CS325 - Homework 4

1.

‘b’ and ‘d’.

(a) There can be other algorithms too ex. brute force algorithm to solve TSP, but poor complexity.
(b) There is no efficient algorithm for NP-complete problem that can solve in P, if P is not equal to NP
(c) We cannot infer this, because if we say we can efficiently solve TSP, which means P = NP.
(d) No NP-complete problem will be P, as per assumption P is not equal to NP.
(e) The brute force approach takes n! to solve. So, It can’t be P.
(f) Not necessarily, there can be optimal cases where we can solve for some best-case inputs. Ex. All cities in horizontal axis.

2.

‘d’ and ‘g’.
We know solution for Y, and X can be reduced to Y. So, we can solve X and Y and we know that X will be as much hard (or may be less) as Y. Therefore, If Y can be solved in Polynomial time, then X can also be solved in polynomial time.

If X is NP complete and y is NP then Y is NP complete.

X reduces to Y implies if we can solve Y efficiently using black box, then we can use the black box to solve X efficiently.

3.

If we are given a certificate for the Hamiltonian path in the form of vertices that are visited, we can verify the certificate in polynomial time. This means that the HP is in NP. The verification is done by making sure that the vertices in the certificate have an edge between the consecutive vertices and each vertex is visited exactly once. This verification process takes polynomial time.

We know that the HC problem is NP-complete, we have to show, that a known NP complete problem is reducible to our problem. That is, HC $\leq_P$ HP which means we have a subroutine that could solve HP in polynomial time. How could we use it to solve HC in polynomial time?

Given a problem instance of HC, G=(V,E), construct G’ with the same vertices and edges as G and take an arbitrary vertex v from V, and split it into two vertices v1 and v2, such that these two vertices are connected to the same vertices as v and connect v1 and v2 by one edge. Finally, set s = v1 and t = v1.

As we connected v1 to v2, If G have a Hamiltonian cycle, then G’ has a Hamiltonian path from s to t. If G has a Hamiltonian cycle then it must go through vertices s, v, and t, with no other vertices in between. Deleting edges (s,v) and (t,v) and vertex v, clearly forms a Hamiltonian path from s to t in G’.

Thus, the Hamiltonian cycle problem can be reduced to Hamiltonian path problem.

4.

Long Path is NP because the path is the certificate (we can easily check in polynomial time that it is a path, and that its length is k or more).
In addition, it is NP-complete because Hamiltonian Path is a special case of Long Path, namely where k equals the number of vertices of G (n) - 1.

We showed that Hamiltonian Path problem is NP complete (Showed in Problem 3): Given an undirected graph G = (V, E), determine whether G contains a Hamiltonian path, i.e., a path that visits every vertex exactly once. It is easy to verify that Hamiltonian Path is in NP.

5.

a.

We can do this in two ways:
Check whether the graph is Bi partite or not. A bipartite is a graph whose vertices can be divided into two disjoint sets U and V (independent sets) such that every edge connects a vertex in U to one in V. Vertex set and are often denoted as partite sets.

Use BFS. Pick some start node and color it red. Color all neighbors blue. Color their neighbors red, and so on. While assigning colors, if we find a neighbor colored with same color as current vertex, then the graph cannot be colored with two colors.

b.

The decision problem is: Given an undirected graph G and an integer k, can we color G with k colors such that adjacent vertices have different colors.

1.) If we have solution to this problem in polynomial time, we can assign k values from V to 2 and check if it is colorable, and stop when k reaches some value that it is not colorable. Then we choose the minimum number from those numbers that make the graph colorable, the time is still polynomial.
2.) If Graph-coloring problem is solvable in polynomial time, then we know the minimum number of colors needed, say x, then if k < x, then we can answer NO, in decision problem; if k ≥ x, then we can answer YES in decision problem. It must be polynomial time solvable.

c.

Given a colored graph G and k, we can verify whether c includes =< k colors. Color each node of G as specified by c. For each node, check that it has a unique color from each of its neighbors. If all checks pass, accept; otherwise, reject. Given a colored coloring function c, we can find the colored graph G and k as mentioned in part b. Then, we can verify it has k colors and adjacent vertices have different colors in polynomial time. So K-Color is NP.

d.

Part 1. 4-COLOR is in NP. The coloring is the certificate (i.e., a list of nodes and colors). The following is a verifier for 4-COLOR.
Input = (G, k)
1. Check that k includes <= 4 colors.
2. Color each node of G as specified by k .
3. For each node, check that it has a unique color from each of its neighbors.
4. If all checks pass, YES; else NO.

Part 2. 4-COLOR is NP-hard. We give a polynomial-time reduction from 3-COLOR to 4-COLOR. The reduction maps a graph G into a new graph G' such that G is 3-COLOR if and only if G' is 4-COLOR. We do so by setting G' to G, and then adding a new node v' and connecting v' to each node in G'. If G is 3-colorable, then G' can be 4-colored exactly as G with v' being the only node colored with the additional color. Similarly, if G' is 4-colorable, then we know that node v' must be the only node of its color because it is connected to every other node in G'. Thus, we know that G must be 3-colorable.

Since 4-COLOR is in NP and NP-hard, we know it is NP-complete.

Submitted By:

Lakshman Madhav Kollipara