**Lab 14**

1. Show that TSP is NP-complete. (Hint: use the relationship between TSP and Hamiltonian Cycle discussed in the slides. You may assume that the Hamiltonian Cycle problem is NP-complete.)

**Solution:**

To show that TSP is NP complete, first we will show that TSP is in NP.

Suppose we are given a graph that does have Hamiltonian cycle C in G that the sum of the edges in C is at most K. To verify all these steps required Polynomial times so we can say that TSP belongs to NP.

Now we will check the reducibility of the both problems to prove that we can transform the instance of TSP to an instance of Hamiltonian cycle and solution to one yield to the solution of second.

As we know that Hamiltonian cycle is NP complete. Suppose R is a solution to NP problem so

R(poly)← HC (poly)← TSP

So, it means that R(poly)←TSP

1. True/False. Explain.
   1. If Problem A is polynomial reducible to B and A is in NP, then B is in NPH. **FALSE (IF A is polynomial reducible to B it means that B is at least as hard as B.B might be NPH but reduction doesn’t provide the information about the classes)**
   2. If Problem A is polynomial reducible to Problem B, then B is polynomial reducible to A. **FALSE (A is reducible to B means that B is at least as hard as A But in the second case where B is Polynomial reducible to A means that A is at least as hard as B which is a contradictory statement with the previous one so its false)**
   3. If someone can find a polynomial time algorithm to solve one of the NP- Complete problems, then all NP-complete problems can be solved in polynomial time. **TRUE (It’s a property of NP complete problem that if you can solve one NP complete problem and there is a relation between others so its mean that we can find the all NP complete problem)**
   4. Suppose A is an NP-complete problem and A is polynomial reducible to B. Then B is also NP-complete. **False (Being A in NP complete doesn’t mean that B also in NP complete. It might be but reduction doesn’t provide us this information)**
2. Show that the worst case for VertexCoverApprox can happen by giving an example of a graph G which has these properties:
   1. G has a smallest vertex cover of size *s*
   2. VertexCoverApprox outputs size 2\*s as its approximation to optimal size.

**Solution:**

Let’s take a simple example of two disconnected graphs.

A←B←C

D←E←F

According to the vertex cover solution it must have at least one Vertex v or w so vertex cover is equal to {(B) and (E)} which have size 2 But in VertexcoverApprox algorithm it will be {A,B,D,E} having the size of 4 which is 2\* size of Vertex cover.

1. The decision problem formulation of the Vertex Cover problem is this: Given a positive integer *k*, and a graph *G*, is there a vertex cover for *G* having size  *k*? Show that this decision problem belongs to *NP*

**Solution:**

Assuming that a solution of U vertices to a vertex cover problem. In order to prove this decision problem to be NP, we need to prove that to verify every Instance of this problem runs in Polynomial time bounded algorithm. To verify that U is correct we need to prove these things.

1. Show that U is a subset of V
2. Verify that each edge is E has an endpoint U.
3. Show that U. size is less than equal to k

As the step first requires O(n) if we use HashMap, Step 2 requires O(m) and Step 3 requires O (1) so all of these require polynomial time bounded so we can say that Vertex cover belong to NP class.