# Exercise Set Solutions #9

**"Discrete Mathematics" (2025)**

---

**E1.** Characterize the graphs $(V, E)$ for which every connected subgraph $(V', E')$ is equal to the induced subgraph on the respective set of vertices $V'$.

> ***Solution:*** It is trees. They are by definition connected and without cycles, so if we remove an edge they become disconnected. If a graph is connected with cycles the additional edge could be in the induced subgraph but not in $E'$ and $(V', E')$ is still connected.

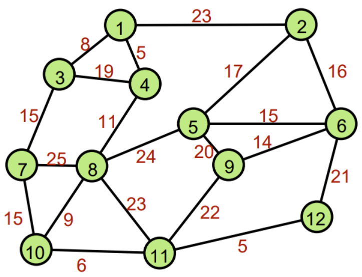
**E2.** Let $G$ be a connected weighted graph. Assume it has at least one cycle, and let edge $e$ be an edge that has strictly greater cost than all other edges in that cycle. Show that $e$ does not belong to any minimal weight spanning tree of $G$.

> ***Solution:*** Suppose that $e$ belongs to a minimal spanning tree $T$. Delete $e$ from $T$, and obtain a forest of two components, $A$ and $B$. Since $G$ has a cycle containing $e$, there is another edge $f$ in the cycle in $G$ that connects $A$ to $B$. By the choice of $e, f$ has smaller cost. Replace $e$ by $f$ in $T$, and obtain a smaller cost spanning tree, a contradiction.

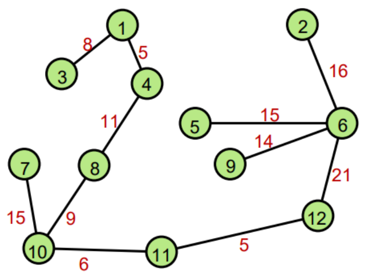
**E3.** Prove that any connected graph with distinct weights assigned to the edges has a unique minimal weight spanning tree.

> ***Solution:*** Suppose there are two minimal weight spanning trees, $T_1$ and $T_2$. Consider all those edges of $T_1$ and $T_2$ that are only in $T_1$ or only in $T_2$. Among these edges, let $e$ be the largest cost edge. We may assume that $e$ is in $T_1$ and is not in $T_2$. Deleting $e$ from $T_1$ disconnects $T_1$ into two components, $A$ and $B$. Since $T_2$ is a spanning tree, there is at least one edge in $T_2$ connecting $A$ and $B$. Let $f$ be the smallest cost such edge. Now, $f$ is in $T_2$ and not in $T_1$, so, by the choice of $e, f$ has smaller cost than $e$. On the other hand, if in $T_1$, we replace $e$ by $f$, we obtain a smaller cost spanning tree, thus, $T_1$ is not of minimal cost, a contradiction.

**E4.** Apply Kruskal's algorithm to the following graph to obtain a minimal spanning tree:



> ***Solution:*** The resulting minimal spanning tree is:



**E5.** Let $G=(V,E)$ be a weighted graph with the weight function $f: E \rightarrow \mathbb{R}$, i.e. the weight of an edge $E$ is euqal to $f(E)$. What can you say about the output of Kruskal's algorithm for $G$ (i.e. in terms of uniqueness or for part (3) and (4) how the output relates to the original output) if

**(a)** $f$ is a constant function. That is for all $e \in E, f(e)=C$ for some fixed $C \in \mathbb{R}$.

**(b)** $f$ is an injective function?

**(c)** $f$ is replaced by the function $-f$ defined as $e \mapsto -f(e)$ ?

**(d)** $f$ is replaced by a weight function $f': E \rightarrow \mathbb{R}$ such that for all $e_1, e_2 \in E, f\left(e_1\right) \leq f\left(e_2\right) \Leftrightarrow f'\left(e_1\right) \leq f'\left(e_2\right)$?

***Solution:***

**(a)** In this case, all spanning trees will have a total weight of $(|V|-1)C$. Hence any spanning tree is a minimal spanning tree. Kruskal's algorithm will therefore give any spanning depending on the choice of edges chosen (all the edges will have the same weight at each step of the algorithm).

**(b)** Exercise 2 above says that there is exactly 1 minimum spanning tree available for the output of Kruskal's algorithm. Hence in this case, the output of the Kruskal's algorithm is unique. You can also see this since at each step there is exactly one unique edge with the required conditions with the minimum weight and hence the algorithm will uniquely choose an output. In fact, following the proof of correctness of Kruskal's algorithm, we get that when $f$ is injective, there is a unique minimal spanning of $G$.

**(c)** For any spanning tree $T$ of $G$, let us denote $\mathrm{wt}_f(T)$ to be the total weight of the tree with respect to the weight function $f$. Then we can observe that $\mathrm{wt}_{-f}(T)=-\mathrm{wt}_f(T)$. Hence if $T, T'$ are two spanning trees then $\mathrm{wt}_f(T) \leq \mathrm{wt}_f\left(T'\right) \Leftrightarrow \mathrm{wt}_{-f}(T) \geq \mathrm{wt}_{-f}\left(T'\right)$.

So if $T$ is the minimal spanning tree of the original graph, after modifying the weight $T$ will be the maximal spanning tree. That is, for any spanning tree $T'$ of $G$ we must have $\mathrm{wt}_{-f}(T) \geq \mathrm{wt}_{-f}\left(T'\right)$.

**(d)** The output of Kruskal's algorithm will be a minimum spanning tree for the modified weight.

We can show this via the following claim. If $F \subseteq E$ is any set of edges of $G$, and if $e \in F$ is such that $f(e) \leq f\left(e'\right)$ for all $e' \in F, f'(e) \leq f'\left(e'\right)$ for all $e' \in F$. Indeed this trivially follows from the claim.

Now consider the set of edges among which the edge with the least weight is selected at each step of the Kruskal's algorithm. We first make a set $F$ of edges that have not been selected so far and that would not create a cycle with the already selected edges.

We then choose an $e \in F$ with the least value of $f(e)$ and add it. But this $e$ will also have the least value of $f'(e)$ according to the previous claim. Hence, Kruskal's algorithm for the modified weight can also choose $e \in F$ at this step. Inductively, we see that a Kruskal's algorithm with the new weight function $f'$ can give the same output as with $f$.

What this tells us is that the minimum spanning tree of $G$ depends only on the relative ordering of $E$ with respect to weights, i.e. how each edge $e \in E$ is ranked according to the weight.

**E6.** The following is called Prim's algorithm: Given a weighted graph $G$

- Initialize a tree with a single vertex, chosen arbitrarily from the graph.
- Grow the tree by one edge: of the edges that connect the tree to vertices not yet in the tree, find the minimum-weight edge, and add it to the tree.
- Repeat step 2 (until all vertices are in the tree).

Prove that the output of Prim's algorithm is a minimum spanning tree of $G$. Apply Prim's algorithm to the graph from Exercise 4.

***Solution:*** The proof is similar to the proof that Kruskal's algorithm produces a minimal weight spanning tree. We will show it in the case when the weights are all different. The case when some weights might be equal requires small modifications. Suppose for a contradiction that there is a minimal weight spanning tree $T$ whose weight is less than that of the tree found by Prim's algorithm, which we will denote by $P$. Number the edges of $P$ as $e_1, e_2, \ldots$ by the order in which they are selected by the algorithm. Let $i$ be the smallest index for which $e_i$ is not in $T$. Let $A$ denote the tree whose edges are $e_1, \ldots, e_{i-1}$. So, the edges in $A$ are shared by $P$ and $T$. Clearly, one of the two endpoints of $e_i$ , denote it by $u$ is in $A$, and the other one, denote it by $v$, is not. Now, $T$ is connected, so let $p$ be a path in $T$ from $u$ to $v$. Let $f$ be the first edge of the path $p$ that connects a vertex of $A$ to a vertex outside of $A$. By the algorithm, $e$ has smaller cost than $f$. On the other hand, $p \cup \{e_i\}$ is a cycle, so, by replacing $f$ by $e_i$ in $T$, we obtain a tree which is of lower cost than $T$, a contradiction.

If we admit the case when some weights might be equal, we have that $w(f)$ might be equal to $w(e_i)$. Using the same argument as above, we know that if we substitute $f$ by $e_i$ in $T$ we get another tree $T'$ and it holds that $w(T') = w(T)$. Anytime we find an edge in $P$ that is not in $T$ we can either have an absurd as in the case above (namely, the edge $e_i$ that we don't have in $T$ has cost strictly smaller than the edge $f$ as constructed above) or we can substitute these edges without changing the weight of the tree. Therefore, after having done this procedure for the whole tree we would find that $w(P) = w(T)$ which is a contradiction to our initial assumption $w(T) < w(P)$. Observe that the tree $T$ that we would have built with the different choices of edges (but with the same weights) could have been constructed by following Prim's algorithm. This happens when there is a step in Prim's algorithm, in which we can choose either to add $e_i$, or to add another edge with the same weight as $e_i$. In our notation this step occurs when we arrive to the vertex $u$ and we can choose either to add $e_i$ or $f$. In both cases, by continuing with the algorithm, we would end with a minimal weight spanning tree.

For example, in the graph from exercise 2 , two minimal weight spanning trees could differ because one contains the edge $\{3, 7\}$ and the other contains the edge $\{7, 10\}$.