CSC373 Assignment 1
24 February 2015

Eric Bannatyne
1000468451

The following algorithm updates the given minimum spanning tree T of G, to produce a new minimum spanning tree T_1 for G_1.

```
UPDATE-MST(V, E, w, T, e_1, w_1)
T_1 = T ∪ {e_1}
D = DFS tree produced by DFS on T_1 starting from u, including information about back edges // CLRS p. 610
e = NIL
weight = 0
// Find the (unique) back edge of D.
// This must have u as an endpoint since e_1 is in the cycle and DFS was started at u.
for x in u.neighbours // Neighbours in T_1
    if {x, u} is a back edge of D
        e = {x, u}
        weight = w(e)
        break
// Traverse up along the cycle in the DFS tree until the root u is reached,
// keeping track of the maximum-weight edge.
while x ≠ u
    if w({x, x.parent}) > weight // x.parent in D
        e = {x, x.parent}
        weight = w(e)
    x = x.parent
T_1 = T_1 − {e}
return T_1
```

Correctness

On a high level, this algorithm updates T by inserting the new edge $e_1 = \{u, v\}$ into T. This produces exactly one cycle in the graph $T \cup \{e_1\}$ (Result given on Piazza). The algorithm then finds and removes the maximum-weight edge e from the cycle, to produce a new tree $T_1 = T \cup \{e_1\} - \{e\}$. This is, in fact, a spanning tree, since the removed edge e is on a cycle, meaning that neither of the endpoints of e become isolated vertices when e is removed. We will show that T_1 is in fact a minimum spanning tree.

By definition, we have $w(T_1) = w(T \cup \{e_1\} - \{e\}) = w(T) + w(e_1) - w(e)$. However, since e is a maximum-weight vertex on its cycle in $T \cup \{e_1\}$ and e_1 lies on that cycle, we have $w(e_1) \leq w(e)$, which implies that $w(T) \geq w(T_1)$.

To show that the spanning tree that this algorithm produces is indeed a minimum spanning tree of G_1, suppose that T_1 is not a MST. Then since G_1 is connected, there must be some MST T_1' for G_1 such that $w(T_1') < w(T_1)$. We have two cases to consider, depending on whether or not T_1' contains e_1.

If $e_1 \notin T_1'$, then T_1' must be a spanning tree for G, which means that $w(T) \leq w(T_1')$. However, since we established that $w(T_1) \leq w(T) \leq w(T_1')$, this contradicts our assumption that $w(T_1') < w(T_1)$. Therefore T_1 must also be a minimum spanning tree.

Now suppose that $e_1 \in T_1'$. Removing $e_1 = \{u, v\}$ from T_1' must disconnect the tree, such that $T_1' - \{e_1\}$ contains exactly two connected components $A = (V_A, E_A)$ and $B = (V_B, E_B)$, such that $u \in V_A$ and $v \in V_B$. Let C be the unique cycle contained in $T \cup \{e_1\}$. It will be helpful to prove the following lemma.

Lemma 1. *There is some edge $e' = \{a, b\} \in C - \{e_1\}$ such that $a \in V_A$ and $b \in V_B$.*

Proof. Since C is a cycle, $C - \{e_1\}$ must be a connected subgraph of T which is a chain of the form

$$u = w_1 \longleftrightarrow w_2 \longleftrightarrow \ldots \longleftrightarrow w_k = v,$$

where "$\longleftrightarrow$" denotes "is adjacent to (in $C - \{e_1\}$)". Since $V_A \cap C$ and $V_B \cap C$ form a partition of the vertices included in C, and we know that $u \in V_A$ and $v \in V_B$, there must be some i such that $w_i \in V_A$ and $w_{i+1} \in V_B$. Choosing $e' = \{w_i, w_{i+1}\}$ completes the proof. $\square$

This means that, if we remove e_1 from T_1', there must be some edge e' in $C - \{e_1\}$ such that $T_1' - \{e_1\} \cup \{e'\}$ is a spanning tree of G. Since $e' \in C$, we know that $w(e') \leq w(e)$, where e is the edge that the algorithm chose to remove from the cycle when producing T_1. Thus, we have

$$\begin{aligned} w(T) &\leq w(T_1') - w(e_1) + w(e') \\ &< w(T_1) - w(e_1) + w(e') \\ &\leq w(T_1) - w(e_1) + w(e) \\ &= w(T_1 - \{e_1\} \cup \{e\}) \\ &= w(T). \end{aligned}$$

Thus, $w(T) < w(T)$, which is a contradiction. Therefore T_1 must be a minimum spanning tree of G_1.

Running Time

We now analyze the running time of UPDATE-MST. Performing depth-first search to obtain the DFS tree D requires $\Theta(|V| + m)$ steps, where m is the number of edges in $T \cup \{e\}$. However, since T is a spanning tree of G, it contains $|V| - 1$ edges, so $m = |V|$, and so this step really only requires $\Theta(|V|)$ time.

The rest of the algorithm proceeds by examining the neighbours of u in $T \cup \{e_1\}$ to find a back edge, and then traversing a single cycle of $T \cup \{e_1\}$, both of which are bounded above by $O(|T|) = O(V|)$ operations, as before. Therefore UPDATE-MST runs in $\Theta(|V|)$ time in the worst case, an improvement over using the standard algorithms to produce a new minimum spanning tree from scratch.