CS 670 Spring 2015 - Solutions to HW 5

Problem 23.1-5

Let T be a minimum spanning tree of G. If T does not contain the edge e, we're done. Suppose T contains the edge e. Dropping the edge e from T leads to a partition of T into two connected components, say A and B. Suppose nodes $p;q$ in $V(G)$ are such that $e = (p;q)$ with $p \in V(A)$ and $q \in V(B)$.

e is a maximum-weight edge on some cycle of G. In particular, the nodes $p;q$ occur in this cycle. Hence, there is a path from p to q in G that does not employ the edge e. It is immediate that there exists an edge on this path, say $(r;s)$, such that $r \in V(A)$ and $s \in V(B)$ and the weight of $(r;s)$ is no more than the weight of e.

Consider the graph T'' formed from T by dropping edge e and adding edge $(r;s)$. This is a connected graph because A was connected, B was connected, and the edge f connects these two components to each other. Further, $w(T'') = w(T) - w(e) + w(r;s) \leq w(T)$. Since T was a minimum spanning tree of G, it follows that the connected graph T'' must be a minimum spanning tree of G. Further, T'' does not employ the edge e.

Problem 23.1-8

Let T and T' be two different MST's for the graph G such that the sorted lists for T and T', say L and L', are different. Let the first position of disagreement in the lists be at k. Without loss of generality, assume $L[k] < L'[k]$. Consider the edges corresponding to $L[1];L[2];:::;L[k]$, these edges form a $n-k$ component forest.

Now, we assert that there exists an edge whose weight is less than $L'[k]$ and adding it, results in a cycle with the maximum weighted edge in the cycle greater than equal to $L'[k]$

The assertion follows from the following argument. Take the first $k-1$ edges in L'. The edges result in $n-k+1$ forest. Since the first k edges in L result in $n-k$ forest, there must be some edge among these k edges adding which to the first $k-1$ edges in L' doesn't create a cycle with them and moreover results in reducing the number of components in the forest.
This edge is the one we are looking for. When this edge e is added to T' it creates a cycle as T' is a tree. Also this cycle involves an edge other than the first $k-1$ edges in T' since e didn't create a cycle with the first $k-1$ edges of L'.

Hence, now using the arguments in the previous problem, we can remove the maximum weighted edge and what we've done in essence is that we've replaced the maximum weight edge in the cycle whose weight is $\geq L'[k]$ with e whose weight is $< L'[k]$. Therefore we a obtain tree which is more optimal than the MST T'. A contradiction!

Problem 23.2-7

Idea: We essentially use the idea from Problem 23.1.5. We can remove the maximum weighted edge from a cycle to obtain a better tree.
Algorithm: Let e be the lightest edge across the cut $(V - v; v)$, where v is the new vertex added, then add e to the tree. Now, for each new edge $(v; w)$ added in the graph do the following. Add the edge $(v; w)$to the tree, remove the maximum edge from the resulting cycle.
Correctness: Observe that the new tree obtained at each stage is the MST for the edges that we seen up till now. The fact that the cycle resulting from the addition of an edge can be handled in the way done in the algorithm follows from an argument similar to that used in Problem 23.1-5.
Time Complexity: Finding the maximum weighted edge takes $O(|V|)$ time. Hence, the entire algorithm takes $O(k|V|)$ time, where k is the number of new edges added.

Problem 23-1

a.We give an example where the second-best MST is not unique. Let $V = \{u; v; w; x\}$, $E = \{(u; v) : 1; (u; w) : 2; (v; x) : 5; (v; w) : 3; (w; x) : 4\}$, where the numbers adjoining the edges are their weights. The MST for the graph is $\{(u; v); (u; w); (w; x)\}$. However, there are two second-best MST's: $\{(u; v); (v; x); (v; w)\}$ and $\{(u; w); (v; w); (w; x)\}$.

Claim: Let G be an undirected, connected graph. Then the minimum spanning tree is unique if all edge weights are distinct.
Proof: Suppose T_1 and T_2 are two distinct minimum spanning trees of G. Let $S = \{e \in E(G) \mid \text{either } e \in E(T_1) \setminus E(T_2) \text{ or } e \in E(T_2) \setminus E(T_1)\}$ be those edges that occur in either T_1 or T_2, but not both. By assumption, S is non-empty. Let e be the maximum-weight edge in S. Without loss of generality, we may assume that e occurs in T_1 and not in T_2. Dropping e from T_1 leads to two connected components, say A and B. Since T_2 is a spanning tree, there must be an edge f in T_2 from A to B. Clearly, $f \neq e$ since otherwise $e \in T_2$, a contradiction. Further, $f \notin E(T_1)$ since otherwise there would be two edges e and f across the cut $(A; B)$, contradicting T_1 being a tree. Hence, $f \in S$. From the definition of e, it follows that $w(f) \leq w(e)$. Since all weights are distinct, $w(f) < w(e)$. The spanning tree $T_1 - e + f$ has weight $w(T_1) - w(e) + w(f) < w(T_1)$, contradicting T_1 being a MST. □

b. We want to show that a second-best minimum spanning tree can be obtained from the minimum spanning tree by replacing only one edge. Bear in mind that any spanning tree has the same number of edges.

Let $T_1 = (V; E_1)$ be a MST of G, and let $T_2 = (V; E_2)$ be a 2B-MST of G such that T_2 *has as many edges in common with* T_1 *as possible* (‡). That is, T_2 is a second-best minimum spanning tree such that for all other second-best minimum spanning trees T, the number of edges in which T_2 differs from T_1 is less than or equal to the number of edges in which T differs from T_1.

Notation. $E_1 - E_2$ is the set of edges which are in E_1 but not in E_2.

$T_1 \neq T_2$ and $|E_1 - E_2| \neq 0$. We have two cases to analyze.

(i) If $|E_1 - E_2| = 1$, then we are done as we can transform T_1 to T_2 by replacing the edge which is in T_1 and not in T_2 by the edge which is in T_2 and not in T_1.

(ii) If $|E_1 - E_2| > 1$, then we show that we get a contradiction.

For $(u; v) \in E_1 - E_2$, define the cycle C by adding $(u; v)$ to T_2. There exists at least an edge on C that belongs to $E_2 - E_1$ for otherwise C is a cycle on T_1 as well and we have a contradiction. Let e be the edge of maximum weight in the cycle C and belongs to $E_2 - E_1$. Consider the cut $(S; V - S)$ that is formed by removing $(u; v)$ from T_1, (let's say u and all nodes in the remaining subtree connected to u are in S, while v

and all nodes in the other subtree connected to v are in $V - S$). Note that $(u; v)$ is the only edge of T_1 that crosses the cut, otherwise we have a cycle on T_1 hence a contradiction. We know that $(u; v)$ is a light edge crossing this cut because it is in T_1. Since $(u; v)$ belongs to the cycle C, there is at least one more crossing edge f beside $(u; v)$ that belongs to C, and we have $w(f) > w(u; v)$ since $(u; v)$ is light all edge weights are distinct. As observed before $(u; v)$ is the only crossing edge from E_1, so $f \notin E_1$ and $f \in E_2 - E_1$. We have argued that f is on the cycle C and belongs to $E_2 - E_1$, so $w(f) \leq w(e)$ and again since all edge weights are distinct, $w(f) < w(e)$. Hence $w(e) > w(u; v)$.

Now, construct the tree $T_3 = T_2 - \{e\} \cup \{(u; v)\}$.

By this construction, T_3 is a spanning tree of G and $w(T_3) < w(T_2)$.

And $T_1 \neq T_3$ so $w(T_1) < w(T_3) < w(T_2)$. So T_3 is a 2B-MST which has more edges in common with T_1 than T_2 has. This is a contradiction due to (‡). Hence $|E_1 - E_2| \not> 1$ and we are left with case (i).

c. Starting from each node v do a DFS, labeling each node. Each node, x, adjacent to v has the label $w(v; x)$. Any other node y, has the label $max\{label(p(y)); w(p(y); y)\}$, where $p(y)$ is the parent of y in the DFS tree. The label on a node x gives $max[v; x]$.

DFS on a graph takes $O(|V| + |E|)$ time. Performing DFS on a tree takes $O(|V|)$ since $|E| = |V| - 1$.

Since we are doing DFS on each node, the time complexity of the algorithm is $O(|V|^2)$.

d. Using the $max[u; v]$ computed in the previous part, simply find an edge $(u; v)$ not in the MST such that $w(u; v) - w(max[u; v])$ is the smallest. Computing the values of $max[u; v]$ costs $O(n^2)$, the number of edges not in the MST is $O(n^2)$. So the total running time is $O(n^2)$.