CSCI 570 - HW 8

Due Friday Nov. 02 (by 23:59)

1. Shortly explain your answer.
2. Assume . Let A and B be decision problems. If A is in NPC and A≤ p B, then B is in P.

**Sol**: False B should be in NP according to definition of polynomial time reducability.

1. If someone proves *P* = *NP*, then it would imply that every decision problem can be solved in polynomial time.

**Sol:** Ip P=NP in any case that mean all NP problems can be solved in Polynomial time.

1. If A ≤ p B and B is in *NP*-hard, then A is in *NP*-hard.

**Sol**: Yes, because A is at-least as Hard as B.

1. If A ≤ p B and B is in *NP*, then A is in *NP*.

**Sol:** No, because A is at-least as Hard as Hard as NP which means that A may be NP Hard which is not in NP.

1. Every decision problem is in NP-complete.

**Sol**: False every Decision Problem is not NP-complete as we can give many examples of some decision problems that can be solved in linear time.

1. Any NP problem can be solved in time O(2^poly(n)), where *n* is the input size and poly(*n*) is a polynomial.

**Sol:** True Because NP problems can be verified in Polynomial time but can be solved in exponential time.

1. State True/False. Assume you have an algorithm that given a 3-SAT instance, decides in polynomial time if it has a satisfying assignment. Then you can build a polynomial time algorithm that finds a satisfying assignment (if it exists) to a given 3-SAT instance. Shortly explain your answer.

**Solution:** True, Let A be a polynomial time algorithm that decides if a SAT instance is satisfiable. Let φ(x1, x2, . . . , xn) =c1∧c2∧. . .∧cm be the boolean formula corresponding to a SAT instance wherec1, c2, . . . , cm are the clauses and x1, x2, . . . , xn are the variables. If A(φ(x1, x2, . . . , xn)) = 0, then return that the instance is non-satisfiable. If A(φ(x1, x2, . . . , xn)) = 1, then we know that in the satisfying assignment, x1would be either 0 or 1. Thus, one of φ0(x2, . . . , xn) :=φ(0, x2, . . . , xn) and φ1(x2, . . . , xn) := φ(1, x2, . . . , xn) must be satisfiable. We can find out which one is satisfiable by calling A(φ0) and A(φ1). If A(φ0) = 1, we can set x1= 0 because we know there exists some satisfying assignment to φ in which x1= 0.Now we repeat the above process with φ0.On the other hand, if A(φ0) = 0, then we know that A(φ1) must be satisfiable. Thus, we set x1= 1 and repeat the process with φ1till we have found an assignment for each of the n variables. Note that as we go on fixing values of the variables to 0 or 1 one at a time, we end up with a SAT problem on one less variable each time. But the key point is that A decides any SAT instance and thus, we can keep using it on the successively smaller SAT problems. We make a total of n+ 1 calls to the polynomial time algorithm A and therefore, can find a satisfying assignment (if it exists) to a given SAT instance in polynomial time.

1. Assume that you are given a polynomial time algorithm that decides if a directed graph contains a Hamiltonian cycle. Describe a polynomial time algorithm that given a directed graph that contains a Hamiltonian cycle, lists a sequence of vertices (in order) that form a Hamiltonian cycle.

**Solution:** Let the given algorithm be HA and Let V be the set of vertices initially empty and Let G be a graph.

If HA == 1 then there is a Hamiltonian cycle which implies that it is Polynomial Time Verifiable.

Let us take a edge e and remove e from graph G and check graph G’ on HA, If it returns true continue the process by saving the vertex v in V until false is encountered. All the vertices V from the graph gives us Hamiltonian cycle. If it returns False that means we have got Hamiltonian cycle already.

1. We want to become celebrity chefs by creating a new dish. There are n ingredients and we’d like to use as many of them as possible. However, some ingredients don’t go so well with others: there is *n×n* matrix *D* giving *discord* between any two ingredients, i.e., *D*[*i,j*] is a real value between 0 and 1: 0 means *i* and *j* go perfectly well together and there is no discord and 1 means they go very badly together. Any dish prepared with these ingredients incurs a *penalty* which is the sum of the discords between all pairs of ingredients in the dish. We would like the total penalty to be small. Consider the decision problem EXPERIMENTAL CUISINE: can we prepare a dish with at least *k* ingredients and with the total penalty at most *p*? Show that EXPERIMENTAL CUISINE is NP-complete by giving a reduction from INDEPENDENT SET.

**Solution:** EXPERIMENTAL CUISINE ≤ p INDEPENDENT SET

Independent Set is the set of Vertices that are not connected. All the vertices in the graph where there is no edge between them that is the solution.  
Let us assume p = 0

Given: Independent Set

To Prove: EXPERIMENTAL CUISINE can be reduced from it

Proof: We know that Independent Set is the set of Vertices that are not connected. All the vertices in the graph where there is no edge between them that is the solution. Draw a matrix of k\*k and put 0 between the vertices that are not connected with edge and 1 otherwise.  
So at-least k ingredients that is all the vertices with no edges can be deduced from the Independent set.

Given: EXPERIMENTAL CUISINE

To Prove: Independent Set can be reduced from it

Proof: we have EXPERIMENTAL CUISINE having k ingredients , Draw a graph such that all the pair of vertices with 0 in matrix will not of edges between them while others having 1 in matrix will have edges between them so when you draw the graph you will find the independent set i.e. the vertices that are not connected.

Q.E.D

1. Longest Path is the problem of deciding whether a graph G = (V, E) has a simple path of length greater or equal to a given number *k*. Prove that the Longest path Problem is NP-complete by reduction from the Hamiltonian Path problem.

**Solution:** We need to Prove that Longest Path ≤ p Hamiltonian Path as we know that Hamiltonian Path Problem is NP Complete than we need to prove that Longest path problem is NP Complete by polynomial time reduction from Hamiltonian Path problem.

Given a solution path P, we just check that P consists of at least k edges, and that these edges form a path (where no vertex is used more than once). This verification can be done in polynomial time. The reduction follows directly from Hamiltonian Path. Given an instance of Hamiltonian Path on a graph G = (V, E). We create an instance of the longest path problem G0, k as follows. We use the same graph, i.e G0 = G and we set k = |V | − 1. Then there exists a simple path of length k in G0 iff G0 contains a Hamiltonian path.