1.) Minimum_roads_danger (T, R):

$D = \forall r \in R\ (d(r) = r.\text{danger})$

new Graph $G(V{:}\ T,\ E{:}\ R,\ Wt{:}\ D)$

return Kruskal's MST(G)

T – set of all Towns
R – set of all roads
$r \in R$ r.danger = danger level for road

I am unsure of the input so I made it Towns & roads with roads having danger level

I create Graph using Towns as vertices, roads as edges, and danger levels to each road as wt.

I then get the MST of the graph.

This gets a tree which would have $|V|-1$ edges, the minimum possible.

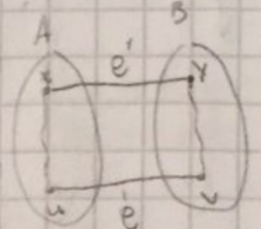
And would contain all the smallest weighted edges to create a spanning tree.

As such, the max weighted edge in the returned tree is the smallest possible. If not then the returned tree is not an MST.

$O(|E|)$ to get set of danger levels per road

$O(|E|\log|E|)$ – Kruskals on graph of towns, roads and danger levels

$\rightarrow O(|E|\log(|E|))$



e is edge in T, returned tree
$wt(e') < wt(e)$
T' replaces e for e'
$wt(T') < wt(T)$

$\rightarrow$ So T isn't MST if there exists smaller edge than e that can connect subtrees e is connecting

Not sure if needed.

2.) Prove GCP - decreasing of v_i / r_i

Minimize $\sum_{i=1}^{n} v_i \cdot f_i$

proof:

OPT - optimal solution that doesn't meet above GCP
Two customers i & j where i is right before j
and

$v_i / r_i < v_j / r_j \rightarrow v_i \cdot r_j < v_j \cdot r_i$

i starts at t
j starts at $t + r_i$
$f_i = t + r_i$
$f_j = t + r_i + r_j$
$V_{ij} = v_i \cdot t + v_i \cdot r_i + v_j \cdot t + v_j \cdot r_i + v_j \cdot r_j$

if we swap order of j & i $\rightarrow$ New - new solution (i & j swapped)
$f'_i = t + r_j + r_i$
$f'_j = t + r_j$

$V'_{ij} = v_j \cdot t + v_j \cdot r_j + v_i \cdot t + v_i \cdot r_j + v_i \cdot r_i$

$V'_{ij} < V_{ij}$ since $v_i \cdot r_j < v_j \cdot r_i$

$\rightarrow$ sum of New < sum of OPT

New is optimal since sum smaller

Repeat for until no inversions in OPT
(will create solution satisfying GCP)

3.) Prove GCP - farthest possible on current charge

Minimum charging stops

Proof:

OPT - optimal solution | sets of towns stopped in to charge
GCP - greedy solution using above GCP

For arbitrary number $k \geq 0$, $OPT(0..k) == GCP(0..k)$
where the towns stopped in are the same

$\rightarrow$ $S(1) \ldots S(k)$ are same in OPT & GCP,
where $S(i)$ is ith town stopped at

$S(k+1) = (k+1)$th stop in OPT
$S'(k+1) = (k+1)$th stop in GCP

$S(k+1) \neq S'(k+1)$

$>$

Based on GCP, $S'(k+1)$ is farther than $S(k+1)$

Change OPT to have $S(k+1) = S'(k+1)$

OPT now matches GCP until $k+1$ stops.
Can continue until OPT matches GCP (inductively)

4.) Tree_has_short($G(V,E,wt)$, $T(V,E')$, S_0):

get weights of tree using dfs

```
dt[S0] = 0    ∀v∈V dt[v] = ∞
tree_weights(S0)
```

dijkstra's

```
dg[S0] = 0    ∀v∈V dg[v] = ∞
create PQ with all V and dg[v]
while PQ not empty
    v = extract min(PQ)
    For ∀(v,u) ∈ E
        if dg[u] > dg[v] + wt(v,u) in G
            dg[u] = dg[v] + wt(v,u) in G
            update PQ (u, d[u])
```

compare weights

```
for ∀v ∈ V
    if dt[v] > dg[v]
        return false

return true
```

```
tree_weights(v):
    v.visited = true
    for ∀(v,u) ∈ E'
        if u.visited == false
            dt[u] = dt[v] + wt(v,u) in G
            tree-weights(u)
```

Since T is a tree, a node's wt would be the wt of its parent plus the weight of the edge between the two. So I can use dfs to recurse down tree branches to get the weights of every v.

I use dijkstra's to generate an array of the shortest path weights.

I then compare the two arrays. If the tree's weight is larger than dijkstra's minimum weight then it doesn't have the shortest path.

This works when their are multiple shortest paths. When multiple, we can't just compare dijkstra's tree to T so we compare wt.

Dijkistra's is $O((|V|+|E|)\log|V|)$ ← largest

Getting weights by tree edges E' using dfs is $O(|E'|)$ or $O(|E|)$ since $E' \subseteq E$

Comparing weights is $O(|V|)$

Two inits for dt & dg are $O(|V|)$

$\rightarrow \in O((|V|+|E|)\log|V|)$