W8.

a) To determine if you can get from every town to every other town with your car, you must first determine which towns you can and cannot travel to. To do so, initialize a graph and add the vertices and edges from G such that the weight of any edge is less than L. Since the cost of checking the weight of an edge is in constant time, the cost of this is O(V+E). We then need to check if the graph is connected using a modified backtracking algorithm. In this tree, once a node has been explored, it no longer needs to be explored again. In this method, the search does not stop until the leaf of a node has been reached where you can check for connectivity of the graph. This is known as the depth first search algorithm and can be executed in linear time since we explore each node at most once.

b) Base Case:
- Start by sorting the edges by weight and initialize the lower and upper bound on the weights of the edges.

Recursive Case:
- Use binary search by setting the middle value as the average of the upper and lower bound of the weights of the edges
    - Keep track of the edges accounted for so far in a minimum spanning tree (Prim's algorithm)
    - If the MST connects, let the upper bound be the current middle value to search for smaller values
    - If the MST does not connect, set the lower bound to the middle value + 1 to search for larger values
- The smallest middle value found is L*

Sorting the edges by weight can be achieved using a mergesort algorithm which runs in O(nlogn) time in terms of the length of the instance. Thus, the cost of sorting the edges in G would be O(|E|log|E|).

Since we rely on Prim's algorithm to create the minimum spanning tree in our recursive case, the running time of such is O(mlogn). For the graph G, the running time is O(|E|log|V|).

Thus, the running time of this algorithm is O(|E|log|E|) + O(|E|log|V|). Since we know that E <= V, then this can be simplified to O(|E|log|V|), thus proving our statement.