Minimum Spanning Tree
Jayadev Misra
12/12/98

1 Spanning Tree

A *spanning tree* of an undirected graph of n nodes is a set of $n-1$ edges that connects all nodes. This note develops two algorithms for finding the minimum spanning tree.

Properties of spanning trees In a spanning tree:

- There is no cycle: a cycle needs n edges.
- There is exactly one path between any two nodes: there is at least one path between any two nodes because all nodes are connected. Further, there is not more than one path between a pair of nodes because then there would be a cycle that includes both nodes.
- Adding a non-tree edge creates a cycle: Suppose a non-tree edge (x, y) is added to a spanning tree. Now there are two distinct paths between (x, y), the added edge and the path in the tree. Hence there is a cycle.
- Removing an edge from a cycle as above creates a spanning tree: after removal of the edge there are $(n-1)$ edges. All nodes of the graph are connected: suppose edge (x, y) is removed that belonged to the original graph. The nodes x, y are still connected because x, y were on a cycle. For other node pairs, in the path in the original graph replace the edge (x, y) by the path between x, y.

Consider a graph whose edges have non-negative weights. A *minimum spanning tree* is a spanning tree whose total edge weight is minimum. Heneceforth, we assume that all edge weights are distinct.

Let M be a minimum spanning tree and e be an edge such that $e \notin M$. Let C be the subset of M such that $C \cup e$ is a cycle (C exists, from the property of spanning trees).

Lemma 1 All edges in C have lower weight than e.

Proof:: On the contrary, if C contains an edge, f, of higher weight then replacing f by e in the spanning tree creates a tree of lower weight.

2 Kruskal's Algorithm

Let $e_0, .., e_m$ be the sequence of edges in order of increasing weight and $E_i = \{e_j |\ j < i\}$. Let n be the number of nodes in the graph. Suppose M is the minimum spanning tree. This algorithm computes $T_0, T_1, ..,$ where $T_i = M \cap E_i$.

- initially: $E_0 = \phi$. Hence, $T_0 = \phi$.
- Whenever T_j has $n-1$ edges, $T_j = M$:

$$\begin{array}{ll} & |T_j| = n-1 \\ \Rightarrow & \{T_j = M \cap E_j,\ |M| = n-1\} \\ & T_j \subseteq M \wedge\ |T_j| = n-1 \wedge\ |M| = n-1 \\ \Rightarrow & \{\text{Set Theory}\} \\ & T_j = M \end{array}$$

- $$\begin{array}{l} T_{i+1} \\ = M \cap E_{i+1} \\ = M \cap (E_i \cup e_i) \\ = (M \cap E_i) \cup (M \cap \{e_i\}) \\ = T_i \cup (M \cap \{e_i\}) \\ = \left\{ \begin{array}{lll} T_i & \text{if} & e_i \notin M \\ T_i \cup \{e_i\} & \text{if} & e_i \in M \end{array} \right. \end{array}$$

The following theorem shows how to determine if $e_i \in M$.

Theorem $(e_i \notin M) \equiv (T_i \cup \{e_i\})$ has a cycle.
Proof:: Proof is by mutual implication.

- $(e_i \notin M) \Rightarrow (T_i \cup \{e_i\})$ has a cycle:

$$\begin{array}{ll} & e_i \notin M \\ \Rightarrow & \{C \cup \{e_i\} \text{ is a cycle where } C \subseteq M.\} \\ & C \subseteq M \text{ and } C \cup \{e_i\} \text{ is a cycle} \\ \Rightarrow & \{\text{Lemma 1: } C\text{'s edges are lighter than } e_i, \text{ i.e., } C \subseteq E_i.\} \\ & C \subseteq E_i \wedge C \subseteq M \\ \Rightarrow & \{C \subseteq (M \cap E_i).\ T_i = M \cap E_i.\} \\ & C \subseteq T_i \\ \Rightarrow & \{C \cup E_i \text{ has a cycle}\} \\ & T \cup E_i \text{ has a cycle} \end{array}$$

- $(T_i \cup \{e_i\})$ has a cycle $\Rightarrow e_i \notin M$

$$\begin{array}{ll} & T \cup E_i \text{ has a cycle} \\ \Rightarrow & \{T_i \subseteq M, \text{ from } T_i = M \cap E_i\} \\ & M \cup E_i \text{ has a cycle} \\ \Rightarrow & \{M \text{ is a spanning tree}\} \\ & e_i \notin M \end{array}$$

The complete algorithm is given below.

initially $T := \{\}$; $i := 0$
while $|T| \neq n-1$ do
  if e_i does not form a cycle with T
  then $T := T \cup \{e_i\}$

endif;
$i := i + 1$
od

Note: Cycle detection in Kruskal's algorithm can use the Union-Find Algorithm.

3 Prim/Dijkstra Algorithm

Let U, V be two non-empty sets of vertices into which all vertices have been partitioned. Let T be a set of edges, $T \sqsubseteq M$. The following algorithm constructs a minimum spanning tree.

initially $T := \{\}$; $U :=$ some node
while $|T| \neq n - 1$ do
let (x, y) be the edge of the lowest weight where $x \in U$, $y \in V$;
add (x, y) to T and y to U
od

It is sufficient to show that $T \sqsubseteq M$. When T has $n - 1$ edges $T = M$, because M has $n - 1$ edges. Initially, $T \sqsubseteq M$. Next, we show that edge (x, y) is in M. If not, add (x, y) to M creating a cycle. Label the nodes along this cycle U or V depending on which subset they belong to. Since there is an adjacent UV pair (corresponding to edge (x, y)), there is another UV pair in the cycle. That edge has weight higher than that of (x, y), contradicting that M is the minimum spanning tree.