**Question: unique MST** Let G be an undirected graph with costs on the edges. Prove by contradiction that G has a unique minimum spanning tree if all the edge costs are distinct (i.e. no two edges have the same cost).

**Solution** Suppose for a contradiction there are two minimum spanning trees T and T that are different. That is, they have the same weight, because they are both minimum, but they consist of different sets of edges. There must be an edge T ' that is not in T. Let e be such an edge. ($e \in T' \setminus T$.)

Consider the elementary cycle of e with respect to T: let P be the path between the endpoints of e in T. As we have seen in class $T \setminus \{e\} \cup \{f\}$ is a spanning tree for any edge $f \in P$. Since T is an MST, $w(e) \geq w(f)$ (otherwise we could construct a cheaper spanning tree). Since the weights are unique, $w(e) > w(f)$. Therefore, e is the heaviest edge in the cycle $P \cup \{e\}$.

Delete e from T ' creating trees $T'_1$ and $T'_2$ Since P is a path between the endpoints of e, P starts at a vertex of T and ends at a vertex of $T'_1$. Therefore, P must contain an edge g not in T ' with one endpoint in

$T'_1$ and the other endpoint in $T'_2$. $T'_1 \cup T'_2 \cup \{g\}$ is a spanning tree. Further, since e is the maximum-weight edge of $P \cup \{e\}$, $w(e) > w(g)$ and the weight of the new tree is less than w(T), contradicting that T was an MST.

**Question 5.9 from DPV**

(a) false;

simple counterexample

(b) true; proof idea:

Assume this edge e with heaviest weight is in the MST. Taking this edge from the tree gives us a cut (S, V – S). To make that e is in MST, e must be the edge with minimum weight connecting S and V – S. But we know that e is the unique heaviest edge in graph, so it must be the only edge connecting S and V – S. This means it cannot be on a cycle. (contradiction).

(c) true; proof idea:

We only need to show that there exists an MST containing this edge. Run Kruskal's algorithm, and it is always possible to select this edge as the first one to add to minimum spanning tree.

(d) true; proof idea:

Every possible MST can be constructed by Kruskal's algorithm (by considering all cases of breaking up ties). Kruskal's algorithm will always add this lightest edge as the first one to the MST, so it is part of every MST.

(e) true; proof idea:

We can prove this by contradiction. Assume e is not a lightest edge across any cut of G. Because e is part of an MST, by removing e from the tree, we disconnect the tree into two parts; let the nodes in these two parts be S and V – S respectively. Since (S, V – S) is a cut, and e is not a lightest edge across any cut, then there must exist some other edge e' connecting (S, V – S) and its weight is smaller than e's. Replacing e in the original MST with this new edge e', we get a spanning tree with smaller weight. This contradicts the fact that the original spanning tree is minimum.

(f) true; proof idea:

Kruskal's algorithm can construct every possible MST if we consider all cases of breaking up ties. We can use this fact to prove that e must be part of every MST.

In some stage of Kruskal's algorithm, the edges in this cycle will be considered. Because edges are sorted before we start constructing the spanning tree, and e is the unique lightest edge in the cycle, we will always add it to the spanning tree before other edges in the cycle. It is also impossible for us to violate any union condition; none of the edges in that cycle has been considered yet, so the two endpoints of e have different parent pointers at this stage. Because Kruskal's algorithm can construct every MST and will add e every time, e must be part of every MST.

(g) false;

simple counterexample

(h) false; counterexample:

same graph as the one above; shortest distance between A and B is 100, but that edge is not in MST.

(i) true.

It is always possible to just add a very big positive number to every edge weight and then run Prim's algorithm on this new graph without negative edges. The MST of this new graph will be the same as the MST of the old graph.

(j) true; proof idea:

We prove this by contradiction. Assume there exists an MST T of G that does not contain an r-path from s to t, meaning on this path p' there is at least one edge with weight greater than or equal to r. Pick the heaviest edge e' on p', then e' must be such an edge with weight ≥ r. Therefore, between s and t, there are at least two paths, one is the original r-path p, the other is p from MST.

Sort the edges and run Kruskal's algorithm. To make sure that p' is in the MST, we must add e' to the spanning tree. At that moment, all the edges lighter than e' have already been considered, which means all edges on path p and all the other edges on p' have been looked at and their endpoints are unioned. This means the two endpoints of e' now have the same parent pointers, so e' can never be added to the MST. This contradicts the fact that p' is in MST.