Exercise set 6

NP-completeness

6.1 Two computational problems P_1 and P_2 are called *polynomial-time equivalent* if there exist polynomial-time reductions $P_1 \leqslant P_2$ and $P_2 \leqslant P_1$. Prove or disprove: Every two NP-Complete problems are polynomial-time equivalent.

6.2 Let IS-HAM-CYCLE denote the computational problem that, given an undirected graph G, decides whether G contains just those edges necessary to form a Hamiltonian cycle in G (no more, no less). Prove or disprove: IS-HAM-CYCLE is NP-Complete.

6.3 Let $G = (V, E)$ with $|V| = n$. Devise a polynomial-time algorithm to decide whether G contains an $(n-1)$-clique. Why does this polynomial-time algorithm not prove $\mathrm{P} = \mathrm{NP}$?

6.4 Let LONG-CYCLE denote the computational problem of deciding whether an undirected graph G on n vertices contains a cycle of length $\geqslant n/2$. Prove that LONG-CYCLE is NP-complete.

6.5 Prove that the problem r-COLOR for undirected graphs is NP-Complete for all $r \geqslant 3$.

6.6 Let DOUBLE-SAT denote the problem of deciding whether a Boolean formula ϕ has at least two satisfying assignments. Prove that this problem is NP-Complete.

6.7 A Boolean formula is said to be in the *disjunctive normal form* (or *DNF* or the *sum-of-products form*) if it is the disjunction (OR) of conjunctions (AND) of literals. For example, $(x_1 \wedge \bar{x}_3) \vee (\bar{x}_2 \wedge \bar{x}_3 \wedge x_4) \vee (\bar{x}_2)$ is in the DNF. By DNF-SAT, we refer to the computational problem of deciding whether a Boolean formula in the DNF is satisfiable. Prove that DNF-SAT $\in$ P.

6.8 Let ϕ be a CNF formula in m variables $x_1, x_2, \ldots, x_m$. We say that ϕ is not-all-equal statisfiable if for some truth assignment of the input variables, each clause of ϕ has at least one true literal and at least one false literal. The problem NAESAT stands for deciding whether ϕ is not-all-equal satisfiable. Prove that NAESAT is NP-Complete.

6.9 Let ϕ be as in Exercise 6.8. If each clause of ϕ contains exactly k literals, then the NAE satisfiability problem for ϕ is called NAE-k-SAT. Prove that NAE-3-SAT is NP-Complete.

6.10 Two (simple undirected) graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are said to be *isomorphic* if there exists a bijection $f : V_1 \to V_2$ such that $(u, v) \in E_1$ if and only if $(f(u), f(v)) \in E_2$.

A *subgraph* of a graph $G = (V, E)$ is a graph $H = (V', E')$ with $V' \subseteq V$ and with $(u, v) \in E'$ whenever $(u, v) \in E$ (for $u, v \in V'$). In other words, H is a subgraph of G if its vertex set is a subset of the vertex set of G and if all edges of G with both vertices in $V(H)$ are also edges of H.

Let SUBGRAPH-ISOMORPHISM denote the computational problem to decide, for the input of two graphs G and G', whether G contains a subgraph isomorphic to G'. Prove that SUBGRAPH-ISOMORPHISM is NP-Complete.

6.11 Let GRAPH-ISOMORPHISM denote the computational problem to decide whether two graphs G and G' are isomorphic. Prove that GRAPH-ISOMORPHISM is a problem in NP. (**Remark:** GRAPH-ISOMORPHISM is perhaaps not NP-Complete. László Babai in 2015 presents a quasi-polynomial-time algorithm for this problem. Babai's claim is still under scientific scrutiny.)

6.12 Let $G = (V, E)$ be an undirected graph. Your task is to find out a largest subset $X \subseteq V$ such that every edge $e \in E$ has at most one endpoint in X. Prove that this problem in NP-Complete.

6.13 Let S be a finite set, and $S_1, S_2, \ldots, S_k$ a collection of subsets of S. A subcollection $S_{i_1}, S_{i_2}, \ldots, S_{i_l}$ with $1 \leqslant i_1 < i_2 < \cdots < i_l \leqslant k$ is called a *cover* of S if $S = \bigcup_{j=1}^{l} S_{i_j}$. In this case, l (the number of subsets in the cover) is called the *size* of the cover. The decision problem SET-COVER takes as input a set S, a collection $S_1, S_2, \ldots, S_k$ of subsets of S, and a positive integer l, and decides whether S has a cover (in the given subsets) of size exactly l. Prove that SET-COVER is an NP-Complete problem. You may assume any standard representation of sets (such as sorted/unsorted arrays, linked lists, or trees).

[H^2] **6.14** Let S be a given set of n positive integers, and t a given positive integer. The problem of deciding whether t equals the sum of the elements of some subset of S is called the *subset-sum problem*. Prove that the subset-sum problem is NP-Complete.

6.15 This exercise is a generalization of SUBSET-SUM (Exercise 6.14). Here, we allow S to consist of negative integers. Likewise, t can be positive, negative, or zero. Is this problem still NP-Complete?

6.16 A set $S = \{a_1, a_2, a_3, \ldots, a_n\}$ of positive integers is called *superincreasing* if $a_i > a_1 + a_2 + \cdots + a_{i-1}$ for all $1 \leqslant i \leqslant n$. Prove that the subset-sum problem for a superincreasing set S is in P.

6.17 Let S be a given set of n positive integers with $2t = \sum_{a \in S} a$. The problem of deciding whether t equals the sum of the elements of some subset of S is called the *partition problem*. (In this case, the remaining elements of S too sum up to t.) Prove that the partition problem is NP-Complete.

6.18 Let S be a given set of n positive integers. The task is to partition S into two (disjoint) subsets A and B such that $S = A \cup B$, and the absolute difference $|\sum_{a \in A} a - \sum_{b \in B} b|$ is as small as possible. This is called the *balanced set partition problem*. Pose this problem as an equivalent decision problem, and prove that this problem is NP-Complete.

6.19 Let S be a set of n positive integers, and T a target integer (positive, negative, or zero). The task is to partition S into two disjoint subsets A, B (with $A \cup B = S$) such that $\sum\limits_{a \in A} a - \sum\limits_{b \in B} b = T$. Prove that this SUBSET-DIFF problem is NP-Complete.

6.20 Let $A = (a_1, a_2, a_3, \ldots, a_n)$ and $B = (b_1, b_2, b_3, \ldots, b_n)$ be two arrays of positive integers. You are also given a target sum T (a positive integer). Your task is to find a partition of the index set into two disjoint subsets I, J with $I \cup J = \{1, 2, \ldots, n\}$ such that $\sum\limits_{i \in I} a_i + \sum\limits_{j \in J} b_j = T$. Prove that this 2-SET-SUM problem is NP-Complete.

6.21 Let $A = (a_1, a_2, a_3, \ldots, a_n)$ and $B = (b_1, b_2, b_3, \ldots, b_n)$ be two arrays of positive integers. Your task is to find a partition of the index set into two disjoint subsets I, J with $I \cup J = \{1, 2, \ldots, n\}$ such that $\sum\limits_{i \in I} a_i = \sum\limits_{j \in J} b_j$. Prove that this 2-SET-PARTITION problem is NP-Complete.

6.22 You have n objects of positive integer weights $w_1, w_2, \ldots, w_n$, and (an unlimited number of) bins each with weight capacity C. The BIN-PACKING problem deals with the packing of the objects in the smallest possible number of bins such that the weight capacity of no bin is exceeded. Pose this problem as an equivalent decision problem, and prove that this problem is NP-Complete.

6.23 Consider the following decision version of the knapsack problem. Given n objects $O_1, O_2, \ldots, O_n$ with respective weights $w_1, w_2, \ldots, w_n$ and with respective profits $p_1, p_2, \ldots, p_n$, and given a knapsack of capacity C and a profit bound P, decide whether there exists a subcollection $O_{i_1}, O_{i_2}, \ldots, O_{i_k}$ of the given objects such that $\sum_{j=1}^{k} w_{i_j} \leqslant C$ (knapsack capacity cannot be exceeded), and $\sum_{j=1}^{k} p_{i_j} \geqslant P$ (at least a profit of P can be made).

(a) Prove that the decision version of the knapsack problem can be solved in polynomial time if and only if the optimization version of the knapsack problem can be solved in polynomial time.

(b) Prove that the decision version of the knapsack problem is NP-Complete.

6.24 [*Ferry-loading problem*] Several vehicles wait in a long queue near a river bank. The lengths of the vehicles are $l_1, l_2, l_3, \ldots, l_n$ (positive integers) in that order from the beginning to the end of the queue. A big boat with two decks (left and right) comes to carry the vehicles across the river. Each deck of the boat has length L (again a positive integer). The vehicles must be loaded to the boat in the order they appear in the queue. For each vehicle, a decision is to be made about which deck it will join (provided that there is enough length remaining in that deck). The objective is to maximize the number of vehicles that can be loaded. For example, if $L = 10$, and the first four vehicles have lengths $5, 5, 6, 4$, then all the four of them can be loaded ($5 + 5 = 4 + 6 = 10$). However, if the first two vehicles are loaded to different decks, then no other vehicle can be loaded.

Let us consider the decision problem whether all of the vehicles can be loaded. Clearly, if $l_1 + l_2 + \cdots + l_n > 2L$, the answer is *No*. So assume that $l_1 + l_2 + \cdots + l_n \leqslant 2L$.

(a) Prove that the optimization problem is polynomial-time equivalent to the decision problem.

(b) Prove that the decision problem in NP-Complete.

6.25 Let C be a set of classes, each having a fixed time schedule (start time and end time). If two classes have overlapping times, the classes cannot be assigned to the same classroom. Given a positive integer k, we want to find out whether k classrooms suffice for scheduling all the classes in C on a single day. Prove/Disprove: This problem in NP-Complete.

6.26 Let C be a set of classes as in Exercise 6.25. Each class is now specified by its duration, can be scheduled at any time of the day, but must be allowed to finish to completion. Given a positive integer k, we want to find out whether k classrooms suffice for scheduling all the classes in C on a single day. Prove/Disprove: This problem in NP-Complete.

6.27 Let $G = (V, E)$ be an undirected graph. A cut of G is a partitioning of V into two disjoint subsets V_1, V_2. The size of the cut is the number of edges whose endoints are in the two different subsets.

[H^2] **(a)** [*The Max-Cut Problem*] Prove that the problem of deciding whether a given graph G has a cut of size greater than or equal to a given integer k is NP-Complete.

(b) [*The Min-Cut Problem*] Prove that the problem of deciding whether a given graph G has a cut of size less than or equal to a given integer k is in P.

6.28 Let $G = (V, E)$ be an undirected graph. A 3-cut of G is a partitioning of V into three mutually disjoint subsets V_1, V_2, V_3. The size of the 3-cut is the number of edges whose endoints are in two different subsets. The MAX-3-CUT problem is to decide whether G has a 3-cut of size at least as large as a specified integer k. Prove that MAX-3-CUT is NP-Complete.

[HM] **6.29** Prove that an undirected graph G contains an Eulerian tour if and only if G is connected with every vertex having even degree.

6.30 Prove that if P = NP, every problem in P = NP is NP-Complete.

6.31 Consider the problem of deciding whether a given C program P, upon input I, prints any character (to `stdout`).

(a) Prove that this decision problem is unsolvable.

(b) Prove that this decision problem is NP-Hard.

6.32 Let P be a (decision) problem in NP. Consider the problem $\bar{P}$ having the property Accept$(\bar{P}) =$ Reject(P) (and, of course, Reject$(\bar{P}) =$ Accept(P)). We say that $\bar{P}$ is the complement of the problem P. The class of all complements of problems in NP is denoted by coNP. A problem in coNP has succinct *disqualifications* ("certificates" in the negative sense) for all inputs to be rejected.

(a) Prove that the primality testing problem is in coNP.

(b) State in English the complement of the problem HAM-PATH.

(c) Prove or disprove: DNF-SAT is the complement of CNF-SAT (See Exercise 6.7).

(d) A Boolean formula ϕ is called a *tautology* if it evaluates to *true* for all possible truth assignments of its variables. Prove that deciding whether a Boolean formula is a tautology is in coNP.

(e) A Boolean formula ϕ is called a *contradiction* if it evaluates to *false* for all possible truth assignments of its variables. Prove that deciding whether a Boolean formula is a contradiction is in coNP. What is its complement problem called?

6.33 Prove that $\mathrm{P} \subseteq \mathrm{NP} \cap \mathrm{coNP}$. (**Remark:** It is not known whether the containment is proper.)

6.34 Let SMALL-FACTOR denote the problem of deciding whether a given positive integer n has a factor no larger than a given positive integer k. Prove that SMALL-FACTOR is in $\mathrm{NP} \cap \mathrm{coNP}$.

6.35 Prove that if SMALL-FACTOR can be solved in polynomial time, integers can be factored in polynomial time too, and conversely.

6.36 Prove that if $\mathrm{NP} \neq \mathrm{coNP}$, then $\mathrm{P} \neq \mathrm{NP}$. (**Remark:** It is widely believed that $\mathrm{NP} \neq \mathrm{coNP}$.)

6.37 Prove that if an NP-Complete problem is in coNP, then $\mathrm{NP} = \mathrm{coNP}$.

6.38 Define coNP-Hard and coNP-Complete problems as those to which there exist polynomial-time reductions from *all* problems in coNP. Prove that a problem is coNP-Complete if and only if its complement is NP-Complete. State a few coNP-Complete problems.

6.39 Prove that the TAUTOLOGY problem (see Exercise 6.32(d)) is coNP-Complete.

6.40 Prove that the CONTRADICTION problem (see Exercise 6.32(e)) is coNP-Complete.

6.41 Prove or disprove: The halting problem is coNP-hard.

6.42 Let FSAT denote the problem of finding a satisfying assignment of a Boolean formula ϕ or return failure if ϕ is unsatisfiable. Prove that FSAT can be solved in polynomial-time if and only if $\mathrm{SAT} \in \mathrm{P}$.

Approximation algorithms

6.43 Consider the optimization version of the set covering problem of Exercise 6.13. That is, given a finite set S and a collection of k subsets $S_1, S_2, \ldots, S_k$ of S, we intend to find out a cover of S (from the given collection) of size as small as possible. Let us denote this optimization problem by MIN-SET-COVER.

Let $S = \{x_1, x_2, \ldots, x_n\}$ be of size n, and let f_i be the count of the subsets S_j containing the element x_i. Finally, let $f = \max(f_1, f_2, \ldots, f_n)$. (The counts f_i are the frequencies of the elements, and f is the maximum frequency.)

Design a polynomial-time f-approximation algorithm for MIN-SET-COVER. Establish that your algorithm achieves an approximation ratio of f. Determine whether this approximation ratio is tight.

6.44 Prof. Myopia proposes the following approximation algorithm for solving the MAX-CUT problem (Exercise 6.27).

1. Start with an arbitrary partition S, T of V.