1. Suppose we start with a graph that has n vertices and no edges. This graph contains n connected components. If we add an edge (u, v) to this graph then either 1) u and v are in different connected components and the addition of (u, v) causes the number of connected components to decrease by one; or 2) u and v are in the same connected component and the number of connected components stays the same. For the graph to be connected after the addition of $n-1$ edges, every edge must have connected two different connected components. For the graph to have a cycle, some edge must have connected two vertices in the same connected component, a contradiction.

2. Suppose two minimum spanning trees have a different number of edges of some weight. Choose such a pair of trees that have the most edges in common. Let b be the weight where one tree, T, has more edges of weight b than the other tree R. Let (u, v) be an edge in T with weight b that is not in R. Add (u, v) to R. This creates a cycle C. All the edges on cycle C have weight at most b (otherwise we could remove the heavy edge on C from $R \cup \{(u, v)\}$ and produce a spanning tree of smaller weight). Since T is a minimum spanning tree, there must be at least one edge (x, y) of weight b in C that is not in T. Otherwise we could remove the edge (u, v) from T, splitting T into two connected components one of which contains u and the other v and then reconnect the components by a lighter weight edge from the cycle C, creating a spanning tree of smaller weight. Removing (x, y) from $R \cup \{(u, v)\}$ creates a spanning tree with the same weight as R but with one more edge in common with T, a contradiction.

   If all the edge weights are distinct then every minimum spanning tree is the same, for if two minimum spanning trees differ, then one contains an edge (of, say, weight b) that the other doesn't. Since no other edge has weight b, the trees differ in the number of edges of weight b, contradicting the previous claim.

3. (a) Define the *minimum spanning forest* of an undirected graph G, $\mathrm{MSF}(G)$, to be the union of the minimum spanning trees of its connected components. If G is connected then its minimum spanning forest is the same as its minimum spanning tree. Define the *decomposition forest* of an undirected graph G, $\mathrm{DF}(G)$, to be the union of the decomposition trees of its connected components. If G is connected then its decomposition forest is the same as its decomposition tree. For the input graph G, let $G_i = (V, E_i)$ where E_i is the set of the i smallest weight edges. We claim that, for all values of i, $\mathrm{DF}(G_i) = \mathrm{DF}(\mathrm{MSF}(G_i))$. We prove the claim by induction on i. As a base case, the graph G_0 has no edges and n connected components, so $\mathrm{MSF}(G_0) = G_0$ and $\mathrm{DF}(G_0) = \mathrm{DF}(\mathrm{MSF}(G_0))$. Suppose the claim is true for G_{i-1}. The graph G_i is the same as G_{i-1} except for the addition of one edge. Let's call that edge (u, v).

   If u and v are in the same connected component of G_{i-1} then $\mathrm{MSF}(G_i) = \mathrm{MSF}(G_{i-1})$ (because (u, v) is the maximum weight edge in G_i). In addition, $\mathrm{DF}(G_i) = \mathrm{DF}(G_{i-1})$ since the addition of (u, v) does not create a new connected component in G_i. So $\mathrm{DF}(G_i) = \mathrm{DF}(G_{i-1}) =$ (by induction hypothesis) $\mathrm{DF}(\mathrm{MSF}(G_{i-1})) = \mathrm{DF}(\mathrm{MSF}(G_i))$.

   If u and v join two connected components, A and B, in G_{i-1} then let C be this new component. The decomposition forest for G_i contains a new vertex for C that connects the two vertices A and B. The minimum spanning forest for G_i contains the edge (u, v), since it is the smallest weight edge that connects a vertex in A with a vertex in $V - A$. It

also connects two components in $\text{MSF}(G_{i-1})$, in fact the components A and B, creating the component C. Thus $\text{DF}(\text{MSF}(G_{i-1}) \cup \{u, v\}) = \text{DF}(\text{MSF}(G_{i-1})) \cup \{(C, A), (C, B)\}$. So

$$\begin{aligned}\text{DF}(G_i) &= \text{DF}(G_{i-1}) \cup \{(C, A), (C, B)\} \\ &= \text{DF}(\text{MSF}(G_{i-1})) \cup \{(C, A), (C, B)\} \text{(by induction)} \\ &= \text{DF}(\text{MSF}(G_{i-1}) \cup \{(u, v)\}) = \text{DF}(\text{MSF}(G_i))\end{aligned}$$

(b) Use a slight modification of Kruskal's algorithm. Modify the nodes of the disjoint sets structure so that each holds a pointer to a node of the decomposition tree, in addition to its normal rank and parent-pointer. Initially, make node u of the disjoint sets structure, which represents vertex u of the graph G, point to leaf u of the decomposition tree. During Kruskal's algorithm, whenever the Union operation is called, create a new decomposition tree node C for the connected component created by the Union operation. Give this node, C, the two children, A and B, which are the decomposition tree nodes pointed to by the two roots of the trees in the disjoint sets structure that are being Unioned. Modify the root of the resulting tree in the disjoint sets structure so it points to the new decomposition tree node C (note: it used to point to either A or B.) At the end of the algorithm, the root of the disjoint sets structure points to the root of the decomposition tree. The running time of the algorithm is the same as Kruskal's algorithm except for an additional constant amount of time per Union operation, and $O(n)$ time to initialize the leaves of the decomposition tree and pointers in the disjoint sets structure.