Homework 12, additional problems, solution set

1. The removal of $e = (u, v)$ splits T into two trees T^u and T^v, containing vertices u and v, respectively. The new minimum spanning tree, T^e will be obtained by adding the least weight edge joining T^u and T^v.

To see this, let T' be the MST of the new graph that contains the largest number of edges in common with T, i.e. the largest number of edges from $T^u \cup T^v$. However, suppose that it does not contain all the edges in $T^u \cup T^v$. Let (w, x) be an edge in $T^u \cup T^v$ which is not in T'. We now argue that we can swap (w, x) for an equal-weight edge in T' which is not in $T^u \cup T^v$, thereby producing a new MST with one more edge in common with T. This implies that there is an MST of the new graph that contains all of $T^u \cup T^v$, and therefore by the cut theorem applied to the set of vertices in T^u we conclude that adding e' to $T^u \cup T^v$ yields an MST of the new graph.

Consider the cut of T formed by removing edge (w, x). By the cut theorem, the lightest edge crossing this cut could be added to $T \setminus \{(w, x)\}$ to form an MST, from which we conclude that (w, x) is a lightest weight edge crossing this cut. Now consider the cycle formed by adding (w, x) to T'. There must be at least one edge e'' on the path from w to x in T' that crosses the cut induced by removing (w, x) from T, and $\text{wt}(e'') \geq \text{wt}(w, x)$, as (w, x) was a least weight edge crossing this cut. Therefore we could replace e'' by (w, x), yielding another MST with one more edge in common with T.

For the algorithm, we begin by labeling each vertex $v \in V$ by whether it belongs to T^u or T^v: this can be done by performing a traversal of each tree and takes $O(n)$ time. Then we scan the adjacency lists of the vertices $v \in V$ and keep track of the current lightest weight edge between T^u and T^v, for with the labels in hand we can check whether an edge goes between the two trees in $O(1)$ time. This takes $O(m)$ time over all the edges in the graph.

Thus the overall runtime is $O(n + m)$.

2. T remains an MST. The reason is that the sorted order of the edges is unchanged. Therefore if we run Kruskal's algorithm the edges will be processed in the same order and whether each edge forms a cycle or not when it is processed is unchanged.

However, the array Pred may change. Consider the following 4 vertex graph with vertices u, v, w, x and edges $(u, x), (u, v), (v, w), (w, x)$ of lengths 4,1,1,1; the shortest path tree has the edges (u, v), (v, w), and (w, x). When we increase the lengths by 1 to 5,2,2,2, respectively, the shortest path tree now has the edges (u, x), (u, v), and (v, w).

3. Problem: Let $G = (V, E)$ be a weighted graph in which all the edge weights are distinct. Then G has a single MST.

Solution. Let T be an MST of G, and let $(u, v) \in T$. Consider the cut of T induced by removing the edge $e = (u, v)$. (u, v) must be the lightest edge crossing this cut. Suppose there were another MST T' that does not include e. Consider the path P in T' from u to v. At least one of the edges e' on P must cross the cut of T induced by removing the edge e. Since e was the lightest weight edge crossing this cut, $\text{wt}(e') > \text{wt}(e)$, and then removing e' from T' and adding e would produce a spanning tree of smaller weight that T', contradicting the assumption that T' was an MST. We conclude that every MST of G must include edge e. But e was an arbitrary edge in T; therefore, every MST of G must include every edge in T, which means that T is the only MST of G.