Algorithms and Data Structures 2020/21
Week 9 sheet: Solutions

1. Draw an example of a weighted graph which has 2 MSTs.

   There are lots of these. For example, take a triangle whose edges have values $1, 2, 2$. Then there are 2 MSTs, obtained by dropping either of the "2"-edges.

2. Let $G, \mathcal{W}$ be a weighted graph in which all edge weights are distinct.

   Prove that the MST of $G, \mathcal{W}$ is unique.

   Proof: By contradiction.
   Suppose there are two different MSTs, $T1$ and $T2$. Choose $(u, v) \in T1 \setminus T2$. Let $T1(u), T1(v)$ be the two sub-MSTs obtained by deleting (u, v). Consider the path $p_{u,v}(T2)$ between u and v in $T2$. Let E' be the edges of $p_{u,v}(T2)$ that cross the cut $T1(u), T1(v)$. We are guaranteed that $|E'| \geq 1$.

   Two cases:
   (1) Suppose $\mathcal{W}(e) > \mathcal{W}(u, v)$ for some $e \in E'$. Then define $T2' = T2 \setminus \{e\}) \cup \{(u, v)\}$. This is a spanning tree of cost strictly less than $T2$. Contradiction!
   (2) Suppose $\exists e \in E'$ with $\mathcal{W}(e) < \mathcal{W}(u, v)$. Then define $T1' = (T1 \setminus \{(u, v)\}) \cup \{e\}$. This is a spanning tree of G with cost strictly less than $T1$. Contradiction!

   In either case we prove that one of $T1, T2$ was not a MST.

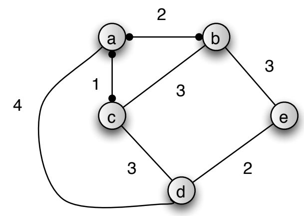
3. This question asks us to consider a modification to Prim's algorithm where we add all the competing minimum-weight fringe edges in a single iteration of Prim's algorithm. The question being asked is whether this strategy would lead to the construction of a Minimum Spanning Tree.

   The answer is **no**.

   Observation 1: This modification will not necessarily maintain a Tree through the life of the algorithm - for example, if the current tree T contains nodes a, b, c, and there are three competing minimum fringe edges $(a, d), (a, f), (c, d)$ all of weight 2, then adding all of them would induce a cycle. However, this answer is fairly easy ... suppose we tweak this algorithm to only add *one fringe edge per fringe-vertex*?

   Well, then the answer is still **no**.

   To see this consider the following graph as an example:



At the stage when the current spanning tree T contains a, b, c, there are three fringe edges $(a, d), (c, d), (b, e)$. Two of those edges, $(c, d), (b, e)$, have the minimum fringe edge weight 3. Also the fringe vertices (d and e) are distinct, so would both be added to give a complete spanning tree of weight 9. However, if we just added one of (c, d) and (b, d), then the weight 2 edge (d, e) would become available at the next step, and we would end up with a tree of value 8.

4. This question considers the case when a graph has more than one edge with the same weight value, in the context of Kruskal's algorithm. In the case where edge weights are non-unique, there may be more than one (in fact, there could be many) MSTa. This question is asking us to "trick" Kruskal's algorithm to return the MST we want. Note that Kruskal's algorithm is *deterministic* - once we have finished sorting the edges in terms of weight, then every step of our algorithm is determined. So our only chance to "trick" KRUSKAL is in doing the sorting.

   This is how we do it.
   Suppose we want the algorithm to return $\mathcal{T}$, but by obeying the rules of Kruskal's algorithm. We use the following trick. Take *any* sorting algorithm and sort the edges of G in terms of edge weight. Now look at this sorted list in the context of our special MST $\mathcal{T}$.
   For every weight w that labels *some* edge of G, the edges of G with weight w will all be grouped together in the sorted list. Identify which weight-w edges actually belong to $\mathcal{T}$ (not just G) and move these to the front of the w-group of edges in the sorted list (notice that after doing this the list is still sorted - but clearly not a stable sort!). Do this for every weight w which labels *some* edge of G.

   After doing this procedure, get KRUSKAL to start processing the edges in this order, adding each edge *iff* its endpoints belong to *different* components of the current forest.

   Claim: the tree that Kruskal returns is $\mathcal{T}$.
   proof: We prove this by induction. For each edge, we need to prove two things:
   (1) If e is in $\mathcal{T}$, it gets added by Kruskal.
   (2) If e is not in $\mathcal{T}$ it does not get added by Kruskal.
   Suppose that after processing k edges we have a forest F of little sub-MSTs such that every edge of F also lies in $\mathcal{T}$ (and such that any edge of $\mathcal{T} \setminus F$ appears *after* position k in our special sorted list). This is the Induction Hypothesis.

Induction step: We must prove the same thing holds, after we process edge e_{k+1}.

Case (1). Suppose $e_{k+1} \in \mathcal{T}$. We must show Kruskal will add e_{k+1} to F.
Consider our current sub-forest F after adding $e_1, \ldots, e_k$. The rule of Kruskal is that e_{k+1} will be added to F *iff* u and v belong to *different* subtrees of F. However, recall that $\mathsf{F} \subseteq \mathcal{T}$. If u and v lay in the *same* subtree of F, that would induce a cycle in $\mathsf{F} \cup \{e_{k+1}\}$, and (since $\mathsf{F} \cup \{e_{k+1}\} \subseteq \mathcal{T}$) therefore also induce a cycle in $\mathcal{T}$. Contradiction! We know $\mathcal{T}$ contains no cycles. Hence there is no problem, and e_{k+1} will be added.

Case (2). Suppose $e_{k+1} \notin \mathcal{T}$. We now show e_{k+1} will not be added to F.
Proof by contradiction. First observe that by our sorting of the edges (and by $e_{k+1} \notin \mathcal{T}$), it must be the case that *for every* $(x, y) \in \mathcal{T} \setminus \mathsf{F}$, $\mathcal{W}(x, y) > \mathcal{W}(e_{k+1})$. Now suppose (setting up for our contradiction) that $e_{k+1} = (u, v)$ such that u and v *are* in different components of F (and hence e_{k+1} will be added to F). Consider the path $p_{u,v}(\mathcal{T})$ between u and v in $\mathcal{T}$. Because u and v do not appear in the same connected component in F, $p_{u,v}(\mathsf{T})$ must contain some edge (x, y) of $\mathcal{T} \setminus \mathsf{F}$. However, if that is the case $\mathcal{W}(x, y) > \mathcal{W}(u, v)$. Then we can obtain an alternative spanning tree $\mathcal{T}' = (\mathcal{T} \setminus \{(x, y)\}) \cup \{(u, v)\}$ such that $\mathcal{W}(\mathcal{T}') < \mathcal{W}(\mathcal{T})$. Hence $\mathcal{T}$ was not a MST at all. Contradiction! So e_{k+1} would never have been added.