A fundamental theorem about minimum spanning trees

Jayadev Misra

May 12, 2020

Abstract

We present a theorem that underlies all algorithms for minimum spanning tree construction.

Keywords: Graph algorithms, minimum spanning tree, safe edge.

1 Background

A *spanning tree* of a connected undirected graph is a subset of its edges that forms a tree over the nodes of the graph. Given real-valued edge lengths, possibly negative, the length of a spanning tree is the sum of its edge lengths. A spanning tree of minimum length is called a *minimum spanning tree*, henceforth abbreviated as *mst*.

Safe edge A given set of edges M partition the nodes of a graph into connected components, where two nodes are in the same component iff there is a path between them that uses only the edges from M. Edge x—y, where x and y belong to distinct components X and Y respectively, is an *inter-component* edge with respect to M. Edge x—y is *safe* for X wrt M if it is a shortest inter-component edge incident on X. Write $X \xrightarrow{e}_M Y$ to denote that edge e is incident on nodes in components X and Y, and e is safe for X under the partition induced by M; we drop the subscript M when it is clear from the context. Note that e may not be safe for Y. A component may have several safe edges, and they are of equal length by definition. Call edge e safe wrt (with respect to) M if $X \xrightarrow{e}_M Y$ for *some* component X.

Notation $\|e\|$ is the length of edge e and $\|M\|$ the sum of lengths of the edges in M. □

The following lemma proves an independence result among safe edges.

Lemma 1 Let edges e and f of distinct lengths be safe wrt M. Then f is safe wrt $M \cup \{e\}$.

Proof: Suppose $X \xrightarrow{e}_M Y$. In $M \cup \{e\}$ components X and Y are merged to form a single component, call it XY. Suppose f is safe for component Z. Consider the following cases for Z.

• $Z \notin \{X, Y\}$: The components wrt $M \cup \{e\}$ and M are the same except that X and Y are combined to form a component in the former case. So, Z and its inter-component edges are unaffected wrt $M \cup \{e\}$, so f is a safe edge for Z wrt $M \cup \{e\}$.

• $Z = X$: Both e and f are safe for X, so they have the same length, contradicting that e and f have distinct lengths.

• $Z = Y$: If f is incident on X then both e and f are incident on both X and Y. Since e is safe for X, $\|e\| < \|f\|$, and f safe for Y implies $\|f\| < \|e\|$, contradiction.

Therefore, f is incident on Y and a component other than X. So, f is an inter-component edge of XY. We show that f is safe for component XY. Write $IE(X)$ and $IE(Y)$ for the inter-component edges of X and Y, respectively, wrt M, and $IE(XY)$ for the inter-component edges of XY wrt $M \cup \{e\}$. Then $IE(XY) \subseteq IE(X) \cup IE(Y)$.

$$
\begin{array}{ll}
 & \mathit{true} \\
\Rightarrow & \{e \text{ and } f \text{ are safe for } X \text{ and } Y, \text{ respectively}\} \\
 & \quad \|e\| \leq \min\{\|f\| \mid f \in IE(X)\} \wedge \ \|f\| \leq \min\{\|f\| \mid f \in IE(Y)\} \\
\Rightarrow & \{e \text{ is incident on } Y \text{ and } f \text{ safe for } Y. \text{ So, } \|f\| < \|e\|\} \\
 & \quad \|f\| \leq \min\{\|f\| \mid f \in IE(X)\} \wedge \ \|f\| \leq \min\{\|f\| \mid f \in IE(Y)\} \\
\Rightarrow & \{IE(XY) \subseteq IE(X) \cup IE(Y)\} \\
 & \quad \|f\| \leq \min\{\|f\| \mid f \in IE(XY)\} \\
\Rightarrow & \{f \text{ is an inter-component edge of } XY\} \\
 & \quad f \text{ is safe for } XY
\end{array}
$$

□

2 A fundamental theorem of minimum spanning tree

Theorem 1 Let M be a subset of some mst and E a set of safe edges of distinct lengths. Then $M \cup E$ is a subset of some mst.

Proof: The proof is by induction on the size of E.

• E is empty: $M \cup E = M$ is a subset of some mst.

• E has exactly one element e: Suppose M is a subset of mst M'. If $e \in M'$, then $M \cup \{e\} \subseteq M'$, so the proof is complete. Next, suppose $e \notin M'$ and e is a safe edge for component X. A property of a spanning tree is that $M' \cup \{e\}$ has a cycle. This cycle includes another edge f where $f \in M'$, and f is incident on X. Then $M'' = (M - \{f\}) \cup \{e\}$ is a spanning tree. We show that M'' is a mst because $\|M''\| \leq \|M'\|$.

Since e is a safe edge for X and f an inter-component edge of X, $\|e\| \leq \|f\|$. So, $\|M''\| = \|M'\| - \|f\| + \|e\| \leq \|M'\|$.

• E has more than one element: Let $e \in E$. Then $M \cup \{e\}$ is a subset of some mst, from the case proved above. From Lemma 1 (page 1), all edges of $E - \{e\}$ are safe wrt $M \cup \{e\}$. Inductively, $(M \cup \{e\}) \cup (E - \{e\}) = M \cup E$ is a subset of some mst. □

On distinct edge lengths Theorem 1 requires that all edge lengths in E be distinct. To see the necessity of this condition consider a graph of 3 nodes that are pair-wise connected by edges of equal length. Given that each node is a component, each edge is safe. If E includes all the edges, the constructed mst has a cycle. The requirement of distinct edge lengths in E avoids this scenario.

The condition of distinct edge lengths is clearly met if E is a singleton. This is the property exploited in Kruskal's algorithm in Section 3.1 (page 3) and Dijkstra-Prim algorithm in Section 3.2 (page 4). Borůvka's algorithm, Section 3.3, is applied on graphs in which all edge lengths are distinct; so, the condition of the theorem is trivially met.

3 Algorithms for mst

The abstract algorithm, below, works as follows. Start with an empty set for M which is a subset any mst. Then every node is a component and all edges are inter-component. Identify the safe edges incident on each node, pick some subset E of these edges that have distinct lengths and add them to M. Repeat this step until M is a spanning tree; then M is guaranteed to be a mst from the fundamental theorem.

Abstract algorithm for minimum spanning tree

$M := \{\}$**;**
while M is not a spanning tree **do**
  {$(\exists M' : M' \text{ is a mst} : M \subseteq M')$**}**
  choose E to be a set of safe edges of distinct lengths**;**
  $M := M \cup E$
  {$(\exists M' : M' \text{ is a mst} : M \subseteq M')$**}**
enddo
{ $(\exists M' : M' \text{ is a mst} : M \subseteq M')$, M is a spanning tree **}**
{ M is a mst **}**

The abstract algorithm permits a number of choices for E. This allows for the development of a family of programs. Performance of different programs depend on the data structures and algorithms they employ for computing a set of safe edges of distinct lengths.

3.1 Kruskal's algorithm

Kruskal's algorithm [2] starts with an empty set for M. It scans the edges in order of increasing lengths where edges of equal length are scanned in arbitrary order. For each scanned edge e if e is safe, it is added to M, otherwise it is discarded from further consideration. This strategy meets the requirement of Theorem 1 (page 2) because any edge that is added is safe, and it has distinct length by itself. The steps continue until M is a spanning tree.

3.2 Dijkstra-Prim algorithm

This algorithm was discovered independently by Dijkstra [1] and Prim [3]. At all points during the algorithm there is a *primary* component (let us call it *prim*; the coincidence in naming is entirely intended) that may consist of one or more nodes, and remaining components that have exactly one node each. Initially every node is a component by itself, and one of the components is chosen arbitrarily as the primary component. At all points M is a mst over the nodes of the primary component. In each step a shortest inter-component edge incident on *prim* is added to M until M has $n - 1$ edges. The correctness of the algorithm is immediate from Theorem 1 (page 2).

3.3 Borůvka's algorithm

Perhaps the first mst algorithm was designed by Borůvka in 1926. The algorithm has been discovered and rediscovered several times. The algorithm is applied on graphs in which edge lengths are all distinct. It chooses E to include all safe edges in each step. It is a greedy algorithm in the sense that it does all that is possible in a step.

Every step reduces the number of components by at least half, usually more, so the number of steps is bounded by $\log_2 n$. Each step, of course, takes longer than joining just two components.

The algorithm permits efficient parallel implementation. Large graphs can be processed by multiple computers simultaneously because it is sufficient to find a safe edge of a component independent of other components. This parallel version of the algorithm was popularized by Sollin [4].

References

[1] E.W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:83–89, 1959.

[2] Joseph. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, Feb 1956.

[3] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36(6):1389–1401, November 1957.

[4] Georges Sollin. "le trac de canalisation". *Programming, Games, and Transportation Networks*, 1965.