CS340 Analysis of Algorithms
Fall 2025

Homework:	**9**	**Professor:**	**Elizabeth Dinella**
Due Date:	**12/8/25**	**E-mail:**	`edinella@cs.brynmawr.edu`
Office:	**Park 205**	**URL:**	`https://bmc-cs-340.github.io`

1. NP-Complete problems are usually expressed as *decision problems*, where the answer is "yes" or "no". In practice we usually want to know *why* the answer is so, and if "yes", then also *how*. In this problem, we will show that if we are given access to an algorithm for a decision problem, it is often possible to use this to obtain the entire answer.

1. Hamiltonian Cycle: Given an undirected graph $G = (V, E)$, does there exist a cycle that visits every vertex of G exactly once? Suppose that we have a function `Hamiltonian(G)`, which (by some miracle) ran in polynomial time and returns `true` if G has a Hamiltonian cycle and `false` otherwise. Show that if G has a Hamiltonian cycle, then it is possible to use this function (as a black box) to compute the sequence of vertices on the Hamiltonian cycle in polynomial time.
2. 3-colorability: Given an undirected graph $G = (V, E)$ can the vertices of G be labeled with three colors such that no edge is incident to two vertices of the same color? Suppose that we had a function `3Col(G)`, which (by some miracle) ran in polynomial time and returns `true` if G is 3-colorable and `false` otherwise. Show that if G is 3-colorable, then it is possible to use this function (as a black box) to determine the assignment of colors to the vertices in polynomial time. Note that our miracle `3Col` doesn't accept input with partial coloring. In other words, you can not assign some vertex coloring and then have `3Col` check whether your coloring was right or wrong. Or your can think of it as `3Col(G)` will ignore your assignments and return true or false based solely on whether there is a valid 3-coloring of G.

For both of these problems, you should give the usual description, pseudo code and time analysis. You may skip the correctness proof.

2. The Graph Isomorphism problem is the problem of determining if an invertible mapping f can be created between graphs $G = (V, E)$ and $G' = (V', E')$ such that $f(v) = w, v \in V$ and $w \in V'$, and for any $u, v \in V, (f(u), f(v))$ is an edge in E' if and only if (u, v) is an edge in E. For more information and examples on Graph Isomorphism, refer to section 10.4 (4th Ed) or 10.3 (5th Ed) of your Discrete Math textbook by Susan Epp. You might also find `this write-up` interesting. The first section should be generally readable. After that, it relies on more advanced mathematics.

Prove that the Graph Isomorphism Problem is in NP. In order to prove that a problem is in NP, recall that you need to show that a solution can be checked in polynomial time. In other words, you need to create a polynomial-time certifier. Thus, your proof consists of the following parts:

1. A description of the certifier algorithm.
2. Pseudocode for the certifier.
3. A time analysis showing that the certifier is polynomial-time.

Note that for this problm, the input to the certifier are two graphs that are supposed to be isomorphic to each other and a mapping between the vertices of the two graphs. Also note that the "certifiers" for NP only consider "yes" instances. In other words, you only need to verify the "yes" solutions. You do not need to verify any "no" answers.

3. Prove that the Registrar's Problem is NP-Complete. Consider the decision version of the basic registrar's problem: Is it possible to create a schedule to satisfy a given student preferences value?

For this problem, it's fine if you relax/drop some constraints of the problem as defined by your project to get to a version that's more easily represented by a graph, but you should put in a brief argument of why the changes to the constraints do not affect the reduction, i.e. your changes do not disqualify the original problem from being in NPC.

Please hand in your assignment electronically on Gradescope.