Proving that a problem X is NP-hard requires several steps:

- Choose a problem Y that you already know is NP-hard.
- Describe an algorithm to solve Y, using an algorithm for X as a subroutine. Typically this algorithm has the following form: Given an instance of Y, transform it into an instance of X, and then call the magic black-box algorithm for X.
- Prove that your algorithm is correct. This almost always requires two separate steps:
  - Prove that your algorithm transforms "good" instances of Y into "good" instances of X.
  - Prove that your algorithm transforms "bad" instances of Y into "bad" instances of X. Equivalently: Prove that if your transformation produces a "good" instance of X, then it was given a "good" instance of Y.
- Argue that your algorithm for Y runs in polynomial time.

---

1. Recall the following kCOLOR problem: Given an undirected graph G, can its vertices be colored with k colors, so that every edge touches vertices with two different colors?

   (a) Describe a direct polynomial-time reduction from 3COLOR to 4COLOR.

   (b) Prove that kCOLOR problem is NP-hard for any $k \geq 3$.

2. Recall that a *Hamiltonian cycle* in a graph G is a cycle that goes through every vertex of G exactly once. Now, a ***tonian cycle*** in a graph G is a cycle that goes through at least *half* of the vertices of G, and a ***Hamilhamiltonian circuit*** in a graph G is a closed walk that goes through every vertex in G exactly *twice*.

   (a) Prove that it is NP-hard to determine whether a given graph contains a tonian cycle.

   (b) Prove that it is NP-hard to determine whether a given graph contains a Hamilhamiltonian circuit.