1 Reduction, deduction, induction, and abduction.

The following question is long, but not very hard, and is intended to make sure you understand the following problems, and the basic concepts needed for proving NP-Completeness.

All graphs in the following have n vertices and m edges.

For each of the following problems, you are given an instance of the problem of size n. Imagine that the answer to this given instance is "yes", and that you need to convince somebody that indeed the answer to the given instance is `yes`. To this end, describe:

(I) An algorithm for solving the given instance (not necessarily efficient). What is the running time of your algorithm?
(II) The format of the proof that the instance is correct.
(III) A bound on the length of the proof (its have to be of polynomial length in the input size).
(IV) An efficient algorithm (as fast as possible [it has to be polynomial time]) for verifying, given the instance and the proof, that indeed the given instance is indeed `yes`. What is the running time of your algorithm?

1.A.

Semi-Independent Set

Instance: A graph G, integer k
Question: Is there a semi-independent set in G of size k? A set $X \subseteq \mathsf{V}(\mathsf{G})$ is semi-independent if no two vertices of X are connected by an edge, or a path of length 2.

1.B.

3EdgeColorable

Instance: A graph G.
Question: Is there a coloring of the edges of G using three colors, such that no two edges of the same color are adjacent?

1.C.

Subset Sum

Instance: S: Set of positive integers. t: An integer number (target).
Question: Is there a subset $X \subseteq S$ such that $\sum_{x \in X} x = t$?

1.D.

3DM

Instance: X, Y, Z sets of n elements, and T a set of triples, such that $(a, b, c) \in T \subseteq X \times Y \times Z$.
Question: Is there a subset $S \subseteq T$ of n disjoint triples, s.t. every element of $X \cup Y \cup Z$ is covered exactly once.?

1.E.

SET COVER

Instance: $(\mathsf{S}, \mathcal{F}, k)$:
S: A set of n elements
$\mathcal{F}$: A family of m subsets of S, s.t. $\bigcup_{X \in \mathcal{F}} X = \mathsf{S}$.
k: A positive integer.
Question: Are there k sets $S_1, \ldots, S_k \in \mathcal{F}$ that cover S. Formally, $\bigcup_i S_i = \mathsf{S}$?

1.F.

CYCLE HATER.

Instance: An undirected graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$, and an integer $k > 0$.
Question: Is there a subset $X \subseteq \mathsf{V}$ of at least k vertices, such that no cycle in G contains any vertex of X.

1.G.

Many Meta-Spiders.

Instance: An undirected graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ and an integer k.
Question: Are there k vertex-disjoint meta-spiders that visits all the vertices of G?

A ***meta-spider*** in a graph G is defined by two vertices u, v (i.e., the head and tail of the meta-spider), and a collection Π of simple paths all starting in v and ending at u, that are vertex disjoint (except for u and v). The vertex set of such a spider is all the vertices that the paths of Π visit (including, of course, u and v).

2 Beware of algorithms carrying oracles. Consider the following optimization problems, and for each one of them do the following:

(I) (2 PTS.) State the natural decision problem corresponding to this optimization problem.
(II) (3 PTS.) Either: (A) prove that this decision problem is NP-COMPLETE by showing a reduction from one of the NP-COMPLETE problems seen in class (if you already seen this problem in class state "seen in class" and move on with your life). (B) Alternatively, provide an efficient algorithm to solve this problem.
(III) (5 PTS.) Assume that you are given an algorithm that can solve the **decision** problem in polynomial time. Show how to solve the original optimization problem using this algorithm in polynomial time, and output the solution that realizes this optimal solution.

An example for the desired solution and how it should look like is provided in the last page.

2.A. (10 PTS.)

NO COVER

Instance: Collection $\mathcal{C}$ of subsets of a finite set S.
Target: Compute the maximum k, and the sets $S_1, \ldots, S_k$ in $\mathcal{C}$, such that $S \nsubseteq \cup_{i=1}^{k} S_i$.

2.B. (10 PTS.)

TRIPLE HITTING SET

Instance: A ***ground set*** $U = \{1, \ldots, n\}$, and a set $\mathcal{F} = \{U_1, \ldots, U_m\}$ of subsets of U.
Target: Find the smallest set $S' \subseteq U$, such that S' hits all the sets of $\mathcal{F}$ at least three times. Specifically, $S' \subseteq U$ is a ***triple hitting set*** if for all $U_i \in \mathcal{F}$, we have that S' contains at least three elements of U_i.

2.C. (15 PTS.)

Max Inner Spanning Tree

Instance: Graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$.
Target: Compute the spanning tree $\mathcal{T}$ in G where the number of vertices in $\mathcal{T}$ of degree two or more is maximized.

2.D. (15 PTS.)

Cover by paths (edge disjoint).

Instance: Graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$.
Target: Compute the minimum number k of paths $\pi_1, \ldots, \pi_k$ that are edge disjoint, and their union cover all the edges in G.