Over this module, we will be dealing with quite a few problems that you may not be familiar with already. It is very important that you take the time to go through the specifications and understand precisely what the problem at hand is.

Decision problems. These are problems in which we query whether the underlying object(s) has(have) some property(properties). An algorithm for the problem would return either YES or NO, depending on whether the property is satisfied or not. We say that an instance I of a decision problem Π is an YES-instance (resp. NO-instance) of Π if an algorithm solving Π would output YES (resp. NO) when I is given as input. For example, defined below is the decision version of the Vertex Cover problem. (Recall that a set $S \subseteq V(G)$ is a *vertex cover* of G if and only if for every edge $(u, v) \in E(G)$, $S \cap \{u, v\} \neq \emptyset$.)

VERTEXCOVER
Input: A graph $G(V, E)$ and a positive integer $k \leq |V|$.
Question: Does G contain a vertex cover $S \subseteq V$, such that $|S| \leq k$?

From the perspective of formal languages, the set of all input instances (under an appropriate binary encoding) that a decision problem Π accepts can be considered as a language (that is, a subset of $\{0, 1\}^*$). We can then talk about problems in terms of their corresponding languages and even classify languages into complexity classes.

Optimization problems. In optimization problems, our objective is to find the smallest (or largest, as appropriate) value that some characteristic of the underlying object can take. An algorithm for the problem would return said value. An optimization version of the Vertex Cover problem given above follows.

VERTEXCOVER-OPT
Input: A graph $G(V, E)$.
Question: What is the smallest k for which G contains a vertex cover of size k?

Search problems. Given some property that is satisfied by the underlying object, search problems are concerned with finding some structure in the object that *witnesses* the truth of the said property. For example, an algorithm for the following search version of the Vertex Cover problem would output a vertex cover of size k, if such a vertex cover exists.

VERTEXCOVER-SEARCH
Input: A graph $G(V, E)$ and a positive integer $k \leq |V|$.
Question: Find a vertex cover S of G, with $|S| = k$, if such a vertex cover exists.

A given decision problem need not have any optimization problems corresponding to it. But, every search or optimization problem has an underlying decision version.

Listed below, for easy reference, are the definitions of some decision problems. We may see these, as well as some of their variants, across this module. As with VERTEXCOVER, the -OPT and -SEARCH suffixes can, respectively, be used to refer to the corresponding optimization and search versions.

Given a graph $G(V, E)$, a set $S \subseteq V$ is an *independent set* in G if and only if for any $u, v \in S$, $(u, v) \notin E$ (in other words, the induced subgraph $G[S]$ has no edges).

INDEPENDENTSET
Input: A graph $G(V, E)$ and a positive integer $k \leq |V|$.
Question: Does G contain an independent set $S \subseteq V$, such that $|S| \geq k$?

Note that the *induced subgraph* $G[S]$ is the subgraph of G obtained by deleting every vertex in $V \setminus S$ from G, along with the edges incident on these vertices.

Given a graph $G(V, E)$, a set $S \subseteq V$ forms a *clique* in G if and only if the induced subgraph $G[S]$ is a complete graph.

CLIQUE
Input: A graph $G(V, E)$ and a positive integer $k \leq |V|$.
Question: Is there a set $S \subseteq V$, with $|S| \geq k$, such that S forms a clique in G?

In set theory, a set $\mathcal{F}$ containing subsets of another set U is often referred to as a *family of sets* over U; the set U itself is referred to as the *universe*. Given a family $\mathcal{F}$ of sets over U, we say that a sub-family $\mathcal{H} \subseteq \mathcal{F}$ is a *set cover* of U if and only if $U = \bigcup_{S \in \mathcal{H}} S$ (that is, every element in U is present in some set S in $\mathcal{H}$).

SETCOVER
Input: A universe set U, a family $\mathcal{F}$ of sets over U and an integer $k \leq |\mathcal{F}|$.
Question: Is there a set $\mathcal{H} \subseteq \mathcal{F}$, with $|\mathcal{H}| \leq k$, such that $\mathcal{H}$ covers U?

Given a family $\mathcal{F}$ of sets over an universe U, a set $H \subseteq U$ is an *hitting set* of G if and only if for each set $S \in \mathcal{F}$, $H \cap S \neq \emptyset$.

HITTINGSET
Input: A universe set U, a family $\mathcal{F}$ of sets over U and an integer $k \leq |\mathcal{F}|$.
Question: Is there a set $H \subseteq U$, with $|H| \leq k$, such that $\mathcal{H}$ is a hitting set of U?

Given a boolean formula in Conjunctive Normal Form (CNF), on n boolean variables and having m disjunctive clauses, we say that the formula is *satisfiable* if and only if there exists an assignment of truth values to the n variables such that the formula take truth value TRUE. In other words, the assignment of truth values should be such that at least one literal in every clause has truth value TRUE.

SAT
Input: A boolean formula $\mathcal{F}$ with m clauses $C_1, \ldots, C_m$ over a set of n variables $x_1, \ldots, x_n$.
Question: Is $\mathcal{F}$ satisfiable?

UNSATISFIABILITY
Input: A boolean formula $\mathcal{F}$ with m clauses $C_1, \ldots, C_m$ over a set of n variables $x_1, \ldots, x_n$.
Question: Is $\mathcal{F}$ not satisfiable?

3SAT
Input: A boolean formula $\mathcal{F}$ with m clauses $C_1, \ldots, C_m$ over a set of n variables $x_1, \ldots, x_n$, such that each clause has exactly 3 literals.
Question: Is $\mathcal{F}$ satisfiable?

MAXSAT
Input: A boolean formula $\mathcal{F}$ with m clauses $C_1, \ldots, C_m$ over a set of n variables $x_1, \ldots, x_n$, and integer $k \leq m$.
Question: Does there exist a truth assignment that satisfies at least k clauses in $\mathcal{F}$?

A cycle in a graph is a Hamiltonian cycle if and only if the cycle passes through every vertex in the graph. Similarly, a path in a graph is a Hamiltonian path if and only if the path contains every vertex in the graph. For an edge-weighted graph, the weight of a cycle (resp. path) is the sum of weights of the edges lying on the cycle (resp. path).

HAMCYCLE
Input: A directed graph $G(V, E)$.
Question: Does G contain a Hamiltonian cycle?

HAMPATH
Input: A directed graph $G(V, E)$.
Question: Does G contain a Hamiltonian path?

TSP
Input: A directed edge-weighted graph $G(V, E)$, with weights specified by a function $w : E(G) \to \mathbb{R}^+$, and a real value W.
Question: Does G contain a Hamiltonian cycle with weight at most W?

A proper k-coloring of a graph G is a function $\chi : V(G) \to \mathbb{N}_k$ such that if $(u, v) \in E(G)$, then $\chi(u) \neq \chi(v)$. We say that a graph is k-colorable if and only if the graph can be properly colored using at most k colors.

GRAPHCOLORING
Input: A graph $G(V, E)$ and a positive integer $k \leq |V|$.
Question: Is G k-colorable?

Certificates and Verifiers

Let (G, k) be an YES-instance of the VERTEXCOVER problem. Suppose that your objective is to provide a proof that (G, k) is indeed an YES-instance. For this proof, it is sufficient to give a set S such that $S \subseteq V(G)$ is a vertex cover of G and $|S| \leq k$. Moreover, its easy to design an algorithm that takes the instance $(G, k; S)$ as input and verifies whether the set S is enough to prove that (G, k) is an YES-instance of VERTEXCOVER or not.

Here, the set S certifies/witnesses/proves the fact that (G, k) is an YES-instance of VERTEXCOVER, and consequently, we refer to S as a *certificate/witness/proof* for this fact. And the algorithm that takes $(G, k; S)$ as input and verifies whether S is indeed a valid certificate or not is referred to as a verification algorithm, or simply as a *verifier*.

Important Note: When talking about verifiers, we always assume that the certificate S is somehow given; we are not concerned with *finding* a certificate. Finding a certificate is a separate problem/concern.

We can equivalently have certificates and verifiers for NO-instances as well. If (G, k) is a NO-instance of VERTEXCOVER, what would an appropriate certificate to certify this fact be? Think about designing a verifier for such a certificate.

- P is the class of problems that can be solved by an algorithm that takes polynomial time (that is, in $\mathcal{O}(n^c)$ time, for some constant c) in the worst case.
- NP is the class of problems that have polynomial time (in terms of the size of the original instance) verifiers for YES-instances.

1. (Transitivity of reductions) If $A \preceq_{\mathsf{P}} B$ and $B \preceq_{\mathsf{P}} C$, then show that $A \preceq_{\mathsf{P}} C$.

2. Prove the following statements.
   (a) If Π is the decision version of an optimization problem Π-OPT, then $\Pi \preceq_{\mathsf{P}} \Pi$-OPT.
   (b) If Π is the decision version of a search problem Π-SEARCH, then $\Pi \preceq_{\mathsf{P}} \Pi$-SEARCH.

3. Using the verifier based definition of NP, argue that P $\subseteq$ NP.

4. (a) If A is a decision problem such that $A \preceq_{\mathsf{P}} B$ and $B \in$ NP, then show that $A \in$ NP.
   (b) Using Part (a), show that P $\subseteq$ NP.

5. (a) Let S be a vertex cover of a graph $G(V, E)$. Show that the set $I = V \setminus S$ is an independent set in G.
   (b) Suppose that $W \subseteq V$ is *not* a vertex cover of the graph $G(V, E)$. Show that the set $J = V \setminus W$ is not an independent set in G.
   (c) Show that VERTEXCOVER $\preceq_{\mathsf{P}}$ INDEPENDENTSET.

6. Show that VERTEXCOVER-OPT$\preceq_{\mathsf{P}}$ VERTEXCOVER.

7. Show that
   (a) VERTEXCOVER $\preceq_{\mathsf{P}}$ HITTINGSET.
   (b) 3SAT $\preceq_{\mathsf{P}}$ SAT.

8. Consider a decision problem A. We define the decision problem $\overline{A}$ (read as A complement) as follows: I is an YES-instance of $\overline{A}$ if and only if it is a NO-instance of A. If $A \in$ NP, can we conclude that $\overline{A} \in$ NP? Justify your answer.

9. For each of the problems defined in this document, decide what the certificates should be for an YES-instance. Also, design verifiers for each of them and determine which of the problems are in NP.