Kruskal's Algorithm Generates a Minimum Weight Spanning Tree

This is a reworkup of the proof I gave in class to show that Kruskal’s algorithm actually produces a minimum weight spanning tree, filling in a little more detail than I gave in class.

Recall that the input to the algorithm is a connected, weighted graph $G$. The algorithm starts with $G$ and a set $M$ of marked edges that is initialized to $\emptyset$ (so no edges are marked at the beginning). At each step, the algorithm chooses an unmarked edge of smallest possible weight that doesn’t form a cycle with the marked edges and marks that edge. It stops if no such edge exists. At that stage, we claim that the subgraph $H$ whose edges are the marked ones (and whose vertices are the end vertices of those edges) forms a minimum weight spanning tree for $G$. (This isn’t quite what the book says; the book says the algorithm stops when the marked edges form a spanning tree. But we don’t know that the algorithm always gets to a spanning tree, so that’s the first thing to prove.)

At each stage of the algorithm, the marked edges form an acyclic subgraph, because we only mark edges that don’t form cycles with the previously marked edges. The subgraph formed by the marked edges might not be connected, but each of its connected components will certainly be acyclic. That means that, at each stage of the algorithm, the subgraph is a forest.

Note that the algorithm stops when adding any of the unmarked edges would produce a cycle. First, suppose that there is a vertex $v_0$ of $G$ that is not a vertex of $H$ (the subgraph formed by the marked edges when the algorithm stops), then we haven’t marked any edges that have $v_0$ as an end vertex. But $G$ is connected, so there must be at least one edge in $e$ of $G$ that has $v_0$ as an end vertex. For an edge to add a cycle to $H$, it must be the case that both of its end vertices are already in $H$ (in fact, in the same connected component of $H$), so $e$ could not add a cycle to $H$. This means that the algorithm could mark another edge, which contradicts our assumption that $H$ is the subgraph produced when the algorithm stops.

Otherwise, suppose that $H$ has all the vertices of $G$ but isn’t connected. Since $G$ is connected, there must be an unmarked edge whose end vertices are in different connected components of $H$. Adding that edge to the set of marked edges can’t create a cycle, so again the algorithm could mark another edge, contradicting our assumption that the algorithm stopped.

So we’ve shown that when Kruskal’s algorithm stops, the subgraph consisting of the marked edges is a connected acyclic graph that includes all the vertices of $G$. That means it’s a spanning tree. (It’s possible to say this in a more compact way that shows that the graph is connected and includes all the vertices in only one step, but I think this way of presenting it is a little easier to follow.)

So we know that the subgraph $H$ produced by Kruskal’s algorithm is a spanning tree and we have to show that it has minimum weight. Let $n$ be the order of $G$ and let the edge marked by the algorithm on step $i$ be $e_i$; then the edges of our spanning tree are $e_1, \ldots, e_{n-1}$ and the weight of our spanning tree is $\sum_{i=1}^{n-1} w(e_i)$.

If $H$ is not a minimum weight spanning tree, let $T$ be a minimum weight spanning tree that has the largest possible number of edges in common with $H$. Since $H$ is not of minimum weight and both $H$ and $T$ have $n-1$ edges, there must be some edge of $H$ that is not in $T$. Let $i$ be the smallest index such that $e_i$ is not in $T$. Since $T$ is a spanning tree, the graph $T + e_i$ formed by adding the edge $e_i$ to $T$ will have a cycle $C$ containing $e_i$. Moreover, since $H$ is acyclic, some edge $e$ in $C$ must
not be in $H$. If we delete the edge $e$ from $T + e$, the new graph, $T + e_i - w$ will be a spanning tree for $G$ whose weight will be $w(T) + w(e_i) - w(e)$.

Now, $T$ is a minimum weight spanning tree, so we must have $w(T) \leq w(T + e_i - e) = w(T) + w(e_i) - w(e)$. That means that $w(e_i) \geq w(e)$. If $i = 1$, then $w(e_i)$ is the smallest weight of all edges of $G$; if $i > 1$, then the weight of $e_i$ is the smallest among those edges that can be added to $e_1, e_2, \ldots, e_{i-1}$ without forming a cycle. Since $e_1, \ldots, e_{i-1}$ and $e$ are edges of the acyclic graph $T$, we could have added $e$ to $e_1, \ldots, e_{i-1}$ without forming a cycle. So $w(e_i) \leq w(e)$. So we’ve shown that $w(e_i) = w(e)$.

But then $w(T + e_i - e) = w(T)$, so $T - e_i + e$ is also a spanning tree of minimum weight. And $T - e_i + e$ has strictly more edges in common with $H$ than $T$ does. This contradicts our choice of $T$ as a minimum weight spanning tree that has the largest possible number of edges in common with $H$. So $H$ must have minimum weight.