# Exercise 10: More NP-Completeness and Beyond

1. The Knapsack problem is defined as follows. We have $n$ items, each with positive integral *weight* $w_j$ $(j = 1, \cdots, n)$ and positive integral *value* $c_j$ $(j = 1, \cdots, n)$ and an integer $b$. The question is to find a subset of the items with total weight at most $b$ and maximal value.

   Answer the following questions:

   (a) Give the decision version of the knapsack problem, KNAPSACK.

   (b) Prove that KNAPSACK problem is NP-complete.

2. Given a graph $G = (V, E)$, a *vertex cover* is a subset $C$ of vertices in $V$ such that for any edge $(u, v) \in E$, $\{u, v\} \cap C \geq 1$. In the *Minimum Vertex Cover* problem, we aim at finding the minimum vertex cover in the given graph.

   (a) Give the decision version of the Minimum Vertex Cover, VC

   (b) Show that the decision version of the Minimum Vertex Cover, VC, is NP-complete.

3. Consider the maximum weighted vertex cover problem: given a graph $G = (V, E)$ and each vertex $v \in V$ has weight $w(v)$, find a vertex cover with minimum weight.

   Answer the following questions:

   (a) Give the decision version of the minimum vertex cover problem, WEIGHTEDVC.

   (b) Prove that WEIGHTEDVC is NP-hard.

4. Recall that in the load balancing problem, there are $m$ machines and $n$ jobs, where job $i$ has processing time $p_i$. The load of a machine is the total processing time of the jobs assigned to it. The goal is to assign the jobs on the machines such that the highest load among the machines is minimized.

   Show that the load balancing problem is NP-complete.

5. Consider the *machine minimization* problem as follows. There are $n$ jobs $J_1, J_2, \cdots, J_n$. Each job $J_i$ has *processing time* $p_i$ and *feasible interval* $I_i = [r_i, d_i]$ where $J_i$ should be assigned. There are unlimited number of machines. Each machine can execute at most one job at a time. A *feasible* schedule is an assignment of every job $J_i$ to a machine $M_j$ at certain time $t_i$ such that $[t_i, t_i + p_i] \subseteq [r_i, d_i]$, and there is no other jobs $J_k$ assigned to the same machine such that $[t_k, t_k + p_k] \cap [t_i, t_i + p_i] \neq \phi$.

   The decision version of the machine minimization problem is

   $$\text{MACHINEMINIMIZATION} = \{\langle S, P, L, k\rangle \mid S = \{J_1, J_2, \cdots, J_n\}, P = \{p_1, p_2, \cdots, p_n\}, \text{ and}$$
   $$L = \{I_1, I_2, \cdots, I_n\}.\ S \text{ is a set of jobs where each job } J_i \text{ has processing time } p_i$$
   and feasible interval $I_i$. The jobs in $S$ can be feasibly scheduled on at most $k$ machines.$\}$

   Show that MACHINEMINIMIZATION is NP-hard.

6. Given a graph $G = (V, E)$, a *dominating set* is a set of vertices $D \subseteq V$ such that for any vertices $v \in V$, $v \in D$ or at least one of the neighbors of $v$ is in $D$. The goal of the Minimum Dominating Set problem is to find a dominating set of the given graph with the minimum cardinality.

   Show that the Minimum Dominating Set problem is NP-complete.

7. The EmptyTM language is defined as $\{\langle M \rangle \mid M$ is a Turing machine and $M$ accepts nothing$\}$. Show that EmptyTM is undecidable.