William Lash

A&C HW 6

1. We will show that if $SAT \in co\text{-}NP$, then $NP = co\text{-}NP$.

Step 1: show $NP \subseteq co\text{-}NP$

Since SAT is NP-complete, there exists a polynomial reduction from SAT to all languages in NP. For a language b be in co-NP, there must exist a polynomial time certificate for Turing machine M and input x such that $M(x,u) = 0$. Since SAT is NP-complete, and $SAT \in co\text{-}NP$, we can do a polynomial time reduction from its verifier to all problems in NP, so all languages in NP have a polynomial time verifier, so all of $NP \in co\text{-}NP$, or $NP \subseteq co\text{-}NP$.

FYI: Got these definitions from my 3510 notes, figured they didn't need justification

Step 2: show $co\text{-}NP \subseteq NP$:

By definition $\overline{co\text{-}NP} \in NP$. Referring back to our conclusion from step 1, $NP \subseteq co\text{-}NP$, $\overline{co\text{-}NP} \in co\text{-}NP$. Using the same definition again, $\overline{\overline{co\text{-}NP}} \in NP$. or, $co\text{-}NP \in NP$, or $co\text{-}NP \subseteq NP$.

Since we have shown given $SAT \in co\text{-}NP$, $NP \subseteq co\text{-}NP$ and $co\text{-}NP \subseteq NP$, then given $SAT \in co\text{-}NP$, $NP = co\text{-}NP$. ∎

2. If $P = NP$, there exists a polynomial time reduction, f, from any language $A \in NP$ to any language $B \in P$. Since $P = NP$, this also means there exists an f between any two languages in $P = NP$. This means there is also a reduction from any NP-complete problem to any other language in $P = NP$. Since all these languages are also in NP, both the NP-hard and NP conditions of being in NP-Complete are satisfied for all problems in P, with the exception of Σ^* and $\emptyset$.

The exception for Σ^* and $\emptyset$ exists because they cannot be reducible from an NP-complete problem. This is because when reducing from A to B, we need to map positive instances in A to positive in B, and negative in A to negative in B. In Σ^* and $\emptyset$ however, there are no negative or positive instances respectively, so they cannot be NP-hard. With these exceptions, given $P = NP$, all languages in P are NP-Complete ∎

3a. We will show that 42-SAT is NP-complete by performing a reduction from 3-SAT. Consider the clause in 3-SAT below:

$$(a \vee b \vee c) \Rightarrow (\bar{d} \vee c \vee e \vee f) \wedge (\bar{a} \vee \bar{d} \vee f \vee t) \wedge (\bar{e} \vee \bar{e} \vee \bar{e} \vee f) \wedge (t \vee t \vee t \vee f)$$
$$(\bar{b} \vee \bar{d} \vee f \vee t) \wedge (a \vee b \vee d \vee t) \wedge (\bar{f} \vee \bar{f} \vee \bar{f} \vee e)$$

We add d, e, f, t as dummy variables in ways that it does not change the satisfiability of the problem. First notice that $(\bar{e} \vee \bar{e} \vee \bar{e} \vee f)$ and $(\bar{f} \vee \bar{f} \vee f \vee e)$ force e and f to be false, we will use this later. Likewise, t must be true due to $(t \vee t \vee t \vee f)$. Next, we use d as a logical equivalent to the first two variables, in this case, $a \vee b$. Due to $(\bar{a} \vee \bar{d} \vee f \vee t) \vee (b \vee \bar{d} \vee f \vee t)$, if either a or b is true, d must become false as desired, otherwise these clauses are true. Due to $(a \vee b \vee d \vee t)$, if both a and b are false, then d must become true as desired, otherwise the clause is true. Since $\bar{d}$ has logical equivalency to the first two variables, $(\bar{d} \vee c \vee e \vee f)$ has logical equivalency to the original 3-SAT clause, and the rest of the clauses are tautologies given proper assignments of the dummy variables. Therefore, this is a valid reduction from 3-SAT to 42-SAT for a single clause. This reduction can be completed for all clauses in the original 3-SAT problem in polynomial time, meaning there is a polynomial reduction from 3-SAT to 42-SAT. Since $3\text{-SAT} \leq_p 42\text{ SAT}$, and 3-SAT is NP-complete, 42-SAT is NP-hard. It is also NP since a valid assignment of the variables can easily be checked in polynomial time by counting the number of trues in each clause and seeing if it is greater or equal to 2. Since 42-SAT is NP-hard and $\in NP$, it is NP-Complete. ∎

7b. We will show that Subgraph-Weight is NP-Complete by performing a reduction from the subset-sum problem. For some set of integers in the subset-sum problem $\{x_1, x_2, x_3, \ldots, x_n\}$, create a graph with no edges where the weight of each vertex i is x_i in the subset-sum set. Notice that finding a subgraph of weight K is equivalent to finding a subset of weight K in the subset-sum problem. Therefore, this is a valid polynomial reduction from subset-sum to subgraph-weight. Since subset-sum $\leq_p$ subgraph-weight, and subset-sum $\in$ NP-complete, subgraph-weight $\in$ NP-hard. Also, a valid subgraph can easily be checked in polynomial time by summing the values of the vertices and edges, so subgraph-weight $\in$ NP. Since subgraph-weight $\in$ NP-hard and $\in$ NP, subgraph-weight $\in$ NP-complete. ■