How To Write a Proof of NP-completeness

Ran Libeskind-Hadas

November 25, 2013

Proving that a problem is NP-complete involves several parts. First, we must show that the problem is in the class NP. That is, we must show that if the answer to our decision problem is "yes" then we can verify this if the answer (or "certificate") is given to us.

Next, we must show that every problem in NP is polynomial-time reducible to our problem. Generally, we do this indirectly by showing that some known NP-complete problem is polynomial-time reducible to our problem. Let's let L denote our problem. Then, we try to find a known NP-complete problem that is as similar to L as possible because this usually makes for an easier reduction. Sometimes we can find a similar problem for our reduction, but sometimes we are forced to use something generic such as 3SAT.

So, let K denote the known NP-complete problem that we've selected for our reduction. We now must establish that $K \leq_p L$. Be careful here! Showing that $L \leq_p K$ is not at all useful! This is the single biggest pitfall in NP-completeness proofs.

For example, to prove that VERTEX COVER is NP-complete, a reduction **from** 3SAT is useful (i.e. 3SAT $\leq_p$ VERTEX COVER) whereas a reduction **from** VERTEX COVER provides you with no new information, and certainly does not help with proving that VERTEX COVER is NP-complete. That is, showing that VERTEX COVER $\leq_p$ 3SAT is useless to us, since we already know that every problem in NP, and thus VERTEX COVER in particular, is polynomial-time reducible to 3SAT.

The reduction requires several steps. First, we must describe the reduction itself. Then we must show that the reduction takes polynomial time. Finally, we must show that the reduction is correct in the sense that if we start with a "yes" instance of K then the reduction constructs a "yes" instance of L and if we start with a "no" instance of K then the reduction constructs a "no" instance of L. Since dealing with "no" is often awkward, we instead usually show the contrapositive: If the constructed instance of L was "yes" then the original instance of K that we started with was also a "yes" instance.

Writing the NP-completeness proof requires care and precision. Let's demonstrate what's expected by giving the full NP-completeness proof for Vertex Cover that we saw in class.

Consider an instance of Vertex Cover. This comprises an undirected graph with n vertices, m edges, and positive integer k. The decision problem asks whether or not there exists a vertex cover in this graph of size k or less. We begin by showing that Vertex Cover is in NP. If the instance is a "yes" instance then a certificate that demonstrates this (that is, a solution) comprises a set of k or fewer vertices. We can verify that this is a valid solution in time polynomial in the size of the problem (namely $n + m$) by first checking if the given solution uses only vertices that are in the graph, the number of vertices does not exceed k, and that every edge in the graph does indeed have at least one of its two endpoints covered. This clearly takes time polynomial in $n + m$ and thus the problem is in NP. **Editor's Note: Notice the statement "This clearly takes polynomial time ...". This could be shown more rigorously, but it is common practice to just leave it at this since anyone with minimal algorithm background should be able to see this.**

We now establish a reduction from 3SAT to Vertex Cover. Consider an arbitrary instance of 3SAT with n variables and m clauses. We construct a graph G as follows: For each of the n variables we construct a pair of vertices x_i and $\overline{x}_i$ and connect these two vertices with an edge. We call each such vertex pair and its edge a "variable gadget". For each clause in our 3SAT instance, construct three vertices labeled with the literals in that clause and fully connect these three vertices (forming a triangle). We call each such triangle a "clause gadget." Finally, for each vertex in a clause gadget, we connect it with an edge to the corresponding vertex in a variable gadget. To formulate a Vertex Cover decision problem, we must also give a positive integer k where we ask if the graph has a vertex cover of size k or less. We set $k = n + 2m$. **Editor's Note: This completes the description of the reduction.**

Now we must show that this reduction takes time polynomial in the size of the given 3SAT instance. The 3SAT instance has n variables and m clauses. Each variable gadget takes constant time to construct. Each clause gadget takes constant time to construct. Computing the number k takes constant time. Therefore, the entire reduction takes time $O(n + m)$ which is polynomial in the size of the 3SAT instance. **Editor's Note: This completes the argument that the reduction takes polynomial time to compute.**

Next, we establish that the reduction is correct. Consider an arbitrary instance,

I, of 3SAT and its corresponding instance, I', of VERTEX COVER. Assume that the answer to I is "yes". Then there exists a satisfying assignment for this instance. For each variable x_i in I, if it is assigned "true" then include in the vertex cover the vertex labeled x_i in the variable gadget for x_i. Similarly, if x_i is assigned "false" then include in the vertex cover the vertex labeled $\overline{x}_i$ in the variable gadget for x_i. This uses n vertices so far. Next, for each clause C in I we know that at least one of its literals evaluates to "true". Identify one such literal in clause C. Now, in the clause gadget in I' corresponding to clause C, select the vertices corresponding to the other two literals and include them in the vertex cover. This uses exactly 2 vertices per clause gadget for a total of $2m$ additional vertices. In total, $k = n+2m$ vertices have been used. Now, we must establish that the $n+2m$ vertices selected do, in fact, form a vertex cover. There are three types of edges in the graph: Those in the variable gadgets, those in the clause gadgets, and those between a vertex in a clause gadget and a vertex in a variable gadget. The edges in the variable gadgets are all covered since each variable gadget had one of its two endpoints selected. The edges in the clause gadgets are all covered since two vertices in each clause gadget were selected and any two vertices in a clause gadget suffice to cover all three of the edges in that gadget. Finally, consider an edge between a clause gadget and a variable gadget. Assume that the endpoint of that edge in the clause gadget was not selected. This endpoint vertex denotes a literal x_i or $\overline{x}_i$. Consider the first case. The other endpoint of the edge in question is a vertex in a variable gadget also labeled x_i. The literal x_i must have been "true" in our satisfying assignment for I since the vertex labeled x_i in the clause gadget was not selected. Therefore, the edge in question is covered by its endpoint in the variable gadget. A symmetric argument applies to the case of $\overline{x}_i$. **Editor's Note: This completes one direction of the correctness of the reduction. Next, we must show that if the answer to I is "no" then the answer to the constructed instance I' is also "no". We usually do this by using the contrapositive instead: We show that if the answer to I' is "yes" then the answer to I is "yes". Notice that I' is not arbitrary! It was constructed by applying our reduction to I!**

Finally, assume that the answer to the instance I' is "yes". That is, there exists a vertex cover of size $n + 2m$ in this graph. Notice that we must select at least one vertex in each variable gadget and at least two vertices in each clause gadget in order to have a valid vertex cover since each clause gadget comprises two vertices connected by an edge and each clause gadget comprises three vertices in a completely connected graph. Thus, in order to have a vertex cover of size $n+2m$ we must have exactly one vertex in each variable gadget and exactly two vertices in each clause gadget. We now show that this vertex cover induces a valuation for the variables in

I and then show that this valuation is a satisfying valuation. The valuation follows from the fact that each variable gadget has exactly one of its two endpoints in the vertex cover; if x_i is selected in the vertex cover then set variable x_i to "true" in I. Conversely, if $\overline{x}_i$ is selected then set variable x_i to "false". This valuation is well-defined (no conflicts) by our observation above. We now claim that this valuation satisfies the 3SAT instance I. Consider an arbitrary clause C in I. If C is unsatisfied in our valuation, then all of its literals evaluate to false. Examine the clause gadget corresponding to C in I'. This clause gadget has exactly one edge e that emanates from the gadget and is not covered by a vertex in that gadget. Let the endpoint of e in the variable gadget be labeled x_i or $\overline{x}_i$. In the former case, the vertex labeled x_i must have been selected in order to have a valid vertex cover. However, this would imply that the variable x_i was set to "true" and that x_i appears in clause C, contradicting the assumption that C was not satisfied in this valuation. The second case is entirely analogous. Q.E.D