(a) The following lgorithm checks the validity of the given $d(v)$s in $O(m)$ time: If for any edge $e = (v, w)$, we have that $d(v) > d(w) + c_e$, then we can immediately reject. This follows since if $d(w)$ is correct, then there is a path from v to t via w of cost $d(w) + c_e$. The minimum distance from v to t is at most this value, so $d(v)$ must be incorrect. Now consider the graph G' formed by taking G and removing all edges except those $e = (v, w)$ for which $d(v) = d(w) + c_e$. We will now check that every node has a path to t in this new graph (this can easily be checked in $O(m)$ time by reversing all edges and doing a DFS or BFS). If any node fails to have such a path, reject. Otherwise accept. Observe that if $d(v)$ were correct for all nodes v, then if we consider those edges on the shortest path from any node v to t, these edges will all be in G'. Therefore we can safely reject if any node can not reach t in G'.

So far we have argued that when our algorithm rejects a set of distances, those distances must have been incorrect. We now need to show that if our algorithm accepts, then the distances are correct. Let $d'(v)$ be the actual distance in G from v to t. We want to show that $d'(v) = d(v)$ for all v. Consider the path found by our algorithm in G' from v to t. The real cost of this path is exactly $d(v)$. There may be shorter paths, but we know that $d'(v) \leq d(v)$, and this holds for all v. Now suppose that there is some v for which $d'(v) < d(v)$. Consider the actual shortest path P from v to t. Let x be the last on P for which $d'(x) < d(x)$. Let y be the node after x on P. By our choice of x, $d'(y) = d(y)$. Since P is the real shortest path to t from v, it is also the real shortest path from all nodes on P. So $d'(x) = d'(y) + c_e$ where $e = (x, y)$. This implies that $d(x) > d'(x) = d(y) + c_e$. But then we have a contradiction, since our algorithm would have rejected upon inspection of e. Hence $d'(v) = d(v)$ for all v, so the claim is proved.

The same solution can also be phrased in terms of the modified cost function given in part (b).

There are two common mistakes. One is to simply use the algorithm that just checks for $d(v) > d(w) + c_e$. This can be broken if the graph has a cycle of cost 0, as the nodes on such a cycle may be self consistent, but may have a much larger distance to the root than reported. Consider two nodes x and y connected to each other with 0 cost edges, and connected to t by an edge of cost 5. If we report that both of these are at distance 3 from the root, the algorithm faulty will accept the distances. The other common mistake was to fail to prove that the algorithm is correct on both yes and no instances of the problem.

(b) Given distances to a sink t, we can efficiently calculate distance to another sink t' in $O(m \log n)$ time. To do so, note that if only edge costs were non-negative, then we could just run Dijkstra's algorithm. We will modify costs to ensure this, without changing the min cost paths. Consider modifying the cost of each edge $e = (v, w)$ as follows: $c'_e = c_e - d(v) + d(w)$. First observe that the modiffied costs are non-negative, since if $c'_e < 0$ then $d(v) < d(w) + c_e$ which is not possible if d reflect true distances to t. Now consider any path P from some node x to t'. The real cost of P is $c(P) = \sum_{e \in P} c_e$. The modified cost of P is $c'(P) = \sum_{e \in P} c'e = \sum_{e \in P} c_e - d(v) + d(w)$. Note that all but the first and last node in P occur once positively and once negatively in this sum, so we get that $c'(P) = c(P) - d(x) + d(t')$. Hence

[1] ex738.372.857

the modiffied cost of any path from x to t' differs from the real cost of that same path an additive $d(t') - d(x)$, a constant. So the set of minimum cost paths from x to t' under c' is the same as the set under c. Furthermore, given the min distance from x to t' under c', we can calculate the min distance under c by simply adding $d(x) - d(t')$. Our algorithm is exactly that: calculate the modiffied costs, run Dijkstra's algorithm from t' (edges need to be reversed for the standard implementation), and then adjust the distances as described. This takes $O(m + m \log n + n) = O(m \log n)$ time.