.

Problem 3

We first need to prove HAM-PATH is in NP. We can do this by choosing edges from G that will be included in the overall path. We ensure each vertex is visited once by going through the path and this task can be done in polynomial time hence HAM-PATH being in NP. The next step is reducing a problem in NP-complete to HAM-PATH.

We show that HAM-CYCLE can be reduced to HAM-PATH in polynomial time. If a graph has HAM-CYCLE, then it will also always have a HAM-PATH, but not all graphs with a HAM-PATH will have HAM-CYCLE. Additionally, finding a HAM-CYCLE can be reduced to finding a HAM-PATH using polynomial time.

If we have a graph G = (V, E), we build another graph (G') such that G has a HAM-CYCLE only if G' has a HAM-PATH. If G' has a HAM-PATH, we can transform this into a HAM-CYCLE in G.

Using the statements above, we can infer that **HAM-PATH is NP-complete.**

Problem 4

a) We can use a BFS algorithm to achieve our goal
i. We can start the algorithm on any random vertex

ii. Starting on level 0, vertices with be colored accordingly, then they will be colored differently at level 1, then the original color at level 2, and so on. Vertices will be colored alternately based on their level.

iii. Check to see if the colors of 2 vertices are different given each edge

iv. If an edge has different colors on each connected vertex, return True. Else, otherwise return False. True proves we have a 2-COLOR graph, False obviously says we don't.
Runtime: O(V + E)

b) We can prove 4-COLOR is in NP.
Similar to Part A above, we can check that a graph G is 4-COLOR and is in NP if we check that each node has a different color from each neighbor. If there are no edges with the same colors on both neighbors, G is 4-COLOR. Else, G is not.
This results in G being 4-COLOR.

We can also reduce 3-COLOR to 4-COLOR which would result in proving 4-COLOR is NP-hard. We already know 3-COLOR is NP-complete, thus making it NP as well. We take a graph G and reduce to a new G, G', such that if G belongs to 3-COLOR only if G' also belongs to 4-COLOR. We get G' by adding a new node (***n***) to G and connecting ***n*** to all nodes in the graph. We can get G if ***n*** is removed from G'.
If G belongs to 3-COLOR, then coloring the new node n with a new color makes G' belong to 4-COLOR. If each mentioned graph belongs to their respective parts, we have proved that G belongs to 3-COLOR if and only if G' belongs to 4-COLOR.
4-COLOR is NP and NP-hard, thus proving that 4-COLOR is NP-complete.