# CPSC 320 Sample Solution: The Futility of Laying Pipe, Part 1

March 19, 2017

UBC recently replaced its aging steam heating system with a new hot water system. A set of locations needs water delivered and there's another set of intermediate points through which we can deliver water. Some of these points can be connected—at varying costs—by laying new pipe, others cannot. You'd like to figure out the cheapest way to connect every delivery location to water.

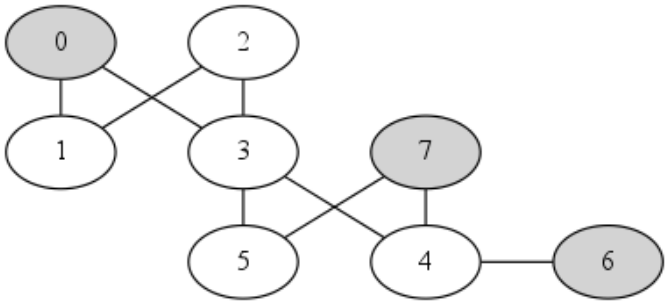
## 1 Steiner. . . Something-or-Others

Let's abstract and formalize this problem. We'll call it the Steiner Problem (SP).

An instance of SP is an undirected graph $G = (V, E)$ and a subset $S \subseteq V$ of the vertices to which we must deliver water. A solution to the instance is a subset $E' \subseteq E$ of the edges which connects all vertices in $S$ (and perhaps some in $V$). The **best** solution is the one with the fewest edges. (We could make this into an "input file format", e.g., by reading a number $n$ indicating $|V|$ followed by $n$ lines, where line $i$ lists the vertex numbers of all vertices connected to $i$, etc.)

(Although we've ignored the costs, we could easily have included them by making the edges weighted.)

1. Here's a SP instance, where shaded nodes are in $S$. Indicate a solution to this problem.




**SOLUTION:** We'd use the edges: $(0, 3), (3, 4), (4, 7), (4, 6)$ to produce the minimum Steiner Tree. Note that this is a tree but **not** a spanning tree!

2. Build three trivial SP instances with their solutions.

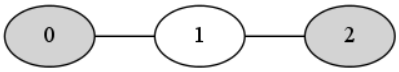
**SOLUTION:** Some trivial examples include:

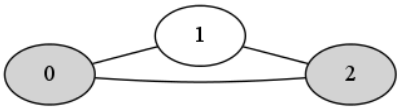
- the empty graph (solution is the empty set of edges),
- any graph of any size with one or zero shaded vertices (solution is also the empty set of edges), and
- much less trivially: any graph where all vertices are shaded (solution is any spanning tree, which we can find with DFS or BFS in polynomial time).

We should probably assume that the graph is connected, but any disconnected graph where two or more connected components contain a shaded vertex is a different, somewhat trivial example. In those cases, there is no solution.

3. Build two small but non-trivial SP instances with solutions.

   **SOLUTION:** Here are two small examples that don't fit in our trivial categories above yet remain simple. The first shows that we may end up with a spanning tree, while the second shows that we need to be ready **not** to (since its solution is the one edge connecting nodes 0 and 2).




4. Give at least two problems similar to this one that have we solved before.

   **SOLUTION:** There are problems that we've solved that are similar in various ways, mostly in the sense that they work with graphs (e.g., shortest path which worries about costs of paths or vertex cover, which worries about "coloring" particular nodes to cover the edges).

   However, the problem with what seems the greatest similarity is the minimum spanning tree problem. We can solve that in polynomial time (quite efficiently). Any spanning tree of the graph will connect **all** nodes in the graph and therefore solve the SP instance, but it may not be the optimal solution, since it may waste edges connecting unnecessary white nodes. We could "trim off leaf white nodes" repeatedly until there are none left, but there's still no guarantee that we'll get an optimal solution.

   We can imagine extending SP to a weighted version, in which case MST feels even more similar.

5. Now, think about what a solution to SP looks like. Is it a path? A cycle? A tree? A graph?

   **SOLUTION:** It's a tree. (It is also, of course, a graph, since trees are graphs. It can be a path but need not necessarily be one, and it cannot be a cycle. That is, the **best** solution cannot be a cycle.)

6. Once you've figured that out, give a **very** similar problem we've solved before in **polynomial time**. Can we just use the solution we used for that problem? If you wanted to try using the solution to that problem, how would you modify it when reporting a solution to the SP instance?

   **SOLUTION:** See above. MST and no.

7. Describe how to turn SP into a decision problem (one where the answer is `YES` or `NO`). (Remember how we did this for, e.g., independent set (IS). The original version of IS was "given a graph, find the largest independent set". The decision version was "given a graph and a number $k$, is there an independent set at least as large as $k$?". Do the same sort of transformation to SP.)

   **SOLUTION:** Much like independent set or vertex cover, we can add a parameter to the problem (besides $G$ and $S$, the set of "shaded" vertices) to turn it into a decision problem.

   The standard way to do this is to add the non-negative integer parameter $k$ and ask whether there is a tree $T$ that is a subgraph of $G$ with at most $k$ edges (or, equivalently, $k+1$ nodes) that includes all of the nodes in $S$.

8. Prove that the decision problem is in NP. Remember: it's in NP if it's "efficiently certifiable". The "certificate" is usually what we'd think of as the solution to the non-decision problem.

   **SOLUTION:** This decision problem is in NP if we use polynomial-length evidence (the certificate) consisting of the list of edges in a satisfactory Steiner Tree. In polynomial time, we can check that (1) the list has no more than $k$ edges and (2) the edges form a single tree that touches each shaded vertex. (One way to handle (2) would be to DFS the subgraph of $G$ formed by deleting all edges but the ones in the certificate. During the DFS, we "check off" each shaded node as we reach it. If we fail to reach any, then this certificate does not show that the problem's solution is `YES`.)