Homework 9: Complexity Theory

Problem 1. Given an undirected graph $G = (V, E)$ and a subset of its vertices V', the sub-graph induced by V' is defined as $G' = (V', E')$ where E' includes all edges from E with both endpoints in V' (that is $E' = (V' \times V') \cap E$).

A set $V'' \subseteq V$ is a *vertex cover* of G if all edges in E have at least one endpoint in V''. The *size* of a vertex cover is the number of vertices in it. We say that V'' is a *connected vertex cover* if the subgraph induced by V'' is connected.

The "k-connected vertex cover problem" (k-CONCOV) is a decision problem for which, given as input an undirected graph G and a positive integer value k, we want to decide whether there exists a connected vertex cover of G of size k or less.

(a) Present a deterministic algorithm for solving k-CONCOV. Your algorithm should run in $O\left(n^{k+2}\right)$ worst-case time, where n is the number of vertices in the graph. Argue the correctness of your algorithm and analyze its running time.

(b) Prove that k-CONCOV $\in NP$.

Solution. 1. We claim that it suffices to only consider vertex covers of size k: indeed, if there exists a vertex cover of size $k' < k$, then appending $k - k'$ vertices to it does not change the fact that it is a vertex cover, so we can always pad to make a cover of size k. With this in mind, we can brute force the problem by considering every possible subset of k vertices: we can, say, throw these k vertices into a set. Then, for each edge e, check if at least one of the endpoint lies in the current set. If so, we have a vertex cover. To check if the graph is connected, run a DFS on the induced subgraph.

This results in a runtime of $O(V) + O(E) + O(V + E) = O(V + E) \in O(n^2)$ for checking if a subset of k vertices is a connected vertex cover. The terms in order are constructing the set, checking edges have endpoints that lie in the set (checking set inclusion is constant time), and the DFS. We know this is in $O(n^2)$ since $O(E) = O(V^2) = O(n^2)$. It remains to compute the number of vertices of size k: this is precisely

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \leq \frac{n!}{(n-k)!} = n(n-1)\cdots(n-k+1) \in O(n^k)$$

The total runtime is now $O(n^{k+2})$, as desired.

2. Suppose we non-deterministically produce a solution: we wish to verify this solution in polynomial time. To do this, we can apply the exact same algorithm as in (a) for checking if a subset of k vertices is a connected vertex cover. We already showed this is $O(n^2)$, so the problem is in NP.

□

Problem 2. Consider the languages:

$$\begin{aligned}\mathsf{TFSAT}_k = \{\phi \mid \phi \text{ is a Boolean formula in } \mathsf{CNF} \text{ with k elements in each clause.}\\ \text{There exists an assignment such that in each clause,}\\ \text{there is at least one true and one false literal}\}\end{aligned}$$

Show that $TFSAT_4$ is NP-complete.

Solution. We reduce $TFSAT_4$ from 3SAT. Let ϕ be an input to 3SAT, and take any clause $C = (x_1 \vee x_2 \vee x_3)$. Consider the following transformation $f(C)$:

$$(x_1 \vee x_2 \vee x_3 \vee y)$$

Here, y is an auxiliary variable unique to $f(C)$ in the transformed boolean formula. Note that applying f to each clause in ϕ gives a boolean formula in CNF with 4 elements per clause. We claim that if ϕ is a solution to 3SAT, $f(\phi)$ is a solution to $TFSAT_4$: indeed, just take y as false, since we are guaranteed to have at least one true in x_1, x_2, x_3 (otherwise it would not satisfy 3SAT).

For the reverse direction, start with some satisfying assignment to $f(\phi)$: if y is false, then there is at least one true among x_1, x_2, x_3, so we can just use the same values for a satisfying assignment to 3SAT. If y is true, then there is at least one false among x_1, x_2, x_3. Then, $\neg x_1, \neg x_2, \neg x_3$ must have at least one true: we can then use this as our satisfying assignment to 3SAT. Having shown both directions, we get $TFSAT_4$ is NP-hard.

To show it is NP-complete, it suffices to show it is NP. Select some assignment nondeterministically: we can verify for each clause that there is at least one true or one false in linear time and then evaluate the expression in linear time as well. Therefore, we have a deterministic, polynomial time verifier, proving $TFSAT_4 \in NP$. $\square$

Problem 3. Consider the languages:

$$\mathsf{EX1SAT} = \{\phi \mid \phi \text{ is a satisfiable Boolean formula in } \mathsf{CNF} \text{ with exactly 1 literal true per clause}\}$$

Show that $EX1SAT$ is NP-complete.

Solution. We want to first reduce 3SAT to EX1SAT. Let ϕ be an input to 3SAT. Let us take any arbitrary clause in ϕ, say containing $(x_1 \vee x_2 \vee x_3)$. Then we create 6 new sets of variables with the following expression: $(a_1 \vee a_2 \vee a_3) \wedge (\neg x_1 \vee a_1 \vee b_1) \wedge (\neg x_2 \vee a_2 \vee b_2) \wedge (\neg x_3 \vee a_3 \vee b_3)$. Here, a_i and b_i are variables not used elsewhere in ϕ. We do this for every clause, each time using 6 new variables to make ϕ'. We want to show that $\phi \in 3SAT \implies \phi' \in EX1SAT$. If $\phi \in 3SAT$, at least one of x1, x2, and x3 were true. WLOG assume x1 was true. Then, there are 3 cases for the other 2 variables. In case 1, both x2 and x3 are false. Then we set $a_1 = T$, and the other 5 variables as F. In case 2, exactly one of x2 and x3 is true. WLOG assume x2 was true. Then we set $a_1, b_2 = T$ and the other 4 variables as F. In case 3 all 3 variables are T and we set $a_1, b_2, b_3 = T$ and the other 3 variables as F. We see by plugging these values in that in each case, exactly one literal from each clause is T. Thus, $\phi' \in EX1SAT$.

Now we want to show $\phi' \in EX1SAT \implies \phi \in 3SAT$. For each set of 4 clauses, we identify which of a_1 to a_3 are T. WLOG assume a_1 is T. But then x_1 has to be T to avoid two literals being true in $(\neg x_1 \vee a_1 \vee b_1)$, making the $(x_1 \vee x_2 \vee x_3)$ clause true in ϕ. Thus, we are able to construct a satisfying assignment for 3SAT. This means we have successfully reduced 3SAT to EX1SAT in polynomial time.

Lastly, we need to show EX1SAT is in NP. We can nondeterminstically choose a truth assignment for the variables, and accept if one literal is true per clause. This takes polynomial time to check, so EX1 is in NP. Since it is reducable from 3SAT it is also NP-Hard, making it NP-Complete. □