Homework 8

Due 8:00pm, March 27
(accepted until 9:59 pm, no credit after)

This homework has 8 questions, for a total of 100 points and 8 extra-credit points.
Unless otherwise stated, each question requires *clear*, *logically correct*, and *sufficient* justification to convince the reader.
For bonus/extra-credit questions, we will provide very limited guidance in office hours and on Piazza, and we do not guarantee anything about the difficulty of these questions.
We strongly encourage you to typeset your solutions in LaTeX.
If you collaborated with someone, you must state their name(s). You must *write your own solution* for all problems and *may not use any other student's write-up*.

(0 pts) 0. **Before you start; before you submit.**

- Read Handout 3: NP-Hardness Proofs to understand the components of such proofs.
- If applicable, state the name(s) and uniqname(s) of your collaborator(s).

Solution:

(10 pts) 1. **Self assessment.**

Carefully read and understand the posted solutions to the previous homework. Identify one part for which your own solution has the most room for improvement (e.g., has unsound reasoning, doesn't show what was required, could be significantly clearer or better organized, etc.). Copy or screenshot this solution, then in a few sentences, explain what was deficient and how it could be fixed.

(Alternatively, if you think one of your solutions is significantly *better* than the posted one, copy it here and explain why you think it is better.)

If you didn't turn in the previous homework, then (1) state that you didn't turn it in, and (2) pick a problem that you think is particularly challenging from the previous homework, and explain the answer in your own words. You may reference the answer key, but your answer should be in your own words.

Solution:

2. **Review of last week's materials.**

(4 pts) (a) Define the terms P, NP, NP-Hard, and NP-Complete. (You do not need to define the terms that are used in these definitions.)

Solution:

(8 pts) (b) For each of the following statements, state, with brief justification, that it is *known to be true*, *known to be false*, or its truth/falsehood is *unknown.* Note: as always, if a statement has any counterexample(s), it is false.

(i) Let V be an efficient verifier for $L \in \mathsf{NP}$ and x, c be strings; if $V(x, c)$ accepts, then $x \in L$.

(ii) Let V be an efficient verifier for $L \in \mathsf{NP}$ and x, c be strings; if $V(x, c)$ rejects, then $x \notin L$.

(iii) If $\mathsf{P} \neq \mathsf{NP}$ and L is NP-Complete, then $L \notin \mathsf{P}$.

(iv) If L is NP-Hard, then $L \notin \mathsf{P}$.

Solution:

(4 pts) (c) Briefly explain what the "Boolean satisfiability" (SAT) problem is, and give one example instance that is satisfiable, and one that is unsatisfiable. Each example must include at least 2 variables and at least 3 (not necessarily distinct) logical operators (AND/OR/NOT).

Solution:

(4 pts) (d) Towards proving the Cook-Levin Theorem in lecture, we outlined a proof of the following statement:

Let language $L \in \mathsf{NP}$ by arbitrary, and fix an efficient verifier $\textsc{VerifyL}$ for L. Given any instance x of L, in polynomial time we can construct an instance ϕ of SAT such that ϕ is satisfiable *if and only if* there *exists* a c such that the tableau of $\textsc{VerifyL}(x, c)$ has q_{accept} in it.

Briefly explain why proving this statement proves that SAT is NP-Hard.

Solution:

(10 pts) 3. **Turing reductions vs. poly-time mapping reductions.**

Suppose that $\mathsf{P} \neq \mathsf{NP}$, and let $A \in \mathsf{P}$ be arbitrary. Prove that there is a *Turing* reduction from SAT to A (i.e., SAT $\leq_T A$), but there is *no polynomial-time mapping reduction* from SAT to A. (i.e., SAT $\not\leq_p A$).

Solution:

4. **Subset sum.**

   Consider the following reduction involving 3SAT and the language

   $$\textsc{SubsetSum} = \{(A, s) : A \text{ is a } \textit{multiset} \text{ of integers} \geq 0, \text{ and } \exists\, I \subseteq A \text{ such that } \sum_{a \in I} a = s\}.$$

   A *multiset* A is just like a set, but it can have duplicate elements. A submultiset $I \subseteq A$ also can have duplicate elements, as long as it does not have more copies of an particular element than A does.[1]

   The reduction is given a 3CNF formula φ with n variables $x_1, \ldots, x_n$ and m clauses $c_1, \ldots, c_m$. We assume without loss of generality that there is no repeated clause. (Recall that each clause is the OR of exactly three literals, where a literal is either some variable x_i or its negation $\overline{x_i}$.)

   The reduction constructs a collection of numbers as follows:

   1. For each clause c_j, it constructs integers a_j and b_j, each $n + m$ decimal digits long, where the jth digit of both a_j and b_j is 1, and all other digits are 0.
   2. For each variable x_i, it constructs integers t_i and f_i, each $n+m$ decimal digits long, where:
      (a) The $(m + i)$th digits of both t_i and f_i are 1.
      (b) The jth digit of t_i is 1 if literal x_i appears in clause c_j.
      (c) The jth digit of f_i is 1 if literal $\overline{x_i}$ appears in clause c_j.
      (d) All other digits of t_i and f_i are 0.
   3. It constructs a target sum s that has $n + m$ decimal digits, where the first m digits are 3 and the last n digits are 1.

   The reduction outputs (A, s), where A is the multiset of all the numbers a_j, b_j, t_i, and f_i.

   For example, for the formula $\varphi = (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3})$, the reduction constructs:

Number	$j = 1$	$j = 2$	$i = 1$	$i = 2$	$i = 3$
a_1	1	0	0	0	0
b_1	1	0	0	0	0
a_2	0	1	0	0	0
b_2	0	1	0	0	0
t_1	1	0	1	0	0
f_1	0	1	1	0	0
t_2	0	1	0	1	0
f_2	1	0	0	1	0
t_3	1	0	0	0	1
f_3	0	1	0	0	1
s	3	3	1	1	1

[1] In class we defined a slightly different version of SubsetSum with ordinary sets instead of multisets. With a little more work, the version with ordinary sets can also be proved NP-Complete.

Answer the following:

(2 pts) (a) Fill in the blanks: the above reduction is for showing that ________ $\leq_p$ ________.

Solution:

(15 pts) (b) Prove that the reduction is correct by proving the following, for any 3CNF formula φ:

- $\varphi \in \text{3SAT} \implies (A, s) \in \textsc{SubsetSum}$;
- $(A, s) \in \textsc{SubsetSum} \implies \varphi \in \text{3SAT}$.

Solution:

(5 pts) (c) Why do we set the first m digits of s to be 3? Would the reduction still be correct if we used 2 instead? What about 4?

Solution:

(18 pts) 5. **Tours and cycles.**

Recall the following definitions (in this problem, all graphs are undirected):

$$\textsc{HamCycle} = \{G : G \text{ is an unweighted graph with a Hamiltonian cycle}\}$$
$$\text{TSP} = \{(G, k) : G \text{ is a weighted, complete graph with a tour of weight} \leq k\} .$$

Prove that $\textsc{HamCycle} \leq_p \text{TSP}$. (Recall from lecture and the handout what such a proof involves.) Since $\textsc{HamCycle}$ is NP-Hard, is follows that TSP is also NP-Hard.

Reminder: an instance of $\textsc{HamCycle}$ is an arbitrary unweighted graph, while an instance of TSP is a *complete* graph where each edge has a weight.

Solution:

6. **More Knapsack!**

Recall the 0-1 knapsack problem: an instance is a list of n item weights $W = (W_1, W_2, \ldots, W_n)$, their corresponding values $V = (V_1, V_2, \ldots, V_n)$, and a weight capacity C. (All values are non-negative integers.) The goal is to select items having maximum total value, such that their total weight does not exceed the capacity. We can make this a decision problem by introducing a "budget" K and asking whether a total value of at least K can be achieved (again, subject to the capacity constraint):

$$\textsc{Knapsack} = \{(W, V, C, K) : \exists\, S \subseteq \{1, \ldots, n\} \text{ such that } \sum_{i \in S} W_i \leq C \text{ and } \sum_{i \in S} V_i \geq K\}.$$

(14 pts) (a) Prove that KNAPSACK is NP-Hard, by showing that SUBSETSUM $\leq_p$ KNAPSACK. (See Question 4 for the definition of SUBSETSUM.)

Solution:

(6 pts) (b) Recall that we previously gave a dynamic programming algorithm that solves KNAPSACK in $O(nC)$ time. Does this prove that $\mathsf{P} = \mathsf{NP}$? Why or why not?

Solution:

(8 EC pts) 7. **Optional extra-credit question: Network reliability.**

The Republic of Alvonia owns the internal internet infrastructure for the country and leases out connections to Internet Service Providers (ISPs). We represent that network as an undirected graph. Assume that each edge in the graph has a positive integer *rental cost* associated with it. An ISP is confronted with the problem of spending the minimum amount of money on link rental so that it can provide adequately reliable service to its customers.

Here is how we quantify reliability: We say that two paths in the network are *disjoint* if they have no vertices in common, except for possibly their endpoints. For example, there can be two disjoint paths from vertex v_{42} to v_{100}, but v_{42} and v_{100} can be the only vertices that these paths have in common (since these vertices are the endpoints of the paths). In the interest of reliability, it is desirable to have multiple disjoint paths between pairs of nodes in the network. Some ISPs provide more reliability than others, but they may charge their clients more.

The Network Reliability Optimization Problem (NROP) is now defined as follows. An instance is an undirected graph with n vertices $v_1, \ldots, v_n$, a non-negative integer weight on each edge, and an n-by-n symmetric matrix R_{ij}. The objective is to find a subset S of the edges such that the total cost of the edges in S is minimized, with the requirement that for every pair of vertices v_i and v_j, there are at least R_{ij} disjoint paths from v_i to v_j such that all paths use only edges in S.

You've been hired as a summer intern to develop an efficient algorithm for solving NROP. After working on an algorithm for awhile, your team conjectures that the problem is NP-hard. Here is your task:

1. Define the decision version of this problem, which we will call NRDP;
2. Prove that NRDP is NP-hard via a reduction from an NP-hard problem from lecture.

Solution: