CSCI 270 Homework #6 Solutions

1. Write your name, student ID Number, and which lecture you attend (morning or afternoon). Multi-page submissions must be stapled.

2. Please fill out the online evaluations for this course. I'm looking for feedback and am constantly changing how the course is taught. If you have suggestions, no matter how large or small, I'm interested in seeing them.

3. In the Half-Cover problem, we are given m sets $S_1, S_2, ..., S_m$, each of which contains a subset of the integers $1, 2, ..., n$. Our goal is to determine whether there exists a collection of k sets whose union has size at least $\frac{n}{2}$.

   (a) Suppose we prove that Half-Cover is NP-complete, and that we find an $O(n^4)$ algorithm for Half-Cover. Does this imply that there is a polynomial algorithm for 3-SAT? Does this imply that there is an $O(n^4)$ algorithm for 3-SAT? Explain your reasoning.
   Solution:
   All NP-complete problems are polynomial-time reducible to each other by definition. Thus a polynomial algorithm for Half-Cover would imply a polynomial algorithm for 3-SAT. While we know there is some poly-time transformation, it does not imply that the transformation itself takes less than $O(n^4)$ time. It completely depends on the transformation, and in the absence of more information, we cannot claim there is an $O(n^4)$ algorithm for 3-SAT.

   (b) Show that Half-Cover is $\in NP$.
   Solution:
   Given a collection of sets s, one can efficiently check that the union of all sets does include half of all the elements by walking through each set in the collection. Half-Cover is in NP.

   (c) Suppose we find a polynomial-time reduction from Half-Cover to Set Cover. Would this reduction prove that Half-Cover is NP-complete? Explain your reasoning.
   Solution:
   No, as this reduction is in the wrong direction. Set Cover is NP-complete, which means all problems in NP are reducible to it. So we wasted our time with this reduction, as we already knew Half-Cover reduced to Set Cover!

   (d) Give a polynomial-time reduction from Set Cover to Half-Cover. Make sure to mention how to convert your answer for Half-Cover back into an answer for Set Cover.
   Solution:
   Given a Set-Cover instance with m sets $S_1, S_2, ..., S_m$ and a universe of elements $1, 2, ..., n$, double the number of elements to $1, 2, ..., 2n$, but do not change the sets.

   (e) Briefly justify why your reduction is correct. That is, explain why your Half-Cover reduction has a solution if and only if Set Cover has a solution.
   Solution:
   Since the elements $n + 1, ..., 2n$ will not be in any set, a solution to Half-Cover maps to a solution to Set Cover. If there is a solution to Set Cover, then the same solution must also exist for Half-Cover since it has identical sets.

4. Consider the following modification of the 2-SAT problem: we are given a 2-SAT formula with n variables and m clauses (each clause is a disjunction of exactly 2 variables or their negation), and a positive integer $k \leq m$. We want to determine if there is an assignment of true/false values to the variables such that **at least** k clauses evaluate to true. Prove that this problem is NP-complete. **Hint: you will have more success with this problem if you do not attempt to reduce from a constraint-satisfaction problem.**
   Solution:
   To prove that our problem is in NP, it takes polynomial time to verify whether there is satisfiability of at least k clauses, given a k and assignment values.

   First classify the problem. We are trying to make as many clauses true as possible subject to the constraints of the formula. Thus this sounds like a **packing** problem. We will reduce from Independent Set, but k-Clique works just fine as well.

   Create a variable x_i for each node v_i. The intention is that if we set x_i to true, then we should include v_i in the independent set. An edge between two nodes indicates that we can't set both variables to true. So for each edge $e = (v_i, v_j)$, create the clause $C_e = (\overline{x_i} \vee \overline{x_j})$, which can only be satisfied if we don't choose both endpoints in our independent set. However, we're allowed to satisfy only some clauses, so we will create $n+1$ of each of these clauses, and set k large enough to force us to satisfy these clauses.

   Finally, we want to maximize the number of variables we set to true (so that we maximize the size of the independent set), so add the clause $x_i \vee x_i$ for each node. Set $k_{2SAT} = m(n+1) + k_{IS}$, which forces us to satisfy all $m(n+1)$ clauses that encode the edges, and to set at least k_{IS} variables to true, which correlates to a solution for Independent Set.

5. In the **Bipartite** Directed Hamiltonian Cycle (BDHC) problem, we are given a bipartite directed graph $G = (V, E)$. We want to determine if there is a cycle which visits every node exactly once. Either prove that BDHC is NP-complete, or give a polynomial-time algorithm to solve it.
   Solution:
   Given a sequence of nodes, we can verify that it forms a HC by checking that the sequence includes every node, that the last node is the same as the first node, and that there is an edge between every two successive nodes in the sequence. Since this can be done in polynomial time, BDHC $\in$ NP.

   We'll reduce from DHC. Given an arbitrary graph G=(V,E) for which we have to solve DHC, modify the problem in the following way: For every node $n \in V$, create a new node n' and create an edge (n, n'). For every edge (u, v) in the original graph, change it to (u', v). This new graph is now bipartite, and has a BDHC if and only if the original graph had a DHC. We created the two graphs to be identical, we just split each node into two for the BDHC instance. If there is a hamiltonian cycle in one graph, there must be a hamiltonian cycle in the other.

   If a BDHC is found, simply remove every other node in the cycle (the nodes with a $'$), and you will have a cycle for the original graph. Thus BDHC is NP-Complete.

6. Suppose we want to solve Independent Set on a **tree**. We are given a tree $T = (V, E)$, an integer k, and we want to determine if there is an independent set of size $\geq k$. Either prove

that this problem is NP-Complete, or give a polynomial-time algorithm to solve it.
Solution:
Greedily add any leaf node to your solution. Remove the leaf node and its parent from the graph, and repeat until you have a full set. This solves the problem in polynomial time. If you are interested in the proof of correctness, it is as follows:

Suppose we choose some leaf node x first, but there is no optimal solution which includes x. Consider an arbitrary optimal solution OPT (which obviously does not include x). Since it doesn't have x, it must take x's parent y. If it did not take y, we could add x to OPT and improve the solution which is a contradiction. So OPT contains y: take y out and place x in. x is only adjacent to y, so this new solution is still valid, and still optimal. This contradicts the fact that no optimal solution includes x. Therefore, our first choice is correct.

We can then finish the proof via induction: remove x and its parent from the graph, and repeat the base case proof for the new graph. Our 2nd choice will be the 1st choice on this reduced graph, which is correct by the base case above. Repeat this for all choices, and we can see that our algorithm is optimal.