Question 1b

Consider the following algorithm:

```
UPDATE-MST(V, E, w, T, e_0, w_0)
 1  if e_0 ∈ T
 2       // Suppose e_0 = (u_0, v_0)
 3       for v ∈ V
 4            Augment v to keep track of a representative vertex.
 5       Run DFS on T − {e_0} starting from u_0, where for each encountered vertex u, set u's representative to u_0
 6       Run DFS on T − {e_0} starting from v_0, where for each encountered vertex v, set v's representative to v_0
 7       best-weight = ∞
 8       best-node = NIL
 9       for e ∈ E − {e_0}
10            // suppose e = (u, v)
11            if representative(u) ≠ representative(v) and w(e) < best-weight
12                 best-weight ← w(e)
13                 best-node ← e
14       if best-weight = ∞
15            return NIL
16       else
17            return T − {e_0}∪ best-node
18  else
19       return T
```

0.1 Runtime Analysis

If the removed edge e_0 is not in T, then T is still a valid MST for G, and the algorithm returns T which is $O(1)$. Now let's consider the case where $e_0 \in T$.

1. Augmenting all vertices in V to keep track of a representative involves allocating $O(n)$ memory, which is takes $O(n)$ time.

2. Since T is an MST, removing an edge e_0 from T creates two disjoint trees, say T_1 and T_2, such that $|T_1| + |T_2| = T - 1$. Then running DFS on T_1 and T_2 and setting representatives (which is possible to implement in constant time) on lines 7-8 has a total runtime of $O(n)$.

3. Each loop of the for loop starting on line 9 takes a constant number of operations (accessing the representatives, comparing values, and setting values), which means that the entire for loop is $O(m)$

Hence, the total worst case runtime is $O(n + m)$ which in practice is faster than $O(m \log^* n)$.

0.2 Proof of Correctness

We limit our attention to the case in which $e_0 \in T$.

If there is no edge in E which connects the disjoint trees created by the deletion of e_0 in T (V_1 and V_2 in the algorithm), then removing e_0 also causes a disconnection in G since by definition this means that the edges in E but not in T do not connect V_1 and V_2. In this case, it is impossible to construct a spanning tree, and so there is no solution, which is what is returned in the algorithm in this case (it returns NIL if it iterates through the edges and cannot find an edge connecting V_1 and V_2).

Removing e_0 from T yields two connected components C_1 and C_2, since T was a minimum spanning tree (property). The algorithm simply finds the minimum cost edge e between the two connected components, and adds this to $T - \{e_0\}$ as its result. Suppose, for the sake of contradiction, that the resulting tree $T_1 := T - \{e_0\} \cup \{e\}$ is not a minimum spanning tree. Then there must exist some other tree T' (which is minimum) such that $w(T') < w(T)$. Consider the following cases:

1. Case 1: $e \notin T'$
   Then there must be some other edge e' which connects a vertex from one of the edges in C_1, call it u', to a vertex from one of the edges in C_2, call it v' (otherwise there would be a disconnection and T' would not be spanning). Now, by adding e_0 and removing e' from T', we obtain a tree $T'' \in E$ such that $w(T'') = w(T') - w(e') + w(e_0)$. We know that $w(T') = w(T) - w(e_0) + w(e)$ and, since based on the algorithm e is cost edge connecting a vertex from one of the edges in C_1 to a vertex from one of the edges in C_2, we also know that $w(e) \leq w(e')$. Then $w(T'') = w(T) - w(e_0) + w(e) - w(e') + w(e_0)$ where $w(e) - w(e') \leq 0$, which means that $w(T'') \leq w(T)$, leading to a contradiction since T is an MST for G.

2. Case 2: $e \in T'$
   Then at least one of the subtrees in T' corresponding to C_1 and C_2 (i.e. the subtrees that contain the same vertices as in C_1 and C_2), denote them as C_1', C_2' have a weight less than its counterpart. Suppose without loss of generality that $w(C_1)' < w(C_1)$. Then $w(C_1' \cup C_2 \cup e_0) < w(C_1 \cup C_2 \cup e_0) = w(T)$ and we have a contradiction since this gives a tree $T'' \subseteq E$ with a weight lower than T which is an MST for G.

Since either case leads to a contradiction, there does not exist a tree T' where $w(T') < w(T)$. Hence, T is an MST.