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 $\in NP$.

Solution.

- Algorithm: We enumerate all subsets $V' \subseteq V$ of size at most k. For each such V' we check:

  1. Whether V' is a vertex cover, i.e., for every $(u, v) \in E$, at least one of u or v lies in V'.
  2. Whether the induced subgraph $G[V']$ is connected.

  If a subset passes both checks, we accept.

  Correctness:

  - If the algorithm accepts, then some enumerated set V' has size at most k, covers all edges, and induces a connected subgraph. By definition, V' is a connected vertex cover of size $\leq k$.
  - If a connected vertex cover V'' of size at most k exists, then the algorithm will enumerate V'' and accept when checking it.

  Runtime Analysis:

  - There are at most
$$\sum_{i=0}^{k} \binom{n}{i} = O(n^k)$$
    subsets of size at most k.
  - Checking the vertex cover property requires inspecting all edges, which takes $O(m) \subseteq O(n^2)$ in worst case.

- Checking connectivity of $G[V']$ with BFS/DFS takes $O(|V'| + |E'|) \subseteq O(n^2)$.

Thus each subset is processed in $O(n^2)$ time, giving a total running time:

$$O(n^k) \cdot O(n^2) = O(n^{k+2}).$$

- k-CONCOV is in NP: A valid certificate consists of a set V' of vertices with $|V'| \leq k$. A polynomial-time verifier:

  1. Checks that $|V'| \leq k$.
  2. Checks the vertex cover property in $O(n^2)$ time.
  3. Checks whether $G[V']$ is connected in $O(n^2)$ time.

  The verifier accepts a certificate c if and only if the graph G has a connected vertex cover of size at most k. All checks run in polynomial time, so k-CONCOV $\in \mathsf{NP}$.

$\square$

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.]

Solution.

- BF_3-SAT $\in$ NP: A certificate is a truth assignment to all variables in the formula. Evaluating the formula takes time linear in its size, so the problem is in NP.
- BF_3-SAT is NP-hard: We reduce from 3SAT. Given a 3CNF formula ϕ, variables may appear more than three times. For each variable p that appears $n > 3$ times, we:

  1. Replace each occurrence of p by a fresh variable $p_1, \ldots, p_n$.
  2. Add the cycle of clauses enforcing equivalence:

  $$(\neg p_1 \vee p_2) \wedge (\neg p_2 \vee p_3) \wedge \cdots \wedge (\neg p_n \vee p_1).$$

  Each p_i appears at most three times: once replacing p in its clause, and twice in the equivalence cycle clauses. Call the resulting formula $f(\phi)$.

  Correctness:

  – If ϕ is satisfiable, extend a satisfying assignment by setting all p_i to the original value of p. All equivalence clauses and all original clauses (now rewritten) are satisfied.

  – If $f(\phi)$ is satisfiable, the cycle clauses force $p_1 = \cdots = p_n$. Setting p to this common value satisfies ϕ.

  Runtime Analysis: For each variable occurring n times, we introduce n new variables and $O(n)$ new clauses. Summed over all variables, this yields a polynomial-size formula.

  Thus 3SAT $\leq_p BF_3$ and BF_3-SAT is NP-complete.

□

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 3CNF then return
   $$f(\phi) = u,$$
   where u is a Boolean variable that does not appear in ϕ.
   (c) If ϕ is in 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.

Solution.

1. The complement of a $\neq$-assignment preserves the property: Let α be a $\neq$-assignment. In every clause of ϕ, there exist literals ℓ_i and ℓ_j with $\ell_i(\alpha) \neq \ell_j(\alpha)$. Under the complement assignment $\bar{\alpha}$ we have
   $$\ell(\bar{\alpha}) = \neg \ell(\alpha).$$
   Thus,
   $$\ell_i(\alpha) \neq \ell_j(\alpha) \quad \Rightarrow \quad \ell_i(\bar{\alpha}) \neq \ell_j(\bar{\alpha}),$$
   so $\bar{\alpha}$ is still a $\neq$-assignment.
2. Reduction: Given a 3CNF formula
   $$\phi = (C_1) \wedge \cdots \wedge (C_m),$$
   form $f(\phi)$ by appending a new literal u to each clause:
   $$f(\phi) = (C_1 \vee u) \wedge \cdots \wedge (C_m \vee u),$$

where u does not appear in ϕ.

Correctness:

ϕ satisfiable $\Rightarrow f(\phi)$ has a $\neq$-assignment.

Extend a satisfying assignment for ϕ by setting $u = F$. Every clause already has a true literal from ϕ, and u is false. Hence each clause contains at least one true and one false literal, so it is a $\neq$-assignment.

$f(\phi)$ has a $\neq$-assignment $\Rightarrow \phi$ satisfiable.

Let α be a $\neq$-assignment for $f(\phi)$.

- If $\alpha(u) = F$, then in each clause $(C_i \vee u)$, not all literals can be false. Thus at least one literal of C_i is true, so ϕ is satisfiable.
- If $\alpha(u) = T$, then consider the complement assignment $\bar{\alpha}$. By part (1), $\bar{\alpha}$ is also a $\neq$-assignment, and now $\bar{\alpha}(u) = F$. Using the previous case, ϕ is satisfiable.

Thus

$$\phi \text{ satisfiable} \iff f(\phi) \text{ has a } \neq \text{-assignment.}$$

3. $\neq$-SAT is NP-complete:

   - Membership in NP: Given an assignment, we can verify in polynomial time that each clause has at least two literals evaluating differently.
   - NP-hardness: The reduction above shows 3SAT $\leq_p \neq$ -SAT (with the proof of correctness). The reduction is also completed in polynomial time because we are just doing a linear (in the input size) scan, adding the variable u to each clause.

Thus $\neq$-SAT is NP-complete.

□