CPSC 320 Sample Solution: The Futility of Laying Pipe, Part 3

First, an apparently unrelated side-track: **How many edges must there be in a tree with n nodes?** So... **if you use at most k edges to form a connected graph, how many nodes can you connect?**

SOLUTION: A tree with n nodes has $n - 1$ edges. One way to figure that out is that each edge connects exactly one node to its parent, and every node but the root has a parent.

Similarly, with k edges to form a connected graph, we can include at most $k + 1$ nodes. (One way to prove that would be by induction on the number of edges.)

Moving on to the rest of the problems...

Now, we'll prove that your reduction is correct. Remember: your reduction is correct exactly when the answer to an instance of 3-SAT is `YES` if and only if the answer to the corresponding instance of SP is `YES`. So, your proof should proceed in two steps:

- Assume the answer to the 3-SAT instance is `YES`. Prove that the answer to the SP instance your reduction creates is also `YES`.
- Assume the answer to the SP instance (which your reduction creates from the 3-SAT instance) is `YES`. Prove that the answer to the corresponding 3-SAT instance is also `YES`.

1 If 3-SAT answer is `YES`, SP answer is `YES`

We'll start by assuming that the answer to the 3-SAT instance is `YES` and proving that the answer to the SP instance your reduction creates is also `YES`.

1. We usually want to go further than the assumption that the original instance's answer is `YES` and say that a solution to the orginal instance (i.e., a working certificate) exists.

   Consider what a solution for 3-SAT looks like and use it to finish the statement: "Since the answer to the 3-SAT instance is `YES`, there must be..."

   SOLUTION: "...some assignment of truth values to the variables that makes at least one literal true in every clause."

2. To prove the underlying instance's answer is `YES`, we similarly try to show the existence of a solution (working certificate) to that instance and conclude that therefore the answer to the instance is `YES`.

   What does a solution for SP look like?

   SOLUTION: The "solution" (beyond just `YES`) to an SP problem looks like a set of at most k edges that connect all the shaded nodes.

3. By assumption (because you have a certificate), you now know the truth value of each variable in the 3-SAT instance. What parts of the solution to (i.e., certificate for) the SP instance follow from these truth assignments?

SOLUTION: We designed our reduction so that one "variable gadget" represents the truth (or falsehood) of each variable. Thus, in each "variable gadget", we'll use two edges to connect the pin to the hub via the positive variable if the variable's value is true or the negated one if the variable's value is false. (This must be possible using $2n$ edges, as argued for the variable gadget above.)

4. By assumption (because your certificate works), every clause in the 3-SAT problem has at least one true literal. What parts of the solution to the SP instance follow from this fact?

   SOLUTION: Pick the first literal that is true in each clause. The clause's node has an edge to that literal's node. We'll add that edge to our SP solution-in-progress. The literal node must already be connected to the variable's hub and pin, since we ensured we always connect in the node corresponding to the variable's truth value. (This must be possible using c additional edges.)

5. Finish the proof to show that the SP certificate actually works, i.e., is a solution to the instance.

   SOLUTION: Above we built a candidate solution using exactly $2n + c$ edges. Does it actually connect all shaded nodes? Every variable's pin node is connected to the hub, as discussed above. Further, every clause is connected to a true literal, which must in turn be connected to the hub given the way we chose the variable gadgets' edges. Thus, our SP "solution" is a valid certificate showing the answer to the SP instance is `YES`.

2 If SP answer is `YES`, 3-SAT answer is `YES`

Now go the other way. Assume that the answer to the SP instance (created by your reduction from the 3-SAT instance) is `YES` and prove that the answer to the 3-SAT instance is also `YES`.

Hints: Many pieces of this proof will be very similar to the previous one, but you'll also need to show that any solution (working certificate) for the SP instance has its edges where you intended them to be when you designed your reduction. It may help to think about the number of nodes you can possibly connect, given a maximum of k edges.

SOLUTION: Since the SP instance's solution is `YES`, at most $2n + c + 1$ nodes must be connected by a set of at most $2n + c$ edges in the certificate for that solution.

There are $n + c + 1$ shaded nodes that **must** be connected; so, n of the unshaded nodes can also be connected. For every variable gadget, at least one of its "variable" nodes (positive or negated) **must** be connected (because there's no other way "out" of the variable's pin, which is why we chose to call it a "pin"). Since this consumes all remaining available nodes, we must choose exactly one of each positive/negated variable pair. Furthermore, each clause must be connected to exactly one of its literals, or we'd either introduce a cycle (connecting to two already-connected literals and thus using an unavailable extra edge) or an extra node (to a so-far-unconnected literal and again using an unavailable extra edge).

We can choose each variable in 3-SAT to have a value corresponding to its variable gadget's connected node (i.e., true if the positive node is connected and false otherwise). This must make for at least one true literal in each clause, since each clause's node is connected to one of its literals' nodes that is itself connected. Thus, this is a satisfying assignment in the 3-SAT instance, and the instance's answer is `YES`.

You've now shown that SP is in NP and is NP-hard (because 3-SAT—which is known to be NP-hard—is polynomial-time-reducible to SP). Therefore, SP is NP-complete!

3 Challenge (with two that EVERYONE should try)

We leave all challenges but the first two to you.

1. Assume we use an adjacency list representation for the graph in our SP instance. Adding a node takes constant time. We add a total of one hub node, $3n$ variable nodes (positive, negative, and pin), and c clause nodes. That's clearly linear in $n + c$.

Adding an edge also takes constant time. We add four edges per variable (which takes $O(n)$ time) and three edges per clause (which takes $O(c)$ time).

Finally, we compute the value of k in constant time.

2. Prove that weighted SP is NP-complete. (Everyone should complete this challenge.)

   This is a **much** more straightforward reduction. We should use unweighted SP (USP) in a reduction with weighted SP (WSP). We'll give you some questions to work through.

   First, which direction is the reduction (USP to WSP or WSP to USP)?

   Second, what is the reduction? (It should be **very** simple.)

   Third, prove the reduction correct. (Again, this should be **very** simple.)

   (Your reduction should be so simple, that proving it takes polynomial time is trivial!)