Section 6.2. Minimal Spanning Tree Algorithm.

Note. In this section, we consider a graph (i.e., a network with undirected arcs in Taha's terminology) which is connected and has a weight on each edge. We describe an algorithm that finds a spanning tree of minimal total edge weight. We say "minimal" because there may be other spanning trees of the same total eight, but there is no other spanning tree of smaller total weight. We follow Taha's presentation, but also give a clear statement of the algorithm and give references that justify the claims of minimality.

Note. Let $N = \{1, 2, \ldots, n\}$ be the set of nodes of the network. Define C_k denote the set of nodes that have been permanently connect at iteration k of the algorithm, and let $\overline{C}_k$ denote the set of nodes not yet connected permanently immediately after iteration k of the algorithm. The minimal spanning tree algorithm then has the steps:

Step 1. Set $C_0 = \varnothing$ and $\overline{C}_0 = N$.

Step 2. Start with any node i in the unconnected set $\overline{C}_0$ and set $C_1 = \{i\}$, and set $\overline{C}_i = N \setminus \{i\}$. Set $k = 2$.

General Step k. Select a node j^* in the unconnected set $\overline{C}_{k-1}$ that yields the least weight on a edge (or an "arc") to a node in the connected set C_{k-1}. Link j^* permanently to C_{k-1} and remove it from $\overline{C}_{k-1}$ to obtain C_k and $\overline{C}_k$, respectively. Stop if $\overline{C}_k$ is empty, otherwise replace k with $k + 1$ and repeat this step.

Example 6.2.1. Midwest TV Cable Company needs to connect cable to five houses. Figure 6.5 shows the distances between the five houses along the routes of possible connections. The goal is to determine the most economical cable network.

Solution. We start with $C_1 = \{1\}$ and $\overline{C}_1 = \{2, 3, 4, 5, 6\}$. The algorithm is illustrated in Figure 6.6. We select node $j^* = 2$ since it is the node in $\overline{C}_1 = \{2, 3, 4, 5, 6\}$ of minimum distance from a node in $C_1 = \{1\}$; then 1 is linked to 2, $C_2 = \{1, 2\}$ and $\overline{C}_2 = \{3, 4, 5, 6\}$ $(k = 1)$. Next, node $j^* = 5$ since it is the node in $\overline{C}_2 = \{3, 4, 5, 6\}$ of minimum distance from a node in $C_2 = \{1, 2\}$; then 2 is linked to 5, $C_3 = \{1, 2, 5\}$ and $\overline{C}_3 = \{3, 4, 6\}$ $(k = 2)$. Node $j^* = 4$ since it is the node in $\overline{C}_3 = \{3, 4, 6\}$ of minimum distance from a node in $C_3 = \{1, 2, 5\}$; then 2 is linked to 4, $C_4 = \{1, 2, 4, 5\}$ and $\overline{C}_4 = \{3, 6\}$ $(k = 3)$.

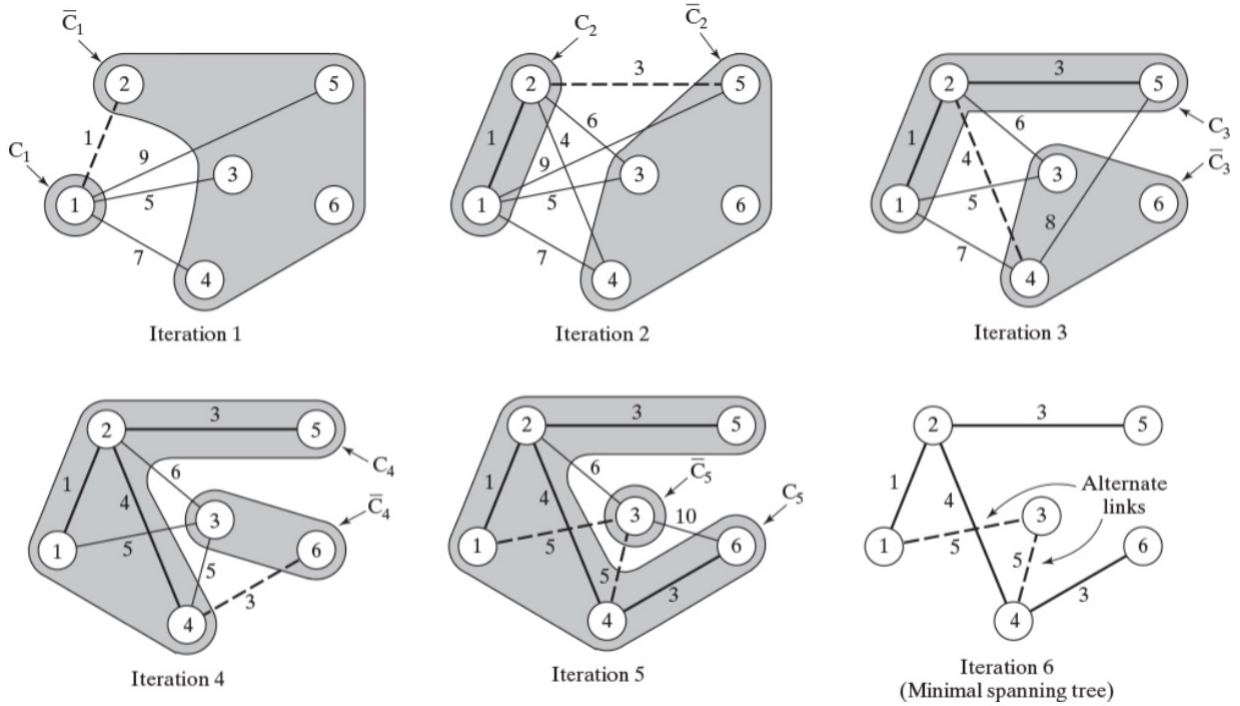

Figure 6.6

Node $j^* = 6$ since it is the node in $\overline{C}_4 = \{3, 6\}$ of minimum distance from a node

in $C_4 = \{1, 2, 4, 5\}$; then 4 is linked to 6, $C_5 = \{1, 2, 4, 5\}$ and $\overline{C}_5 = \{3\}$ $(k = 4)$. Finally, we take $j^* = 3$ because it is the only element of $\overline{C}_5 = \{3\}$; then we link 3 to either 1 or 4 (since both of these yield the minimum distance $(k = 5)$. Then $C_6 = \varnothing$ and the algorithm ends. This results in a minimal of minimal total length $1 + 3 + 4 + 3 + 5 = 16$. We see by this example (in step $k = 5$), that the minimal tree is not unique.

Note. The algorithm described is called Kruskal's algorithm or the Borůvka-Kruskal Algorithm. It was first used by Czech scientist Otakar Borůvka in 1926 to find an electrical network in Moravia, Czech Republic. Otakar Borůvka published his result in Czech. Independently, Joseph B. Kruskal, Jr. found the same algorithm and published his result in "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," *Proceedings of the American Mathematical Society* **7**, 48–50 (1956). A copy of Kruskal's paper is online on the American Mathematical Society webpage (accessed 5/3/2022). The algorithm is described in Introduction to Graph Theory (MATH 4347/5347) in Section 7.1. Spanning Tree Algorithms. It is described in detail in Mathematical Modeling Using Graph Theory (MATH 5870) in Section 8.5. Greedy Heuristics, where a proof that it yields a minimal weight spanning tree is to be given in Exercise 8.5.3 (see also Theorem 8.23). The algorithm is state as:

INPUT: a weighted connected graph $G = G(G, w)$

OUTPUT: an optimal tree $T = (V, F)$ of G, and its weight $w(F)$

1: set $F = \varnothing$, $w(F) = 0$ *(*F *denotes the edge set of the current forest)*

2: **while** *there is an edge* $e \in E \setminus F$ *such that* $F \cup \{e\}$ *is the edge set*

of a forest **do**

3: *choose such an edge e of minimum weight*

4: *replace F by $F \cup \{e\}$ and $e(F)$ by $w(F) + w(e)$*

5: **end while**

6: *return* $(V, F), w(F))$

Notice that this is the same as the algorithm given by Taha. The do/while loop given here is the same as the "General Step k" given above.

Revised: 5/30/2022