9.2. Terminology and Characterizations of Trees

Exercise 1. Let $T$ be a tree, and let $P$ be a path in $T$ that doesn’t repeat edges. Prove that $P$ doesn’t repeat vertices.

Exercise 2. Let $T$ be a tree. Prove that if an edge is added between two vertices, exactly one cycle is created. Consider two cycles to be the same if they use the same edges, regardless of ordering. (See also Exercise 9.2.34.)

9.3. Spanning Trees

Exercise 3 (9.3.3). Use breadth-first search with vertex ordering $hfdbgeca$ to find a spanning tree for the graph on the left.

Exercise 4 (9.3.6). Use depth-first search with vertex ordering $hfdbgeca$ to find a spanning tree for the graph on the right.

9.4. Minimal Spanning Trees

Exercise 5 (9.4.10 and 11). Let $G$ be a connected, weighted graph, let $v$ be a vertex in $G$, and let $e$ be an edge of minimum weight incident on $v$. Show that $e$ is contained in some minimal spanning tree.

If the weights of edges incident on $v$ are distinct, must $e$ be contained in every minimal spanning tree?
Solutions

Solution to Exercise 1. Write

\[ P = (v_1, e_1, v_2, e_2, \ldots, e_n, v_{n+1}) \]

Suppose for some \( i \neq j \) we have \( v_i = v_j \). We may assume \( i < j \). Then \((v_i, e_i, \ldots, v_{j-1}, e_{j-1}, v_j)\) is a cycle, since edges are not repeated. This contradicts the assumption that \( T \) is a tree. \( \square \)

Solution to Exercise 2. Suppose an edge \( e \) is added between vertices \( v \) and \( w \). There is already a (unique) simple path \( P \) in \( T \) from \( v \) to \( w \), by definition of a tree. Taking this path followed by edge \( e \) back to \( v \) gives a cycle. Note that \( P \) does not repeat edges because it is a simple path, and does not use edge \( e \) since \( e \not\in T \); therefore even after adding \( e \) no edges are repeated. Call this cycle \( C_0 \).

Why are no more cycles created? Suppose \( C \) is a cycle in the new graph. \( C \) must use edge \( e \) – otherwise \( C \) would be a cycle in the tree \( T \). Since cycles don’t repeat edges, \( e \) is used exactly once; therefore we can write (possibly after reordering, which we don’t care about here)

\[ C = (v, \ldots, w, e, v) \]

Then \((v, \ldots, w)\) is a simple path in \( T \) from \( v \) to \( w \), which must be the same simple path \( P \) from before, so \( C = C_0 \).

Solutions to Exercises 3 and 4.

The numbers indicate at which stage of the algorithm the edge is added to the tree.

Solution to Exercise 5. Write \( e = (v, w) \) and order the vertices starting with \( v, w, \ldots \). Then \( e \) will be the first edge added by Prim’s algorithm, since it is the edge of minimal weight incident on the starting vertex (having \( w \) come first in the ordering after \( v \) ensures that if some other edge also has minimal weight we’ll still use \( e \)).

If the edge weights are distinct: suppose \( T \) is a spanning tree which does not contain \( e \). Then inserting \( e \) creates a unique cycle, which must use some second edge \( e' \) incident on \( v \).
Removing $e'$ breaks this cycle (without removing any vertices), so that $T' = (T \cup \{e\}) \setminus \{e'\}$ is a spanning tree.

Since $w(e) < w(e')$, the new tree $T'$ has strictly smaller weight than $T$. Therefore $T$ must not have been a minimal spanning tree.