Homework 9: Complexity Theory

Due: November 17, 2025

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.

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

Problem 2. Let BF_k denote the set of Boolean formulas in Conjunctive Normal Form such that each variable appears in at most k places (i.e., in at most k literals). Show that the problem of deciding whether a Boolean Formula in BF_3 is satisfiable is $NP-Complete$. [Hint: You can replace a variable with several variable, adding a to the formula the condition these variables must have the same value.]

Problem 3. Recall that a *literal* in a Boolean formula is either a Boolean variable (e.g., x_i) or its negated form (e.g., $\neg x_i$) appearing in the formula.

Let ϕ be a 3CNF formula. A $\neq$-assignment for the variables of ϕ is one in which each clause contains at least two *literals* with unequal truth values. In other words, a given clause cannot be assigned all true or all false literals in a $\neq$-assignment. For example, $(x_1, x_2, x_3) = (T, T, F)$ is a $\neq$-assignment for the following Boolean formula but $(x_1, x_2, x_3) = (F, T, F)$ is not:

$$(\neg x_1 \vee x_2 \vee x_2) \wedge (x_2 \vee x_2 \vee x_3)$$

1. Show that the negation of any $\neq$-assignment to ϕ is also a $\neq$-assignment of ϕ.
2. Let $\neq$–SAT denote the problem of deciding whether a Boolean formula has a $\neq$-assignment. Show that the following is a valid polynomial time reduction from 3SAT to $\neq$–SAT:

   (a) Given an input ϕ check its format.

(b) If $f(\phi)$ is not in $\mathsf{3CNF}$ then return

$$f(\phi) = u,$$

where u is a Boolean variable that does not appear in ϕ.

(c) If ϕ is in $\mathsf{3CNF}$ format, $f(\phi)$ is a Boolean expression where we add to each of ϕ's clauses an additional literal u, where u is a new Boolean variable that did not appear in ϕ

For example, consider ϕ and $f(\phi)$ below:

$$\phi := (x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee x_1 \vee x_3)$$
$$f(\phi) = (x_1 \vee x_2 \vee x_3 \vee u) \wedge (x_4 \vee x_1 \vee x_3 \vee u)$$

3. Conclude that $\neq$-SAT is NP-complete.