CSCI 570 - HW8

Isaac Delgado
4189182267

1. Shortly explain your answer.

A. Assume $P \neq NP$. Let A and B be decision problems. If A is in NPC and $A \leq_p B$, then B is in P.

False; since we assume that $P \neq NP$, then B cannot be in P but must be in NP or NP-Hard.

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

False; The halting problem is a decision problem but it is not solvable regardless if $P = NP$.

C. If $A \leq_p B$ and B is in NP-hard, then A is in NP-hard.

False; "A" does not necessarily have to be NP-Hard, it only has to be in NP, and there exist some problems that are in NP but not in NP-Hard

D. If $A \leq_p B$ and B is in NP, then A is in NP.

True; B must be at least as hard at A, then this means that A must be in NP

E. Every decision problem is in NP-complete.

False; The halting problem does not belong to NP-Complete but belongs to NP-Hard.

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

True; Using a deterministic Turing machine we can solve any NP problem but it will take exponential run time

2. 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.

True; We can simply use this polynomial time algorithm given to us to iterate through our variables one by one. We assign our first variable to some value and run this algorithm to check if it has a satisfying assignment with our chosen value. If there is a satisfying assignment then we move on to the next variable and repeat this process. If there is NOT a satisfying assignment then we change our value to the opposite value (either True or False depending on what we choose originally) and then repeat this process with the next variable. We continue this process until we have gone through all variables and thus have found a satisfying assignment.

3. 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.

This is similar to the previous problem and thus we can implement a similar method. If the given polynomial time algorithm decides that we do have a Hamiltonian cycle then we can simply guess and check by removing an edge from the graph and running our given polynomial algorithm again to see if we still have a Hamiltonian cycle. If it says we do then we remove another edge otherwise we put the edge back and remove another one. Lastly once we have removed all edges from the graph except for edged in the hamiltonian cycle the we simply travers the edges printing out the vertices in order.

4. 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 $n \times n$ 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.

(1) We know that Independent Set is an NP-Complete problem

We can convert our Independent Set to an instance of our experimental cuisine problem by starting with an undirected graph G where adjacent vertices are discord ingredients. Tus an independent set in this sense would be ingredients that go well together (non-discord).

First we convert our graph G to an adjacency Matrix.
Then we can an instance of EXPERIMENTAL CUISINE on the adjacency Matrix along with additional inputs of k = # of ingredients (or the size of the Independent Set) and p = 0 the optimal penalty (an Independent Set or no adjacent vertices).
If EXPERIMENTAL CUISINE does not give us an Independent Set then we reduce the size of the expected ingredients by 1 (k-1) and try again. We repeat this until we find our Independent Set.

This shows that we can solve EXPERIMENTAL CUISINE in polynomial time and thus by reduction we can solve Independent Set in polynomial time. However, Independent Set is **NOT** solvable in polynomial (1), CONTRADICTION, therefore EXPERIMENTAL CUISINE must be in NP.

5. 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.

(1) We know that Hamiltonian path problem is NP-Complete.

Isaac Delgado
4189182267

We can reduce the Longest Path Problem by taking the length of our Hamiltonian path and running the Longest Path problem where k = *to the size of our Hamiltonian path*. Assuming Longest Path Problem is solved we can simply verify by traversing the longest path and printing our out Hamiltonian path. Thus we can solve our longest path problem in polynomial time using Hamiltonian path problem reduction. Hence we can solve Hamiltonian path problem in polynomial time, however we cannot solve Hamiltonian path in polynomial - CONTRADICTION - thus Longest Path Problem must be NP-Complete.