Introduction to Complexity and Computability - Homework 5

Andrew McIsaac

December 15, 2021

Homework Problem 1

Show that the problem COVER-ORIENTED-CYCLES is NP-complete.

First show that it is in NP. Create a verifier V, which, on input $\langle\langle G, k\rangle, S\rangle$, where S is the certificate for the problem, for every vertex in S, removes edges incident from the vertex and the vertex from the graph G.

On the new graph, use depth-first search to traverse the graph and check for an oriented cycle. This can be done in polynomial time using recursion and keeping a stack of visited vertices for every vertex, so that if any vertex is visited more than once in a given vertex's recursion stack (hence there is a cycle) the algorithm can return that there exists a cycle. Therefore V rejects, as there is a cycle which has all of its vertices not in S. If there is no cycle, all possible oriented cycles must be covered by S, and the verifier accepts (it also checks that $|S| \leq k$).

Now use a reduction from VERTEX-COVER to show that COVER-ORIENTED-CYCLES is NP-hard.

Given a graph $G = (V, E)$, k, and S from VERTEX-COVER, we want a $G' = (V', E')$, k', and S', such that S contains at least one vertex from every edge in G if and only if S' contains at least one vertex from every oriented cycle in G'. Convert G into an oriented graph by taking every edge $\{u, v\} \in E$ and creating directed edges in both directions, so that $\{u \mapsto v\} \in E'$ and $\{v \mapsto u\} \in E'$.

Define $V' = V$, $k' = k$, and $S' = S$. Then if S is a yes-instance of VERTEX-COVER then no cycle will be in G' that does not have a vertex from S' because every edge in G has a vertex in S and we do not introduce any new edges between different vertices that were not in E in some direction.

If S is a no-instance of VERTEX-COVER then in G there will be (at least) a 2-cycle between two vertices that share an incident edge but are not in S, and in G' the same two vertices will not be in S'. Since every edge is converted to a 2-cycle by the reduction there is an oriented cycle that contains no vertices from S' in G', and S' is a no-instance of COVER-ORIENTED-CYCLES.

Finally, it is clear that the reduction can be carried out in polynomial time, as the conversion of the graph can be done in time linear with respect to E.

Homework Problem 2

(a) Show that the PARTITION-PROBLEM is polynomially reducible to the KNAPSACK problem.

Let the KNAPSACK problem have sizes $s'(a) = v'(a) = s(a)$ where $s(a)$ are the values associated with each $a \in A$ from PARTITION-PROBLEM. Let capacity C and value V be $C = V = \frac{1}{2} \sum_{a \in A} s(a)$, and let the subset A' from KNAPSACK be identical to the subset A' from PARTITION-PROBLEM. The sets A are identical in both problems as well.

Then for the PARTITION-PROBLEM there is $A' \subseteq A$ such that $\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)$ if and only if for the KNAPSACK problem there is $A' \subseteq A$ such that $\sum_{a \in A'} s'(a) \leq C$ and $\sum_{a \in A'} v'(a) \geq V$.

For a yes-instance, A' has capacity exactly C and value exactly V. That is, $\sum_{a \in A'} s'(a) = C$ and $\sum_{a \in A'} v'(a) = V$.

For a no-instance, either the capacity of A' is larger than C or the value of A' is smaller than V, breaking one of the constraints.

It is clear that this reduction can be done in polynomial time simply by assigning values to C and V and keeping other variables the same.

(b) Show that the PARTITION-PROBLEM is polynomially reducible to the SCHEDULING problem.

Let the number of processors $m = 2$. Let the set of tasks T be equal to the set A from PARTITION-PROBLEM, and $d(x) = s(a)$ for every respective element of the respective sets T and A. Let $D = \frac{1}{2} \sum_{x \in T} d(x)$ (that is, half the sum of the duration of all tasks x in T).

Then the reduction from PARTITION-PROBLEM assigns A' to T_1 and $A \setminus A'$ to T_2. If A' is a yes-instance for PARTITION-PROBLEM then for both T_1 and T_2 $\sum_{x \in T_i} d(x) = D$. If A' is a no-instance then one of T_1 or T_2 will have a sum of durations larger than D, thus breaking the constraint for the SCHEDULING problem. So $A' \subseteq A$ such that $\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)$ is in PARTITION-PROBLEM if and only if T is partitioned into m parts $T_1, \ldots, T_m$ such that $\sum_{x \in T_i} d(x) \leq D$ for every $1 \leq i \leq m$ in SCHEDULING.

Finally, it is obvious that the reduction is done in polynomial time.