(a)

We will prove by contradiction.

Let's suppose that greedy solution T_G is not the optimal solution T_O. Then that means that there exists an edge $e = \{u, v\}$ that is in T_O that is not in the T_G. We see that e must be a **safe** edge by the property of the MST. Then that means it's possible that we partition the G into two parts S and $V \setminus S$ such that $u \in U$ and $v \in V$. Since we find a spanning tree T_G, there will be a unique path P_{uv} between u to v. Then it must be true that one edge $e' = \{u', v'\}$ exists in the P_{uv} where $u' \in V$ and $v' \in V \setminus S$. We see that since e is the **safe** edge, it must mean that $w(e) < w(e')$.

We see that if replace the e' with e then the T_G will still be a spanning tree but with smaller cost. This is because that previously if there are two nodes u'' and v'' in T_G and the path P'' between them has $e' = \{u', v'\}$ in it, we could replace this single edge with the new walk from u' to u, and then u to v via the newly added edge e, and then v to v'. (There still exists, obviously, the path from u' to u, and v' to v even if we remove the e') So every node in this new T_G is still connected. It also doesn't form any cycle because the new T_G still only has $n - 1$ edges. That means it's possible that we could replace the edge e' with e while make sure that T_G is still an spanning tree.

But notice here, in this case our greedy algorithm would have already done this replacement! This only leads to the fact that T_G couldn't be the greedy solution. This contradicts with our assumption, and this means that the actual greedy solution T_G must equal to T_O.

Q.E.D.

(b)

We name edge alphabetically so that we make every edge distinct. (so we could the proof later). This is mentioned in lecture and won't cause any problem.

The greedy algorithm is straight forward:

- Get the subgraph $G' = (V', E') \subseteq G$ where $V' = V \setminus U$, and the $E' \subseteq E$ such that $e' = \{u', v'\} \in E'$ where $u', v' \in V'$.
- We then check whether the constructed V' is empty.
    - If V' is empty, that means all the nodes are **unreliable**. Check $|U|$
        - If $|U| > 2$, return `null` to indicate there doesn't MST that satisfies the requirement.
        - If $|U| = 0$, then G is an empty graph. Then there is an MST that satisfies the requirement (it's jus the empty tree), return the empty tree.
        - If $|U| = 1$, then G is a graph with only one node, return the MST that only includes that node as the root.
        - If $|U| = 2$, then G is a graph with two nodes. If there is edge between these two nodes, return the MST that includes these two nodes and the edge that connecting them. Otherwise return `null` to indicates that there doesn't MST that satisfies the requirement.
    - Otherwise that means V' is not empty. Invoke Prim's algorithm on this subgraph (or whatever MST algorithm that fit best in this case). The algorithm could tell us whether there is a MST for the subgraph.

- If there doesn't exist such MST, return `null` to indicate there doesn't MST that satisfies the requirement.
- Otherwise, we will find the MST T' for the subgraph. We construct a new MST for the entire graph T, let T first includes all the edges and nodes in T'. Then foreach $u \in U$, we find all the edges that connect the u to some element v in the MST.
  - If such edge doesn't exist, we return `null` to indicate there doesn't MST that satisfies the requirement.
  - Otherwise we choose the edge with minimum weight among all these edges. Add the current u and that edge to T.

  After we do all these and reach here, return T.

We first analyze it's time complexity. We use the Prim's algorithm there so it will take $O(n \log n + m)$ time. Later we iterate through all the nodes in the U, and some edges that connects to them in E. This will cost $O(n + m)$ in worst case scenario. Therefore, the total time complexity is dominated by the time from Prim's algorithm, and it's $O(n \log n + m)$.

To prove the correctness, we will analyze them case by case:

- The case when $|V'| = 0$
  - $|U| = 0$. This is trivially correct since the graph G doesn't even have any node.
  - $|U| = 1$. This is also trivially correct since there is only one unreliable node in the graph G, and the MST is, of course, the tree with itself as the root. It's also, of course, a leaf.
  - $|U| = 2$. This is also trivially correct since there is only two unreliable nodes in the graph G. If there is edge between these two nodes, the G itself is, of course, the MST and these two nodes are, of course, leaves. If there is not edge between these two nodes, the graph is not even connected and of course it's true that such MST doesn't exist.
  - $|U| > 2$. In this case there simply doesn't exist such MST. If G is not connected, then we are done, and MST doesn't exist. So, suppose the G is connected, and we have a way to get all of them connected. Suppose there exists $u \in U$ and $v \in U$ and they are connected via edge $\{u, v\}$, then since $|U| > 2$, we could find another $w \in U$. Since they are all connected, there exists a path from u to w. There could be edge $\{u, w\}$ since u is a leaf. Then that means such path includes the edge $\{u, v\}$ and another edge $\{v, x\}$ where $x \in U$ that eventually connects to U. However, this makes v has two neighbors and is no longer a leaf. So that means there doesn't exist a way to get all of them connected. That means such MST doesn't exist.
- The case when $|V'| \neq 0$. We could try to find the MST of the subgraph G' using Prim's algorithm.
  - If MST doesn't exist for G' then the MST doesn't exist for G and we are done. This is because in this case that means G' is not connected and we couldn't use the nodes in U to connect the components in G', this will make the nodes in U not leaves. So we have no way to make all the nodes connected while satisfying the constraints, so MST for G doesn't exist.
  - Otherwise the MST for G' will exist. In this case, suppose the greedy solution is T_G that is different from the optimal solution is T_O. We will prove by contradiction. Then this means that there must be an edge $e = \{u, v\} \in E$ in the T_O that is not in the T_G. There are few cases for this edge:

- It cannot be a edge that connects two unreliable nodes. Suppose there exists such edge e in the MST. In this case $u \in U$ and $v \in U$, if we want to find a path from u to $w \in V'$. Since u is a leaf, the path it takes must go through v. Since such path must exist, it means that $v \in V$ are now connected both to u and some other node x which eventually goes to w. This will make v no longer a leaf and violates the constraints. So, such edge that connects two unreliable nodes doesn't exist.
- It cannot be a edge that connects one reliable node and one unreliable node. In this case, without loss of generality, suppose $u \in U$ and $v \in V'$. Then we see that this edge will just be the edge with minimum weights among all the edges that connects u to one node in V'. (This is true because suppose this is not the edge with minimum weights, we could always find another e' from u to one element in the v in V', we could use the e' to replace e, this replacement will keep the modified tree still a MST (u is still connected, and there isn't any cycle). However, in this case the modified tree will have lower total weights, this is a contradiction with the definition of optimal solution. So, such edge must be the one with minimum weights). However, in this case we see that our greedy solution will also include this edge because it's the edge with minimum weights among the edges that connects to one node in V'. That means this edge exists both in optimal solution and greedy solution. This is impossible as it contradicts with the assumption that such edge is in T_O but not in T_G. So, this edge could not be a edge that connects one reliable node and one unreliable node.
- It cannot be a edge that connects two reliable nodes. Since it's an optimal solution, that means this edge is a **safe** edge in G. Then that means there exist a partition that make V into Q and $V \setminus Q$ that e is the unique minimum cost edge cross these two partitions. If we exclude all the unreliable node in U from both partitions, this property will still holds. (Since it's still the **minimum**) This means such edge is also a **safe** edge in G' in this case. However, our greedy solution uses Prim's algorithm on the G' to find all the edge, that means the MST of G' in the greedy solution also includes this edge e. That means this edge exists both in optimal solution and greedy solution. This is impossible as it contradicts with the assumption that such edge is in T_O but not in T_G. So, this edge could not be a edge that connects two reliable nodes.

In this case, we see that it's possible to find such e that exists in T_O but not in T_G, this leads to a contradiction, and it means that T_G must be T_O.

Therefore, we have proven that our algorithm gives correct results for every possible cases.

Q.E.D.