CS 361: Theory of Computation

Assignment 8 (due 11/20/2024)

Instructor: Shikha Singh

LaTeX Source for Solutions: https://www.overleaf.com/read/vvjbpbqrmksv#2add1a

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.

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 Graph-2-Color $\leq_p$ Graph-3-Color and that Graph-2-Color $\in$ P.

(b) In lecture, some of you asked if 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 $L \in$ NP such that L is not NP-complete, then P $\neq$ NP.

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

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)

For the purpose of this assignment as well as the exam, you can use any of the problems whose NP-hardness was established in class or Chapter 7 of Sipser. Chapter 12 in Erickson's textbook is also a good reference for NP hard problems and their proofs. Here is a (not necessarily complete) list of problems you may assume are NP hard for the purpose of reducitons:

(a) Independent Set
(b) Vertex Cover
(c) Set Cover
(d) Clique
(e) Circuit-SAT/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.

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

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.

(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.

(c) Conclude that $\neq$SAT is NP-complete.

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

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

Show that MAXCUT is NP-complete. You may assume the result of Problem 3.

Hint: Show that $\neq\mathsf{SAT} \leq_p \mathsf{MAXCUT}$. 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. Draw a picture for an example ϕ to visualize this graph. Fill in the gaps in this reduction and prove that it works.