CS 361: Theory of Computation

Assignment 9 (due 11/20/2025)

Instructor: Shikha Singh

LaTeX Source for Solutions: https://www.overleaf.com/read/gvvvtnfhcxzw#51658b

Partner Work. You may complete this assignment in pairs. Both you and your partner can solve the problems together and submit a single write up on Gradescope. Please make sure to add your partners name to the submission when submitting.

Problem 1. (Understanding Polynomial-time Reductions).

(a) Using an example of Graph-2-Color and Graph-3-Color, prove that the following statement is **False**.

If $A \leq_p B$ and A can be decided in polynomial-time, then B can also be decided in polynomial-time.

In particular, show that (i) Graph-2-Color $\leq_p$ Graph-3-Color and (ii) Graph-2-Color $\in$ P.

Solution.

$\square$

(b) A natural question to ask is there can exist problems in NP that are *known* to be **not NP-complete**. Prove that finding such problems is the same as solving the P versus NP question. In particular, prove that if there exists a non-trivial language $L \in \mathsf{NP}$ such that L is not NP-complete, then $\mathsf{P} \neq \mathsf{NP}$.

Equivalently, prove that if $\mathsf{P} = \mathsf{NP}$, then every non-trivial language $L \in \mathsf{P}$ is NP complete. A language L is **trivial** if $L = \emptyset$ or $L = \Sigma^*$.

Solution.

$\square$

A Note of NP-completeness proofs. The next three questions will ask you to prove that a given language X is NP Complete. A complete solution would include the following:

- Language X is in NP: give a polynomial-time NTM or a polynomial-size certificate and polynomial-time deterministic verifier.
- State a *known* NP hard problem Y from class that you will use to prove X is also NP hard
- Show that $Y \leq_p X$. Remember to:
  - Prove that the reduction is correct by arguing both the "if" and "only if" directions
  - Argue that your reduction is computable in polynomial time (a brief justification is sufficient for this)

Solved Example. Here is a solved example to demonstrate how you should structure your proof. In an undirected graph G, a **feedback-set** F is a subset of vertices that contains at least one vertex from every directed cycle in G, that is, $G \setminus F$ is acyclic. Prove that the problem of finding a feedback-set of size at most k is NP complete.

Solution. First, we prove that FEEDBACK-SET $\in$ NP. Consider the non-deterministic TM N for FEEDBACK-SET below that runs in time linear in the size of G.

```
N = "On input $\langle G, k \rangle$:
    1. Non-deterministically guess a subset $F$ of $V$.
    2. Verify if $F$ is a valid subset of $V$ and contains at most $k$ vertices. If not, reject.
    3. Check if $F$ forms a feedback vertex set by checking if $G \setminus F$ is acyclic using DFS.
       If it is, accept. Otherwise, reject."
```

We now give a reduction from minimum vertex-cover to Feedback-Set. In the minimum vertex-cover problem, given a graph $\langle G, k \rangle$, the problem is to determine if G has a vertex cover of size at most k.

Given an instance $\langle G, k \rangle$ of vertex cover, we create an instance $\langle G', k' \rangle$ of Feedback-Set. Let $k' = k$. For each edge $e = (u, v) \in G$, create a new vertex x_e and add the edges $(u, x_e), (v, x_e)$ and (u, v) in G'. Thus, G' has vertices $V \cup \{x_e \mid e \in E(G)\}$ and edges $\cup_{e \in E(G)} \{e, (u, x_e), (v, x_e)\}$. Note that his mapping from G to G' takes time linear in the size of G. Thus, this is a poly-time reduction.

Correctness proof. Now we prove that the reduction is correct—G has a vertex cover of size at most k iff G' has a feedback-set of size at most k.

($\Rightarrow$) Suppose G has a vertex cover C of size at most k. Then C is a feedback-set of G'. Consider any edge $e = (u, v)$ in G, since C is a vertex cover, it must contain either u or v. Removing either u or v, destroys all cycles in G' that uses the edge e. Thus, $G - C$ is acylic, and C is a feedback-set of size at most k.

($\Leftarrow$) Suppose G' has a feedback-set F of size at most k. If F contains any "new" vertex x_e for an edge $e = (u, v)$, replace x_e with v. Note that this does increase the size of F and F remains a feedback-arc set since the any cycle containing the vertex x_e also contains u and v. Now we have a feedback set F of G' such that $F \subseteq V(G)$ and size of F is at most k. We claim that F is a vertex cover of G. Since F must contain at least one vertex for each cycle $C_3 = \{(u, v), (v, x_e), (x_e, u)\}$ and this vertex cannot be x_e, this implies that F contains u or v for each edge (u, v), i.e., F is a vertex-cover of G. □

Known-NP-Hard Problems. For the purpose of this assignment as well as the exam, you may assume any of the following problems are NP hard and you may use them in your reduction.

(a) Independent Set
(b) Vertex Cover
(c) Set Cover
(d) Clique
(e) SAT/3-SAT
(f) Hamiltonian Cycle
(g) Hamiltonian Path
(h) Graph 3-color
(i) Subset-Sum

Problem 2. A double-Hamiltonian tour in an undirected graph G is a closed walk that visits every vertex in G exactly twice. We want to prove that the problem of deciding whether a given graph G has a double-Hamiltonian tour is NP hard.

(a) Consider the following incorrect reduction from Hamiltonian Cycle: Given a graph $G = (V, E)$ that is an input to the Hamiltonian Cycle problem, create a new graph $G' = (V, E')$ where $E' = E \cup \{(v, v)\}$. That is G' is the same as G with self-loops inserted to each node. This reduction maps "yes" instances of Hamiltonian Cycle to "yes" instances of double-Hamiltonian tour. Show that this reduction is still incorrect as it fails to map "no" instances of Hamiltonian Cycle to "no" instances of double-Hamiltonian tour by giving a counter-example.

Solution. □

(b) Show that double-Hamiltonian tour is NP hard by giving a correct reduction from Hamiltonian cycle.

Solution. □

Problem 3. (Problem 7.26 from Sipser) Let ϕ be a 3-CNF formula. An $\neq$-assignment to the variables of ϕ is one where each clause contains two literals with unequal truth values. In other words, an $\neq$-assignment satisfies ϕ without assigning three true literals in any clause.

(a) Show that the negation of any $\neq$-assignment to the variables of ϕ is also an $\neq$-assignment.

Solution.

□

(b) Let $\neq$SAT be the set of 3-cnf formulas that have an $\neq$-assignment. Show that we obtain a polynomial-time reduction from 3SAT to $\neq$SAT by replacing each clause $c_i = (y_1 \vee y_2 \vee y_3)$ with two clauses

$$(y_1 \vee y_2 \vee z_i) \text{ and } (\overline{z_i} \vee y_3 \vee b),$$

where z_i is a new variable for each clause c_i and b is a single additional new variable. Prove this reduction is correct. (*Hint.* Use part (a).)

Solution.

□

(c) Finally, show that $\neq$SAT is in NP and conclude that it is NP complete.

Solution.

□

Problem 4. (Problem 7.27 from Sipser) A **cut** in an undirected graph is a separation of the vertices V into two disjoint subsets S and T. The size of a cut is the number of edges that have one end point in S and the other in T. Let

$$\mathsf{MAXCUT} = \{\langle G, k\rangle \mid G \text{ has a cut of size at least } k\}.$$

In this problem, we will prove that MAXCUT is NP hard by showing that $\neq\mathsf{SAT} \leq_p \mathsf{MAXCUT}$.

Reduction hint. The variable gadget for the variable x is a collection of $3c$ nodes labeled x and another $3c$ nodes labeled $\overline{x}$, where c is the number of clauses. All nodes labeled x are connected with all nodes labeled $\overline{x}$. The clause gadget is a triangle of three edges connecting three nodes labeled with the literals appearing the clause. Do not use the same node in more than one clause gadget.

(a) Consider an arbitrary instance ϕ of $\neq$-SAT with m variables and c clauses. How many vertices and edges are in the graph G constructed in the hint above?

(b) Prove that the above reduction is correct. Show that ϕ has a $\neq$-assignment if and only if G has a cut of size at least $9c^2 \cdot m + 2c$.

Solution.

□