COMP 360 - Fall 2012 - Assignment 4
Solutions

December 7, 2012

1. (a) First we argue that X is in NP. As a certificate (proof), we take a cycle of length k: $(v_1, v_2, \ldots, v_k)$. The verifier checks that the list does not contain any repeated vertices and that $(v_i, v_{i+1}) \in E$ for $i \in \{1, 2, \ldots, k-1\}$ and $(v_k, v_1) \in E$. This can be easily done in polynomial time.

   Now we argue that X is NP-hard. We reduce HAM-CYCLE to X. The reduction is very straightforward. We show that if there is a polynomial time algorithm for X then we can solve HAM-CYCLE in polynomial time. Here is the algorithm for HAM-CYCLE. On input $G = (V, E)$, we run the algorithm for X on input $\langle G, |V| \rangle$ and the output of this algorithm will be the output of the algorithm for HAM-CYCLE. This works correctly because a Hamiltonian cycle is just a cycle of length $|V|$ and there can be no cycles of length more than $|V|$ since a cycle cannot contain repeated vertices.

   (b) First we argue that X is in NP. As a certificate we take a Hamiltonian path that starts at vertex v. The verifier checks that the path is a valid path, that it visits every vertex exactly once, and that it starts at v. This is easily done in polynomial time.

   Now we argue that X is NP-hard. We reduce HAM-PATH to X. We show that if there is a polynomial time algorithm for X then we can solve HAM-PATH in polynomial time. Here is the algorithm for HAM-PATH. On input G, we go through every vertex v in V and run the algorithm for X on input $\langle G, v \rangle$. If one of these runs returns YES, we output YES. Otherwise we output NO. It is obvious that the algorithm is correct and that it runs in polynomial time provided the algorithm for X runs in polynomial time.

   (c) First we argue that the problem is in NP. A certificate would be numbers $\epsilon_1, \epsilon_2, \ldots, \epsilon_n \in \{-1, 1\}$ and the verifier would check in polynomial time that with these numbers we have $a_1\epsilon_1 + \cdots + a_n\epsilon_n = 0$.

   We will now reduce the SUBSET-SUM problem to our problem. In the SUBSET-SUM problem, we are given positive integers $w_1, w_2, \ldots, w_n$ and W. We want to find out if there is a subset of the w_i's that sum to W. Before we proceed with the reduction, we make two observations.

   First, let $S = \sum_i w_i$. In the SUBSET-SUM problem, without loss of generality, we can assume that $S \geq 2W$. This is because the question of whether there is a subset that sums to W is the same as the question of whether there is a subset that sums to $S - W$ since if a set sums to W, its complement sums to $S - W$ and vice versa. One of W or $S - W$ is less than or equal to $S/2$ so without loss of generality we assume W is always at most $S/2$.

   The second observation is that the problem we are given in the question is the same as the PARTITION problem. In the PARTITION problem, we are given positive integers

$a_1, a_2, \ldots, a_n$ and asked if there is a set $A \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in A} a_i = \sum_{i \notin A} a_i$. The equivalence is clear since the ϵ_i with value 1 represent A and ϵ_i with value -1 represent the complement of A.

Now we present the reduction from SUBSET-SUM to PARTITION. Given an instance of SUBSET-SUM, $\langle \{w_1, \ldots, w_n\}, W \rangle$, we create the PARTITION instance $\{a_1, a_2, \ldots, a_n, a_{n+1}\}$, where $a_i = w_i$ for $i \in \{1, 2, \ldots, n\}$ and $a_{n+1} = S - 2W$, where as before $S = \sum_i w_i$. If $S - 2W = 0$, we ignore a_{n+1}. Now we claim that the SUBSET-SUM instance is a YES instance if and only if the PARTITION instance we created is a YES instance. Let's now prove this. Suppose the SUBSET-SUM instance is a yes instance so that there is a subset $B \subseteq \{1, 2, \ldots, n\}$ so that $\sum_{i \in B} w_i = W$. Let $A = B \cup \{n + 1\}$. Then $\sum_{i \in A} a_i = W + S - 2W = S - W$, and $\sum_{i \notin A} a_i = S - W$. Therefore the PARTITION instance we created is a YES instance as well. Conversely, assume that the PARTITION instance we created is a YES instance. This means that there is a subset $A \subseteq \{1, 2, \ldots, n + 1\}$ so that $\sum_{i \in A} a_i = \sum_{i \notin A} a_i$. Without loss of generality we assume $n + 1 \in A$ (otherwise take A to be the complement of A). Let's denote by A' the set A minus the element $n + 1$. Then the above inequality implies

$$\sum_{i \in A'} a_i + S - 2W = \sum_{i \notin A'} a_i.$$

Rearranging we get

$$\sum_{i \in A'} a_i = \sum_{i \notin A'} a_i - S + 2W.$$

Substituting $S = \sum_{i=1}^{n} a_i$ and canceling, we get

$$\sum_{i \in A'} a_i = -\sum_{i \in A'} a_i + 2W,$$

which implies

$$\sum_{i \in A'} a_i = W.$$

Equivalently,

$$\sum_{i \in A'} w_i = W.$$

Therefore the corresponding SUBSET-SUM instance is a YES instance.

(d) First we argue that the problem is in NP. As a certificate we take numbers $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$, each in $\{1, 2, 3\}$, and verify that

$$\prod_{ij \in E} |x_i - x_j| \neq \prod_{ij \in E} |y_i - y_j|.$$

It is clear that this can be done in polynomial time.

Let's call our problem X. Now we argue that X is NP-hard. We reduce 3-COLORABILITY to X. In fact we will see that X is essentially the same problem as 3-COLORABILITY. We claim that a graph G is a YES instance of X if and only if G is a YES instance of 3-COLORABILITY. Let's prove this claim. If G is a YES instance of X, then there exists numbers $x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n \in \{1, 2, 3\}$ so that

$$\prod_{ij \in E} |x_i - x_j| \neq \prod_{ij \in E} |y_i - y_j|.$$

This implies that right-hand-side and left-hand-side cannot be both 0. Without loss of generality let's assume

$$\prod_{ij \in E} |x_i - x_j| \neq 0.$$

Then we know that each term in the product is non-zero, i.e. for each edge $ij \in E$, $x_i \neq x_j$. Therefore these x_i's are a legal coloring of G with 3 colors. So G is a YES instance of 3-COLORABILITY. To prove the converse, assume G is a YES instance of 3-COLORABILITY. Let the colors be $\{1, 2, 3\}$ and let $x_1, x_2, \ldots, x_n \in \{1, 2, 3\}$ be the coloring of the vertices. Since we have a legal coloring, we must have that for each edge $ij \in E$, $x_i \neq x_j$. This means $|x_i - x_j| \neq 0$. Therefore

$$\prod_{ij \in E} |x_i - x_j| \neq 0.$$

On the other hand, take the numbers $y_1 = y_2 = \cdots = y_n = 1$. For these numbers, we have

$$\prod_{ij \in E} |y_i - y_j| = 0.$$

Therefore there exists numbers $x_1, \ldots, x_n, y_1, \ldots, y_n \in \{1, 2, 3\}$ such that

$$\prod_{ij \in E} |x_i - x_j| \neq \prod_{ij \in E} |y_i - y_j|,$$

which means G is a YES instance of X.

(e) First we show that the problem is in NP. A valid certificate would be an assignment to the variables that satisfies all the constraints. It is then easy to check that this assignment indeed satisfies all the constraints in polynomial time.

To show that the problem is NP-hard, we show that with the new constraints, we can create any 0-1 Integer-Program, i.e. a Linear-Program where we force some or all the variables to be $0/1$ valued. We can force the variables to be $0/1$ valued with the following constraints:

$$(1 - x_i)x_i = 0.$$

Note that for this equality to be satisfied, either $x_i = 0$ or $1 - x_i = 0$, i.e. either $x_i = 0$ or $x_i = 1$. Observe that these constraints can be rewritten as

$$x_i x_i' - x_i^2 = 0,$$

$$x_i' = 1.$$

So any 0-1 Integer-Program can be written with a Linear-Program plus the constraints $\sum_{i,j=1}^{n} a_{ij} x_i x_j = b$. 0-1 Integer-Programming is obviously NP-hard: we can represent many NP-hard problems with an 0-1 Integer-Program. For example we can represent 3SAT. First force the variables to be $0/1$ valued as shown above. Then for every clause $(\ell_1 \vee \ell_2 \vee \ell_3)$, where ℓ_i denotes a literal (a variable or its negation), create the constraint $f(\ell_1) + f(\ell_2) + f(\ell_3) > 0$. Here, if $\ell_i = x_i$ then $f(\ell_i) = x_i$ and if $\ell_i = \bar{x_i}$ then $f(\ell_i) = (1 - x_i)$. Note that for any $0/1$ assignment to the variables, $f(\ell_i) \geq 0$. Let's now argue that this reduction is correct. If the 3SAT formula was satisfiable, then there is an assignment to the variables that makes each clause satisfied. If a clause is satisfied,

there is at least one literal that is set to 1. For that literal, $f(\ell) = 1$ and therefore the corresponding constraint is satisfied. Since each clause is satisfied, each constraint of the Integer-Program is satisfied. Conversely, if the Integer-Program is satisfiable, there is a $0/1$ valued assignment to the variables that makes each constraint satisfied. So for each constraint, there corresponds an ℓ so that $f(\ell) > 0$. This ℓ is set to 1 and makes the corresponding 3SAT clause satisfied.

(f) First we show that the problem is in NP. A valid certificate would be an assignment to the variables that satisfies all the constraints. It is then easy to check that this assignment indeed satisfies all the constraints in polynomial time.

Similar to the previous question, with the new constraints we can force the variables to be -1/1 valued. For each variable, add the constraints

$$|x_i| = 1.$$

Equivalently

$$|x_i| \geq 1,$$

$$-1 \leq x_i \leq 1.$$

Again, there are many NP-hard problems that we can represent with -1/1 Integer-Programming. Similar to the above question, we can represent 3SAT. For every clause $(\ell_1 \vee \ell_2 \vee \ell_3)$, where ℓ_i denotes a literal (a variable or its negation), create the constraint $f(\ell_1)+f(\ell_2)+f(\ell_3) > -3$. Here, if $\ell_i = x_i$ then $f(\ell_i) = x_i$ and if $\ell_i = \bar{x_i}$ then $f(\ell_i) = -x_i$. The argument that this reduction works is the same as in the previous question.