1. 2.2.1. Determine which trees have Prufer codes that

a) contain only one value,

b) contain exactly two values, or

c) have distinct values in all positions.

Solution, part a:
Stars have only one value in their Prufer code.

Solution, part b:
Double brooms have exactly two values in their Prufer code.

Solution, part c:
Trees with only two leaves have distinct values in all positions of their Prufer code.

2. 2.2.7. Use Cayley's Formula to prove that the graph obtained from K_n by deleting an edge has $(n-2)n^{n-3}$ spanning trees.

Solution:
Consider the endpoints u and v of the edge that is deleted from K_n. Then $d(u) = d(v) = n-2$. Without loss of generality, suppose we delete v. The resulting graph is K_{n-1}, which has 2^{n-3} spanning trees by Cayley's Formula. Adding back v and the $n-2$ edges incident to it yields the original graph. Hence, since there are $n-2$ possibilities for connecting v to the spanning trees in K_{n-1}, the original graph has $(n-2)2^{n-3}$ spanning trees. □

3. 2.2.8. Count the following sets of trees with vertex set $[n]$, giving two proofs for each: one using the Prufer correspondence and one by direct counting arguments

a) trees that have 2 leaves

b) trees that have n-2 leaves

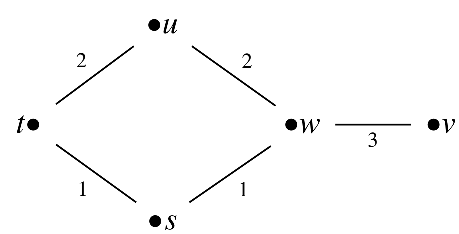
Solution, part a:
First, direct counting. A tree with exactly 2 leaves on n vertices is P_n. Thus, there are $n!$ ways to arrange the vertices. Since reversing the order of the vertices produces the same tree, there are $n!/2$ trees with vertex set $[n]$ that have 2 leaves. Now, the Prufer correspondence counting. Note that all the entries of the Prufer code will be distinct. Then, applying the Prufer code, there are n options for the first entry, then $n-1$ choices for the next one, and $n-3$ for the next, etc. This continues until there are 2 vertices left. Hence, there a $n!/2$ trees that have 2 leaves.

Solution, part b:
First, Prufer correspondence counting. Note that trees that have $n-2$ leaves are double brooms. Thus, there are exactly two values in the Prufer code. There are n options for the first entry, then $n-1$ options for the second entry. For the remaining $n-4$ slots

in the Prufer code, each has 2 choices. Hence, there are $n(n-1)2^{n-4}$ trees that have $n-2$ leaves. Now we count directly. There are $\binom{n}{2}$ ways to pick the two centers of the double broom. There are $n-2$ neighboring vertices left to assign. There are 2 choices for each neighbor, either one center or the other. Yet, since flipping the graph over its double center produces the same graph, $\binom{n}{2}2^{n-2}$ overcounts by a factor of 2. Hence, there are $\binom{n}{2}2^{n-3}$ trees that have $n-2$ leaves, which equivalent to the result from Prufer correspondence. □

4. 2.3.2. Prove or disprove: If T is a minimum-weight spanning tree of a weighted graph G, then the u, v-path in T is a minimum-weight u, v-path in G.

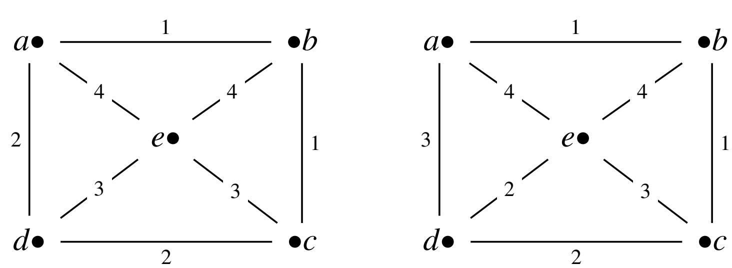
Solution:
False. Let G be the graph below. $T = \{vw, ws, st, tu\}$ is a minimum weight spanning tree for G. Yet, on T, $d(u, v) = 7$ while on G, $d(u, v) = 5$.



□

5. 2.3.4. In the graph below, assign weights (1,1,2,2,3,3,4,4) in two ways one way so that the minimum-weight spanning tree is unique, and another way so that the minimum-weight spanning tree is not unique.

Solution:
In the left graph below, $T = \{ab, bc, ce, cd\}$ and $T' = \{ab, bc, cd, de\}$ are distinct minimum weight spanning trees. In the right graph below, $T' = \{ab, bc, cd, de\}$ is the only minimum weight spanning tree.



6. 2.3.9. Let F be a spanning forest of a connected weighted graph G. Among all edges of G having endpoints in different components of F, let e be one of minimum weight. Prove that among all the spanning trees of G that contain F, there is one of minimum weight that contains e. Use this to give another proof that Kruskal's Algorithm works.

Solution:
Let T be a minimum weight spanning forest containing F. Consider two components, C and C' of F connected by e in G. If T connects C to C' directly, then it either uses e or another edge with the same weight as e. Without loss of generality, suppose in that case that T uses e. Then, to connect C to C', T either uses e or there is a path through other components of F not containing e of equal or lesser weight compared to e. Hence, a path from C to C' in T that doesn't use e must go through other components using connecting edges that have weights equal to or greater than the weight of e. Even if the edges between components of F that T uses are of the same weight as e, T would have to use at least two of them to connect C to C' because it is moving through at least one extra component in F. Hence, this path will be of more weight than a path containing e. Therefore, T is a minimum weight spanning tree containing F that contains e. To show that Kruskal's algorithm works, we consider a spanning forest F of G that has no edges. This is the same start as Kruskal's algorithm. Then, we add an edge e that reduces the number of components in F and is of minimum weight. By the above proof, there must be a minimum weight spanning tree containing e. This will be true for each step of Kruskal's algorithm by the above proof. Hence, Kruskal's algorithm must be true.