

Main Steps

To show a problem X is NP-Complete you must prove that X is in the set NP (pretty easy), and prove that it is at least as hard as some other NP-Complete problem (more involved). In general you'll go through the following steps (although it isn't critical you use this numbering system so long as you have all these components).

Step 1: Prove X is in NP. That is, give an efficient certifier for the problem.

Step 2: Pick an NP-Complete Y problem to reduce to X . As we prove more problems are NP-Complete, we'll have more options for Y . If X is indeed NP-Complete, any other NP-Complete problem must be reducible to X (by definition), but some reductions will be easier than others. Try to find a problem Y that is similar to X ; this won't always work, but has a good shot of giving you a direct reduction. If all else fails, 3-SAT is a good back-up plan.

Step 3: Specify a construction for your reduction. You want to show $Y \leq_p X$. It is easy to go in the wrong direction. You're trying to use a black box for X to solve an arbitrary instance of Y . So you'll assume you've been given an input a , and you want to determine whether a is a member of Y . Typically, you'll specify how to use a to construct a candidate b for problem X with the property that

$$b \in X \iff a \in Y.$$

(I say "typically" because one can actually make a decision based on multiple calls to a black box, by inverting the answer given by the black box, etc.) Your construction of b must require only polynomial time (in the size of a).

Step 4: Prove the \Rightarrow direction. Suppose an arbitrary a is in Y . Prove that the b you constructed is indeed in X .

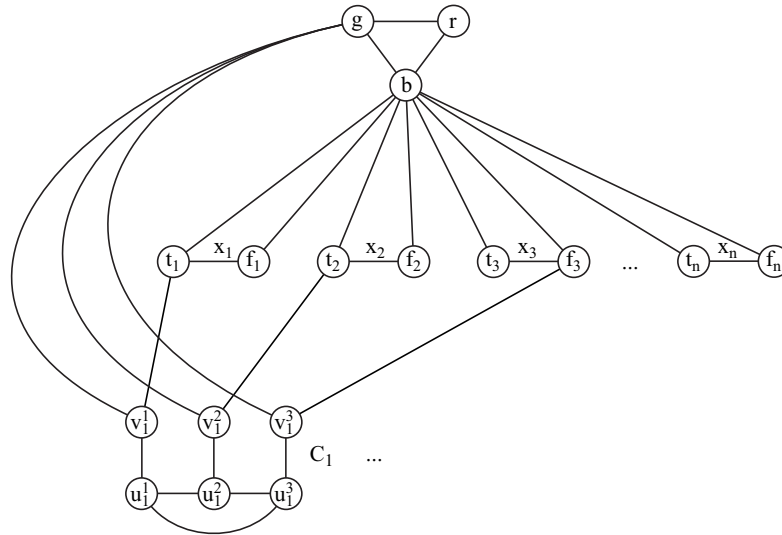
Step 5: Prove the \Leftarrow direction. Suppose an arbitrary b built using your construction is in X . Prove that a must have been in Y .

Steps 2 through 5 constitute the proof that $Y \leq_p X$. Note that proving $X \leq_p Y$ is not terribly useful; such a result just means that X is no harder than Y , but presumably at this point you already know Y is NP-Complete and X is in NP, so while true, this doesn't buy you anything. Thus, your construction only goes in one direction, namely showing that a black box for X can be used to solve Y . The proof that your construction works, however, requires two directions (steps 4 and 5). It is trivial to find constructions that work in one direction.

These proofs can be tricky, and require a fair bit of creativity, but working on various examples will help build your intuition and ability to generate reductions.

An Example: 3-COLORING

A k -coloring of a graph is an assignment of colors to nodes such that every node is assigned one color, no two adjacent nodes are assigned the same color, and only k different colors are used. The 3-COLORING problem is as follows. Given a graph $G = (V, E)$, determine whether there is a 3-coloring of G . We will prove that this problem is NP-Complete.



Proof. First, we argue that 3-COLORING is in NP. Let a certificate be an assignment of colors to nodes in G . Our certifier takes in G and a certificate and verifies that (1) all nodes are assigned a color, (2) only three colors are used, and (3) no pair of adjacent nodes are assigned the same color. This requires only $O(n^2)$ time.

We will now show that $3\text{-SAT} \leq_p 3\text{-COLORING}$.

Suppose F is 3-SAT formula. Let $F = C_1 \wedge C_2 \wedge \dots \wedge C_m$ over variables x_1, x_2, \dots, x_n . Assume clause $C_j = (t_j^1 \vee t_j^2 \vee t_j^3)$ where each t_j^1, t_j^2 , and t_j^3 are all terms (either a variable x_i or its negation, \bar{x}_i). Construct a graph G as follows

- Add nodes r, g and b , all connected to one another. In any valid coloring, we'll be thinking of the colors assigned to r, g , and b as red, green, and blue respectively.
- For each variable x_i , add nodes t_i and f_i . Connect these to each other and to b . Note that since they connect to b , any valid coloring will assign these green and red or red and green. Node t_i colored green (red) will correspond to x_i being set to true (false).
- For each clause C_j , create a gadget containing 6 nodes: $v_j^1, v_j^2, v_j^3, u_j^1, u_j^2$, and u_j^3 . Connect all u nodes within this gadget to each other. Also connect each node u_j^k to v_j^k . Add an edge from v_j^k to g . Finally, if term k in C_j (i.e. t_j^k) is x_i , add an edge from v_j^k to t_i , and if the term is \bar{x}_i , add an edge from v_j^k to f_i .

For example, the above figure contains a clause gadget associated with the clause $C_1 = (x_1 \vee x_2 \vee \bar{x}_3)$. This concludes the construction, which takes polynomial time (we have $6m + 2n + 3$ vertices).

We now prove that

$$F \text{ is satisfiable} \iff G \text{ is 3-colorable}$$

(\Rightarrow) Suppose that F is satisfiable. That is, there is an assignment A of truth values to all variables in F such that F evaluates to true. We'll use A to generate a 3-coloring c of G . First, color g green, r red, and b blue. For each variable x_i , if $A(x_i)$ is true, color t_i green and f_i red. Otherwise color t_i red and f_i green. Note that, thus far, no two adjacent nodes have been assigned the same color.

For each clause C_j , we know A satisfies at least one term. Pick one of these, say t_j^k . Color v_j^k red and u_j^k blue. Of the remaining v nodes, color them both blue. Of the remaining u nodes, color one red and one green. It is easily verified that no edges within a clause gadget are violated. Since green is never used on a v node, the edges to the g node aren't violated. And since the t_j^k evaluates to true, by our construction of G , v_j^k is only connected to a variable node that is assigned green, and thus coloring it red does not cause a conflict either. Thus G is 3-colorable.

(\Leftarrow) Suppose that G is 3-colorable. Let c be such a coloring, and relabel the colors assigned by c such that the color assigned to r is red, to g is green, and b is blue (since these nodes are all adjacent, they must have been assigned distinct colors by c). Construct a truth assignment A as follows. If $c(t_i) = \text{green}$, set $A(x_i) = \text{true}$. If $c(t_i) = \text{red}$, set $A(x_i) = \text{false}$. Note that t_i can't be assigned blue, since t_i is adjacent to b .

Why does this truth assignment satisfy F ? Consider an arbitrary clause C_j , and examine the corresponding clause gadget of G . A 3-coloring of G must assign a distinct color to each u node in this gadget, and in particular, one node must be assigned blue. Suppose this node is u_j^k . Then v_j^k must not be colored blue. But since this node is also adjacent to g , it must not be colored green either. Therefore v_j^k must be colored red. Thus the variable node v_j^k connects to must be assigned green. But by our construction of G , that means the corresponding term is in this clause, and by our definition of A , this term is satisfied. Therefore C_j is satisfied by A , and thus F is satisfied. \square