Exercise 1
Consider the reachability relation $\preceq_G$ on the nodes of a dag G, i.e. $u \preceq v$ iff there exists a path from u to v in G.

a) Prove that the $\preceq_G$ relation in a dag is a partial order.

b) What are the minimal and maximal elements of $\preceq_G$?

c) Suppose G is a directed forest. Under what conditions does $\preceq_G$ have a least element?

d) Can a dag be strongly connected? Explain your reasoning.

Solution:

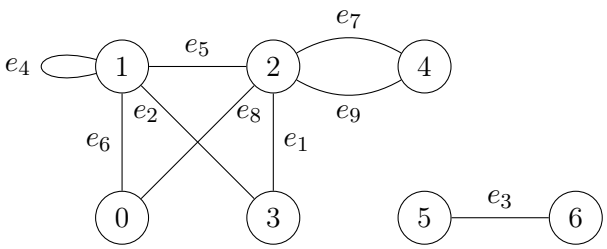
a)
- Reflexivity holds because every node is reachable from itself via the empty path.
- Transitivity holds because $u \preceq_G v$ and $v \preceq_G w$ then there exists a path p from u to v and a path q from v to w. Concatenating p and q yields a path from u to w, so $u \preceq_G w$.
- Antisymmetry holds because if $u \preceq_G v$ and $v \preceq_G u$ then there exists a path p from u to v and a path q from v to u. If one of those paths were non-empty, then their concatenation pq would be a cycle. But since G is a dag, this cannot be. Thus p and q are the empty path and thus $u = v$.

b) The minimal elements are nodes indegree 0 (if G is a directed tree, there is at most one and if there is one, it is a root). The maximal elements are the nodes with outdegree 0 (if G is a directed forest, we called these *leaves*).

c) A node is least with respect to $\preceq_G$ if all nodes are reachable from it. Since G is a directed forest, this is equivalent to G being an arborescence.

d) For a digraph G to be strongly connected, we need to have $x \preceq_G y$ and $y \preceq_G x$ for any nodes x and y. Since we have just shown that $\preceq$ is a partial order if G is a dag, this can clearly only be the case if G contains at most one node. In that case G is indeed strongly connected.

Exercise 2
Consider the following undirected graph G:



a) Apply Kruskal's algorithm to G. Traverse the edges in the order e_1 to e_9.

b) How many spanning forests of G are there?

c) All spanning forests of G have the same number of edges. Is this true in general? If not, give a counterexample. If yes, explain why.

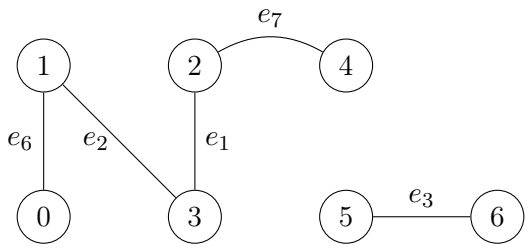
d) Argue why the algorithm would still work if we were to remove all loops and any parallel edges between two nodes (save for one) before running it.

Solution:

a) The set F and the connected components of F (to be more precise: of (V, F)) evolve as follows. To save space, we only write down non-singleton connected components and write sets without braces, commas, etc.

edges of F	components of F	edge being considered	edge added?
$\emptyset$		$e_1 = \{2, 3\}$	✓
e_1	23	$e_2 = \{1, 3\}$	✓
e_1, e_2	123	$e_3 = \{5, 6\}$	✓
e_1, e_2, e_3	123, 56	$e_4 = \{1\}$	✗
e_1, e_2, e_3	123, 56	$e_5 = \{1, 2\}$	✗
e_1, e_2, e_3	123, 56	$e_6 = \{0, 1\}$	✓
e_1, e_2, e_3, e_6	0123, 56	$e_7 = \{2, 4\}$	✓
e_1, e_2, e_3, e_6, e_7	01234, 56	$e_8 = \{0, 2\}$	✗
e_1, e_2, e_3, e_6, e_7	01234, 56	$e_9 = \{2, 4\}$	✗
e_1, e_2, e_3, e_6, e_7			

The spanning forest (V, F) looks like this:



b) e_4 cannot be part of a spanning forest and can be ignored. e_3 has to be part of any spanning forest.

As for the rest of the graph (nodes 0–4), any set of 4 edges that does not form a cycle will work. Clearly, we have to pick exactly one of e_7 and e_9. It remains to pick 3 of the 5 edges e_1, e_2, e_5, e_6, e_8. There are 10 ways to pick 3 elements of a 5-element set, but the combinations e_2, e_5, e_8 and e_5, e_6, e_8 do not work because they form a cycle. The remaining 8 ways do work.

That leaves us with $2 \cdot 8 = 16$ spanning forests (the 2 coming from the fact that we can choose between e_7 and e_9).

c) Yes, this generally holds. All spanning forests have the same nodes and the same connected components as G – say k. Thus any spanning forest F satisfies $|V| = |F| + k$, or, equivalently, $|F| = |V| - k$. In our case this amounts to $5 = 7 - 2$.

d) A loop can never be picked by the algorithm anyway because $u \sim_F u$ always holds, so removing it does not change anything. Moreover, if we have several parallel edges $e_{i_1}, e_{i_2}, \ldots$ between u and v, the algorithm will pick the first one in the list (say e_{i_1}) and ignore the remaining ones (since as soon as e_{i_1} is in F we have $u \sim_F v$). Thus if we remove all the edges between u and v except for e_{i_1}, nothing changes in the result.

Since the algorithm works no matter what order the edges are in, we can pick whichever one of the parallel edges we want by putting it first in the list.

Exercise 3

a) Give an undirected forest that does not have any leaves.

b) Prove: If $G = (V, E)$ is an undirected tree and $v \in V$ is a leaf, then the graph G' obtained by removing v and the edge e attached to it (if there is one) is again an undirected tree.

c) Prove: If $G = (V, E)$ is an undirected tree where V is finite and nonempty, then $|V| = |E| + 1$. **Hint:** Do an induction on the number of nodes.

d) If $G = (V, E)$ is a finite connected undirected graph with $|V| = |E| + 1$, is G always an undirected tree?

Solution:

a) From the lecture, we know that every non-empty finite undirected tree has a leaf. Thus every undirected forest that has at least one finite connected component has a leaf. It follows that when we look for examples of forests without leaves, there are only two kinds that we have to look at: the empty graph and infinite forests.

The empty graph is indeed an undirected forest with no leaves.

As for infinite, the simplest example is the following graph $G = (\mathbb{Z}, \{\{n, n+1\} \mid n \in \mathbb{Z}\})$:

$$\ldots \text{—} -2 \text{—} -1 \text{—} 0 \text{—} 1 \text{—} 2 \text{—} \ldots$$

b) It is clear that if G is cycle-free, then G' is also cycle-free (since any cycle in G' would automatically be a cycle in G). It remains to show that G' is still connected.

Let thus $x, y \in V \setminus \{v\}$ be two nodes in G'. Since G is a tree, there exists a simple path p from x to y in G. Clearly, v cannot occur on that path: it cannot be the start or end node of the path because $v \neq x$ and $v \neq y$. It cannot be an intermediate node either since there would then have to be one edge leading to v and one leading away from it (and then it would not be a leaf). Thus p is also a path in G'.

c) Following the hint, we show the statement by induction over $|V|$.

For the induction step, let $G = (V, E)$ be a finite non-empty undirected tree. If $|V| = 1$, then obviously $E = \emptyset$ and the statement holds. Otherwise let v be a leaf of G. Since G has at least 2 nodes and is connected, there must be an edge e attached to v. Let $G' = (V \setminus \{v\}, E \setminus \{e\})$.

From a) we know that G' is again a tree, and it is non-empty. Since $V \setminus \{v\} \subset V$ we can apply the induction hypothesis to obtain $|V \setminus \{v\}| = |E \setminus \{e\}| + 1$ and thus $|V| - 1 = |E| - 1 + 1$ and from that $|V| = |E| + 1$ as desired.

Note: An alternative proof would be to turn G into an arborescence by picking an arbitrary leaf of G as a root and converting all the undirected edges of G as directed ones pointing away from the root. Then we can use that $|V| = |E| + 1$ for an arborescence, which we proved last week.

d) Yes. We need only show that G is cycle-free. If G were not cycle-free then we could get a spanning forest $G' = (V, F)$ of G. Since G is connected, so is G', i.e. G' is a tree and thus $|V| = |F| + 1$. But then $|E| = |F|$, and with $F \subseteq E$ it follows that $E = F$ and thus $G = G'$.

Bonus exercise

Let $G = (V, E)$ be an undirected graph and $v \in V$ and $w \notin V$. Define $G' = (V \cup \{w\}, E \cup \{v, w\})$.

a) Prove or refute: If G is an undirected forest, so is G'.

b) Prove or refute: If G is an undirected tree, so is G'.

Solution:

a) Since G is cycle-free, any cycle in G' would have to contain the new edge $\{v, w\}$, which is not possible since a cycle cannot contain leaves: every node on a cycle is incident to two edges from the cycle, and since cycles are simple those are two different edges.

b) Cycle-freeness follows from a). It remains to show that G' is connected. If $x, y \in V$ then $x \sim_G y$ and thus also $x \sim_{G'} y$. Furthermore, if $x \in V$, then $x \sim_{G'} w$ since $x \sim_G v$ and thus $x \sim_{G'} v$ and in combination with $v \sim_{G'} w$ we get $x \sim_{G'} w$.