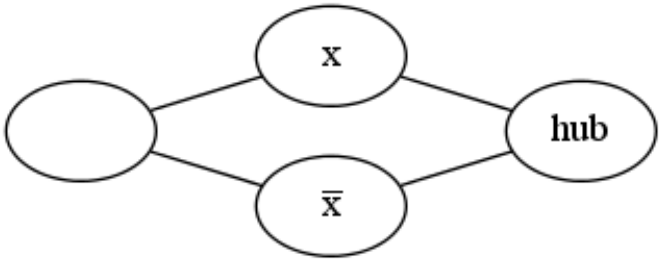
SP is NP-Complete

November 19, 2018

We've already shown that SP $\in$ NP. To show it's NP-complete, we need to also show that it is NP-hard; that is, that it's at least as hard as every other problem in NP. We'll use 3-SAT to help us, since we already know that 3-SAT is at least as hard as every other problem in NP.

1. Which of these would show that SP is at least as hard as 3-SAT: reducing from SP to 3-SAT in polynomial time or reducing from 3-SAT to SP in polynomial time? (Hint: We **know** that any problem in NP can be reduced to 3-SAT in polynomial time. We want to **prove** that any problem in NP can be reduced to SP in polynomial time.)

2. Here is a sketch of a "variable gadget" to help with our reduction. How can we "shade in" (put in S) some of these vertices and choose an appropriate k (maximum number of edges in the solution) to enforce 3-SAT's choice that x or $\overline{x}$ is true but not both?



3. Draw a graph with four variable gadgets (w, x, y, and z), all sharing a single "hub" node. Make sure your layout still enforces choosing either true or false but not both for each of the four variables, yet allows all 16 possible combinations of their truth values.

4. Now, find a way to add one or more nodes and edges to your graph and choose a k in order to represent the clause $(\overline{w} \vee \overline{x} \vee y)$ and enforce that: at least one of $\overline{w}$, $\overline{x}$, and y is true and also (still) each variable is either true or false but not both. (z isn't in this clause, which is fine. In most 3-SAT problems, not all variables are in all clauses.)

5. Give a complete reduction from 3-SAT to SP such that the answer to the SP instance you produce is `YES` if and only if the answer to the original 3-SAT instance is `YES`.

6. Analyze the runtime of your reduction (to show that it takes polynomial time).