# Question 1

To prove that is not **NP**-hard under , a problem will be appropriately chosen, that is in **NP**, and it will be shown that that problem cannot be reduced to . In this case, the chosen problem will be CIRCUIT-SAT, which is already proven to be **NP**-complete. So, it will be shown that CIRCUIT-SAT , which proves that is not **NP**-hard.

CIRCUIT-SAT is the decision problem where given a Boolean circuit, does there exist an assignment of its inputs that makes the output true. In other words, does there exist inputs of and to the circuit that evaluates to . To solve this problem, an algorithm has to try every possible combination of inputs and evaluate the circuit each time to see if it outputs . So, given an input of assignments, although the computation of the circuit to determine the output can be done in constant time, the determination of the inputs is done in exponential time, as the algorithm would have to try every possible combination in the worst-case.

With that being said, now attempt to reduce CIRCUIT-SAT to via . It is stated that the function has the following property: for every , where is the set of all bit-strings. So, in this case, since , then this means that the input to is always a valid bit-string, thus will always evaluate to . This reveals the discrepancy between these two problems: CIRCUIT-SAT will output either or , whereas always outputs . In addition, CIRCUIT-SAT must compute its output based on its input, meaning it must process and determine whether the input is valid, which leads to its exponential time as mentioned earlier, whereas will always evaluate to , leading to a constant time of , putting it in the complexity class **P**.

Since runs in constant time, whereas CIRCUIT-SAT runs in exponential time, it is not possible to say that CIRCUIT-SAT can reduce to . Another perspective: take the bit-string input of , break them up into individual bits, and assign them to some circuit as the inputs. In this case, will evaluate to no matter what the setup of the circuit is, whereas CIRCUIT-SAT may evaluate to or depending on the circuit. Thus, CIRCUIT-SAT cannot reduce to because not only is it more computationally complex than , but it also could result in different outputs for the same input. Therefore, CIRCUIT-SAT , hence is not **NP**-hard.

# Question 2

* Solve this problem by showing: (i) every problem reduces to this problem, and (ii) this problem is itself .
* Carry out something similar to what the proof for Claim 1 Lecture 10(a) does, except that that claim is for **NP** and not **P**.
  + Let be any problem in **P**. We need to show that reduces to our problem.
  + Now we want to leverage what we know about . Specifically, what we know is that there is a polynomial-time algorithm, call it , that given any instance of , correctly outputs true or false.
  + Now we want to think about how we can leverage to reduce to our problem.
* To show that it is in **P**, simply propose a deterministic polynomial-time algorithm.
* It is the **P**-hard part for which a reduction is appropriate.

# Question 3

# Question 4

Example of a valid automorphic mapping:

A picture containing text, clock

Description automatically generated

* So long as at least one vertex is mapped to a different vertex than itself, it is a valid automorphism, because the mapping is no longer the identity.
* is an edge in the one graph if and only if is an edge in the other.

Need to properly modify and , for a polynomial number of times and invoke for each pair. Each modification/transformation should also take polynomial-time.

* Suppose you have and a copy of it . Now, pick a pair of distinct vertices . And now suppose you want to force a mapping of to , and for the other vertices, you don’t care how they are mapped.
* That’s the approach. For every pair of distinct vertices , try and force a mapping of in to in . You know that a valid mapping exists if and only if is automorphic.
* To force a mapping, “hang” something off of each of and that forces any isomorphic mapping to map to only. By “hang something off” we mean construct some kind of graph that forces such a mapping of to only.

# Question 5