Spanning Trees

1. **Prove carefully that the Forest-Growing Algorithm given in class returns an MST, if it exists.**

   Call a forest *good* if it is contained in an MST. The empty graph that the algorithm starts with is clearly good. We will show inductively that all the forests occurring in the algorithm are good, hence also the last forest. Because the last forest satisfies \( |E(F)| = |V(G)| - 1 \), it must be a spanning tree, and then it is minimum because it is good.

   Let \( F \) be a forest in step 2, and suppose it is good, so contained in an MST \( T^* \). We want to show that \( F + e \) is good, if \( w(e) \) is minimum in \( S \) and \( F + e \) does not have a cycle. If \( e \in T^* \), then clearly \( F + e \) is good, so assume \( e \notin T^* \). Then adding \( e \) to \( T^* \) must create a cycle, which must contain an edge \( f \) not in \( F \), such that its endpoints are not in the same component of \( F \) (since \( e \) does not create a cycle in \( F \), its endpoints are in different components, so the vertices of the cycle cannot all be in the same component).

   This implies that \( f \) is still in \( S \): if not, then the algorithm must have previously considered \( f \), but rejected it because it would have created a cycle in a predecessor of \( F \). But this contradicts the fact that its endpoints are in different components of \( F \). Since \( f \in S \) the algorithm must have chosen \( e \) over \( f \), so \( w(f) \geq w(e) \). Then \( T^{**} = T^* - f + e \) is a minimum spanning tree containing \( F \) and \( e \), which means \( F + e \) is good.

   This proves that the final forest is good, hence an MST. To be careful, we should also check that the algorithm is correct when it returns “disconnected”. When it does, we must have \( |E(F)| < |V(G)| - 1 \), so it cannot be a tree, hence is disconnected.

2. **Give a greedy algorithm that finds a minimum spanning tree by removing edges while preserving connectedness. Prove that it works. Give an example to show that the same approach does not work for minimum spanning directed trees.**

   **Pruning Algorithm**
   
   1. Start with \( H = G \) and \( S = \emptyset \);
   2. Pick \( e \in E(H) \setminus S \) with maximum \( w(e) \); if \( E(H) \setminus S = \emptyset \), go to 4;
   3. If \( H - e \) is connected, then remove \( e \) from \( H \); else add \( e \) to \( S \); go to 2;
   4. If \( H \) is connected, return \( H \); else return “disconnected”.

   **Example:** Consider a graph \( G \) with two components \( C_1 \) and \( C_2 \) that are not connected. The pruning algorithm will select an edge \( e \) from \( C_1 \) to \( C_2 \) and add it to \( S \). Then \( H - e \) will consist of two disconnected components, so the algorithm returns “disconnected”.

   **Counterexample:** Consider a directed graph with two components \( C_1 \) and \( C_2 \), where there is a directed edge from \( C_1 \) to \( C_2 \) but no reverse edge. The pruning algorithm will select the directed edge and then return “disconnected”, which is incorrect since \( C_1 \) and \( C_2 \) are connected in the directed graph.
**Proof of correctness:** If $G$ is not connected, then the algorithm will just put every edge in $S$ and correctly return disconnected. If $G$ is connected, then $H$ will be connected throughout.

Call a connected subgraph of $G$ good if it contains an MST of $G$. We’ll show that the final $H$ is good. Since no edge could be removed without disconnecting it, the final $H$ is minimally connected, so a tree, and spanning because $V(H) = V(G)$ all along. It follows that the final $H$ is an MST.

Suppose we have a good $H$ in step 2, containing an MST $T^*$ of $G$. If we remove an $e \notin T^*$, then clearly $H - e$ is good, so assume $e \in T^*$. Write $e = uv$. Removing $e$ from $T^*$ must disconnect it into two components, say $U$ containing $u$ and $V$ containing $v$. Since $H - e$ is connected, there must be a path in it from $u$ to $v$, which must contain an edge $f$ with one endpoint in $U$ and one in $V$ (and $f \notin T^*$). We cannot have $f \in S$, since then $H - f$ should be disconnected, which it isn’t because it contains $T^*$. So $f \in E(H) \setminus S$, which implies $w(e) \geq w(f)$. Therefore $T^{**} = T^* - e + f$ is an MST contained in $H - e$, so $H - e$ is good.

**Directed trees:** Take for example

```
   10
   1
   1
```

Then removing the 10 would leave the graph connected (in the undirected sense), but there would no longer be a spanning directed tree.

You could avoid this by only removing an edge if afterwards there is still a spanning directed tree, which you can check using for instance breadth-first search. But this also fails:

```
   10
   1
   9
   1
```

If you remove the 10, there is still a spanning directed tree, but its weight is 19, whereas there is a spanning directed tree using 10 with weight 12.

3. **Prove that if its weight function $w$ is injective, then a graph has at most one minimum spanning tree.**

Suppose $w$ is injective and there are two distinct MSTs $T$ and $T^*$. Take $e \in T \setminus T^*$ and consider $T^* + e$. It must contain a cycle, which must contain an edge $f \in T^* \setminus T$. Now $w(e) \neq w(f)$, and we can assume $w(e) < w(f)$ without loss of generality. Then $T^* + e - f$ is a spanning tree with $w(T^* + e - f) < w(T^*)$, contradiction.
4. Give an algorithm that finds the second-smallest spanning tree in a weighted undirected graph (in other words, given a graph and an MST $T$, find the smallest among all spanning trees distinct from $T$). Prove that it works.

The question should maybe say “a second-smallest”, since there could be several MSTs, or the MST could be unique, with several second-smallest ones.

**Algorithm:** For each $e \in T$, find an MST in $G - e$. Compare the weights of the trees that you get, and take the smallest.

It’s polynomial, though of course you could implement it more efficiently, not running a complete MST algorithm every time.

That it works follows directly from the following fact:

**Fact:** Given an MST $T$, a second-smallest spanning tree can always be found among the trees of the form $T - e + f$, $e \in T$, $f \not\in T$.

**Proof:** First assume that $T$ is unique.

Let $T$ be an MST, and $T'$ a second-smallest spanning tree. Take $e \in T \setminus T'$ with minimum $w(e)$. Then adding $e$ to $T'$ creates a cycle, which contains an edge $f \in T' \setminus T$. We claim that $w(e) < w(f)$.

Suppose that $w(f) \leq w(e)$. Add $f$ to $T$, creating a cycle, which contains an edge $g \in T \setminus T'$. Now we have $w(g) < w(f)$, otherwise $w(T - g + f) \leq w(T)$, contradicting minimality or uniqueness of $T$. But then $w(g) < w(f) \leq w(e)$, contradicting our choice of $e$. This proves that $w(e) < w(f)$.

It follows that $w(T' - f + e) < w(T')$, so since $T'$ is second-smallest, we must have $T' - f + e = T$.

Suppose that $T$ is not unique.

Take another MST $T'$ with as many edges in common with $T$ as possible. Take $e \in T \setminus T'$, create a cycle in $T'$, containing $f \in T' \setminus T$. If $w(e) \neq w(f)$, then we’d get a smaller MST, so $w(e) = w(f)$. Then $T' - f + e$ is an MST having fewer edges in common with $T$, which implies $T' - f + e = T$. Done.