CS 341: Foundations of Computer Science II
Prof. Marvin Nakayama

Homework 13 Solutions

1. The *Set Partition Problem* takes as input a finite set S of numbers. The question is whether the numbers can be partitioned into two sets A and $\overline{A} = S - A$ such that

$$\sum_{x \in A} x = \sum_{x \in \overline{A}} x.$$

Show that *SET-PARTITION* is NP-Complete. (Hint: Reduce *SUBSET-SUM*, which Theorem 7.56 of the Sipser textbook shows is NP-complete.)

Answer: To show that any problem A is NP-Complete, we need to show four things:
(1) there is a non-deterministic polynomial-time algorithm that solves A, i.e., $A \in$ NP,
(2) some NP-Complete problem B can be reduced to A,
(3) the reduction of B to A works in polynomial time,
(4) the original problem A has a solution if and only if B has a solution.

We now show that *SET-PARTITION* is NP-Complete.

(1) *SET-PARTITION* $\in$ NP: Guess the two partitions and verify that the two have equal sums.

(2) Reduction of *SUBSET-SUM* to *SET-PARTITION*: Recall *SUBSET-SUM* is defined as follows: Given a set X of integers and a target number t, find a subset $Y \subseteq X$ such that the members of Y add up to exactly t. Let s be the sum of members of X. Feed $X' = X \cup \{s - 2t\}$ into *SET-PARTITION*. Accept if and only if *SET-PARTITION* accepts.

(3) This reduction clearly works in polynomial time.

(4) We will prove that $\langle X, t \rangle \in$ *SUBSET-SUM* iff $\langle X' \rangle \in$ *SET-PARTITION*. Note that the sum of members of X' is $2s - 2t$.

$\Rightarrow$: If there exists a set of numbers in X that sum to t, then the remaining numbers in X sum to $s - t$. Therefore, there exists a partition of X' into two such that each partition sums to $s - t$.

$\Leftarrow$: Let's say that there exists a partition of X' into two sets such that the sum over each set is $s - t$. One of these sets contains the number $s - 2t$. Removing this number, we get a set of numbers whose sum is t, and all of these numbers are in X.

2. Let

$$DOUBLE\text{-}SAT = \{ \langle \phi \rangle \mid \phi \text{ is a Boolean formula with two satisfying assignments} \}.$$

Show that *DOUBLE-SAT* is NP-Complete. (Hint: Show that *DOUBLE-SAT* is NP-Hard by reducing *3SAT* to *DOUBLE-SAT*.)

Answer:
(1) *DOUBLE-SAT* $\in$ NP: Simply guess two different assignments to all variables and verify that each clause is satisfied in both cases.

(2) Reduction of *3SAT* to *DOUBLE-SAT*: Given a 3cnf-function ψ, create a new Boolean function ψ' by adding a new clause $(x \cup \overline{x})$ to ψ, where x is a new variable not in ψ. Then check if $\langle \psi' \rangle \in$ *DOUBLE-SAT*.

(3) This reduction clearly works in polynomial time.

(4) We now prove that the original 3cnf-function $\langle \psi \rangle \in$ *3SAT* iff the new Boolean function $\langle \psi' \rangle \in$ *DOUBLE-SAT*. If the original 3cnf-function ψ is unsatisfiable, then the new function ψ' is also unsatisfiable; i.e., $\langle \psi \rangle \notin$ *3SAT* implies $\langle \psi' \rangle \notin$ *DOUBLE-SAT*. If $\langle \psi \rangle \in$ *3SAT*, then use the same assignment of variables that are in ψ, and we also have both $x = 0$ and $x = 1$ are valid assignments. Thus, there are at least two satisfying assignments of the augmented 3cnf-formula ψ', so $\langle \psi' \rangle \in$ *DOUBLE-SAT*.

3. Let G represent an undirected graph. Also let

$$SPATH = \{ \langle G, a, b, k \rangle \mid G \text{ contains a simple path of length at most } k \text{ from } a \text{ to } b \}$$

and

$$LPATH = \{ \langle G, a, b, k \rangle \mid G \text{ contains a simple path of length at least } k \text{ from } a \text{ to } b \},$$

where a simple path is a path that does not repeat any nodes, and the length of a path is the number of edges along the path.

(a) Show that *SPATH* $\in$ P.

Answer:
The marking algorithm in Theorem 7.14 for deciding *PATH* in polynomial time can be modified to keep track of the length of the shortest paths discovered. Here is a detailed description of the algorithm.

"On input $\langle G, a, b, k \rangle$ where m-node graph G has nodes a and b:

1. Place a mark "0" on node a.

2. For each i from 0 to m:

3. If an edge (s, t) is found connecting s marked "i" to an unmarked node t, mark node t with "$i + 1$".

4. If b is marked with a value of at most k, *accept*. Otherwise, *reject*.

(b) Show that *LPATH* is NP-Complete. You may assume the NP-completeness of *UHAMPATH*, the Hamiltonian path problem for undirected graphs.

Answer:
First, *LPATH* $\in$ NP because we can guess a simple path of length at least k from a to b and verify it in polynomial time. Next *UHAMPATH* $\leq_{\mathrm{P}}$ $LPATH$, because the following TM F computes the reduction f.

F = "On input $\langle G, a, b \rangle$ where graph G has nodes a and b:

1. Let k be the number of nodes of G.

2. Output $\langle G, a, b, k \rangle$.

If $\langle G, a, b \rangle \in$ *UHAMPATH*, then G contains a Hamiltonian path of length k from a to b, so $\langle G, a, b, k \rangle \in$ *LPATH*. If $\langle G, a, b, k \rangle \in$ *LPATH*, then G contains a simple path of length k from a to b. But G has only k nodes, so the path is Hamiltonian. Thus, $\langle G, a, b \rangle \in$ *UHAMPATH*.