R-14.1 Hint: The fact that the edge weights are distinct means that no edge weight is unique.

R-14.2 Hint: Think about how we modified DFS and BFS for directed graphs in the previous chapter.

R-14.3 Hint: Think about how to update this tree each time we do a relaxation step.

C-14.2 Hint: Examine closely the justification for why Dijkstra's algorithm works correctly and note the place where it breaks if negative-weight edges.

R-15.1 Hint: Review the MST algorithm in question.

R-15.2 Hint: Review the MST algorithm in question.

C-15.3 **Hint:** Use a crucial fact about minimum spanning trees.

C-15.6 **Hint:** Think about an alternative way of implementing the priority queue, Q, that would support fast insertions and updates at the expense of the removeMind operation.

Solution: The main modification in this case is to implement the priority queue, Q, as an unordered doubly linked list. This allows us to insert and update the key for any vertex in $O(1)$ time and remove the minimum in $O(n)$ time. We perform $O(m)$ insertions and updates and $O(n)$ removeMin operations in the algorithm, and these computations are the bottlenecks. Thus, since m is $O(n^2)$, the total running time is $O(n^2)$ in this case in the worst case.