Vlad Predovic

CS 325

Homework 4

1. Which of the following can we infer from the fact that the traveling salesperson problem is NP-complete, if we assume that P is not equal to NP? Explain

a. There does not exist an algorithm that solves arbitrary instances of the TSP problem.

b. There does not exist an algorithm that solves an arbitrary instance of the TSP problem in polynomial. time.

Since we assume that P!= NP and TSP problem is NP-complete, then there cannot exist a polynomial time(P) algorithm for any NP-complete problem.

c. There exists an algorithm that solves arbitrary instances of the TSP problem in polynomial time, but no one has been able to find it.

d. The TSP is not in P.

Similar concept to b, if P!= NP, then then an NP-complete problem cannot be in P.

e. All algorithms that are guaranteed to solve the TSP run in polynomial time for some of the inputs.

f. All algorithms that are guaranteed to solve the TSP run in exponential time for all of the inputs.

2. Let X and Y be two decision problems. Suppose we know that X reduces to Y. 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.

Since X reduces to Y, then if we used Y as a black box to solve X, Y must be NP-hard. However, if Y is also shown to be in NP, then the conditions are met for Y to be 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.

If Y is in P, then X must be in P if X reduces to Y because X is no harder than Y.

3. 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.

Hamiltonian-Cycle = NP-Complete

We can reduce Hamiltonian-Cycle to Ham-Path, therefore Ham-Cycle is no harder than Han-Path.

Step 1: Give an algorithm showing that the solution can be verified in polynomial time.

Given a graph G with n vertices and a list L of adjacent vertices [(x1, x2), (x2, x3)…]containing the supposed Hamiltonian-path, we can verify that the solution is correct in polynomial time. This is done by checking that L is of size n-1 and that each vertice is adjacent to a different one in a formal order. No vertice can appear twice in the adjacency list as both the exit node and entry node. This algorithm run sin O(n^2) time with the constraining process being checking that each node doesn't appear multiple times as the exit and entry node. Therefore, the solution can be considered to be in NP.

Step 2: Show that R ≤ HAM-PATH, in other words that HAM-CYCLE is just as hard as HAM-PATH

We choose R to be HAM-CYCLE. Now to show Ham-Cycle ≤ HAM-PATH:
Given a graph G = {V, E}, create a similar graph G' where a given vertex v' has the same edges as v in graph G. If a hamiltonian cycle exists in G, then a hamiltonian path must exist in G'. Since we also know that HAM-CYCLE is NP-Complete, we can now say that **HAM-PATH is NP-Complete**.

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.

Step 1:

To verify this problem given a solution longest path L:

Check that P consists of k edges	O(1)
Check that no edge is used more than once	O(n^2)

As seen above this algorithm takes polinomial time to accomplish.

Step 2:

This problem becomes much simpler after showing that HAM-PATH is NP-Complete because we know that HAM-PATH ≤ LONG-PATH which is to say it can be solved using LONG-PATH. Essentially HAM-PATH is the problem of LONG-PATH where k = n – 1 and n is the amount of vertices in the graph. Therefore, LONG-PATH must be 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. The a k-coloring is a function c: V -> {1, 2, … , k} such that c(u) □ c(v) for every edge (u,v) □ 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. Give an efficient algorithm to determine a 2-coloring of a graph, if one exists.

The algorthim used implements the BFS method:

Begin at starting node, fill it with color 1.

For every node connected to this first one: fill with color 2

For every node on this second level, go to each adjacent node and fill with color 1.

Continue this process until either all nodes are filled or a 2-coloring is shown to be impossible.

The total time for this algorithm using a que is O(m+n) where m = edges and n = vertices.

b. Cast 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.

If the graph-coloring problem is solvable in polynomial time, then we can solve the problem k -1 times (from k colors down to 2 colors). Solving this each time will result in a decision, yes or no, on whether the graph is solvable. The minimum value obtained that results in 'yes', is the solution for the graph coloring problem.

Additionally, if the problem can be solved in polynomial time and we know the minimum number of colors needed (lets call it j), then we can just compare the value of j to the given k. If k < j, then the problem is obviously not solvable and the answer is 'no'.

As a result, we have shown that the graph-coloring problem can essentially be reduced to the decision problem K-COLOR and so if the graph-coloring problem is solvable in polynomial time then the decision problem must be solvable in polynomial time.

c. Show that K-COLOR is in NP. That is if you are given the coloring function c, you can verify that the graph is K-colorable.

As shown in part b, if given the coloring function, all we have to do is verify whether the colored graph is possible or not with K colors. The checking algorithm consists of passing through each adjacency list and checking that no neighbors are of the same color. This is doable in polynomial time. At this point all that is left to check is that at most k colors are used (for the reasons in part b).

d. It has been shown that 3-COLOR is NP-complete by using a reduction from SAT. SAT is NP-complete and there exists a polynomial time algorithm that maps satisfiable formulas to 3-colorable graphs and non-satisfiable formulas to non-3-colorable graphs. Use the fact that 3-COLOR is NP-complete to show that 4-COLOR is NP-complete.

Part1: Show 4- COLOR is in NP (can be verified quickly)

Given a graph G with each node given a color in array K,

1. Check that $K \leq 4$
2. For each node(BFS)
    a. Check the color it has different from all of its neighbors
3. If no failure then returns 'yes', else return 'no'

Part2: Show 4-COLOR is NP-Hard

Given a graph G that is 3-COLOR and a graph H that is 4-COLOR, a reduction can be done from 3-COLOR to 4-COLOR. In order to show this the graph H is identical to G except for an extra node *i* that is connected to all other nodes in H. Therefore, in order for 4-COLOR to work this node *i* has to be the only one of its given color since no adjacent node can be the same.

The removal of this node from H results in graph G which is known to be 3-COLOR. The addition of this node takes O(E+1), linear time where is the amount of edges in the graph G.

Therefore since it has been shown that 4-COLOR is both in NP and NP-Hard, 4-COLOR has to be NP-Complete.