Small world networks have the property that characteristic path lengths are low and clustering coefficients are high. Graphs that have these properties can be used as models in the mathematical analyses of the small world phenomenon and its associated concepts. It is interesting to note that other well known networks have exhibited small world traits — the internet, electric power grids, and even neural networks are examples — and this increases even further the applicability of graph models.

Exercises

1. Compute the characteristic path length for each of each of the following graphs: P_{2k}, P_{2k+1}, C_{2k}, C_{2k+1}, K_n, $K_{m,n}$.

2. Compute the clustering coefficient for each of each of the following graphs: P_{2k}, P_{2k+1}, C_{2k}, C_{2k+1}, K_n, $K_{m,n}$.

3. (a) In the Acquaintance Graph, try to find a path from your vertex to the vertex of the President of the United States.

   (b) Your path from the previous question may not be your shortest such path. Prove that your actual distance from the President is at most one away from the shortest such distance to be found among your classmates.

Interesting Note: There are several contexts in which Bacon numbers can be calculuated. While Bacon purists only use movie connections, others include shared appearances on television and in documentaries as well. Under these more open guidelines, the mathematician Paul Erdős actually has a Bacon number of 3! Erdős was the focus of the 1993 documentary *N is a Number* [63]. British actor Alec Guinness made a (very) brief appearance near the beginning of that film, and Guinness has a Bacon number of 2 (can you find the connections?). As far as we know, Bacon has not coauthored a research article with anyone who is connected to Erdős, and so while Erdős' Bacon number is 3, Bacon's Erdős number is infinity.

Textbook: "Combinatorics and Graph Theory" by Harris, Hirst, Mossinghoff

1.3 Trees

"O look at the trees!" they cried, "O look at the trees!"
— Robert Bridges, *London Snow*

In this section we will look at the trees—but not the ones that sway in the wind or catch the falling snow. We will talk about graph-theoretic trees. Before moving on, glance ahead at Figure 1.30, and try to pick out which graphs are trees.

1.3.1 Definitions and Examples

Example, the surest method of instruction.

— Pliny the Younger

In Figure 1.30 graphs A, B, and E are trees, while graphs C and D are not.

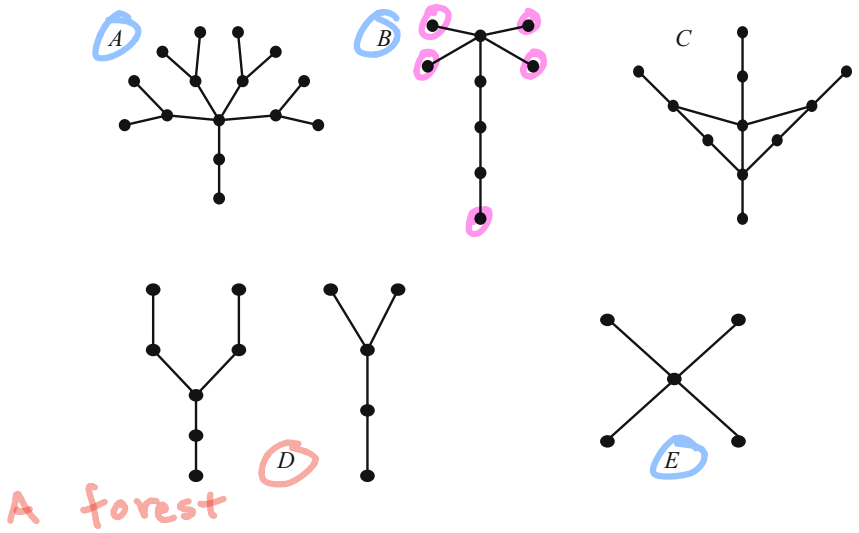


FIGURE 1.30. Which ones are trees?

A *tree* is a connected graph that contains no cycles. Graph-theoretic trees resemble the trees we see outside our windows. For example, graph-theoretic trees do not have cycles, just as the branches of trees in nature do not split and rejoin. The descriptive terminology does not stop here.

Graph D in Figure 1.30 is not a tree; rather, it is a *forest*. A forest is a collection of one or more trees. A vertex of degree 1 in a tree is called a *leaf*.

As in nature, graph-theoretic trees come in many shapes and sizes. They can be thin (P_{10}) or thick ($K_{1,1000}$), tall (P_{1000}) or short (K_1 and K_2). Yes, even the graphs K_1 and K_2 are considered trees (they are certainly connected and acyclic). In the spirit of our arboreal terminology, perhaps we should call K_1 a *stump* and K_2 a *twig*!

While we are on the subject of small trees, we should count a few of them. It is clear that K_1 and K_2 are the only trees of order 1 and 2, respectively. A moment's thought will reveal that P_3 is the only tree of order 3. Figure 1.31 shows the different trees of order 6 or less.

Trees sprout up as effective models in a wide variety of applications. We mention a few brief examples.

Examples

1. Trees are useful for modeling the possible outcomes of an experiment. For example, consider an experiment in which a coin is flipped and a 6-sided die is rolled. The leaves in the tree in Figure 1.32 correspond to the outcomes in the probability space for this experiment.

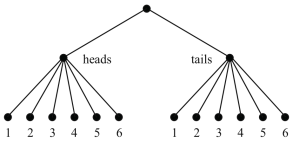


FIGURE 1.32. Outcomes of a coin/die experiment.

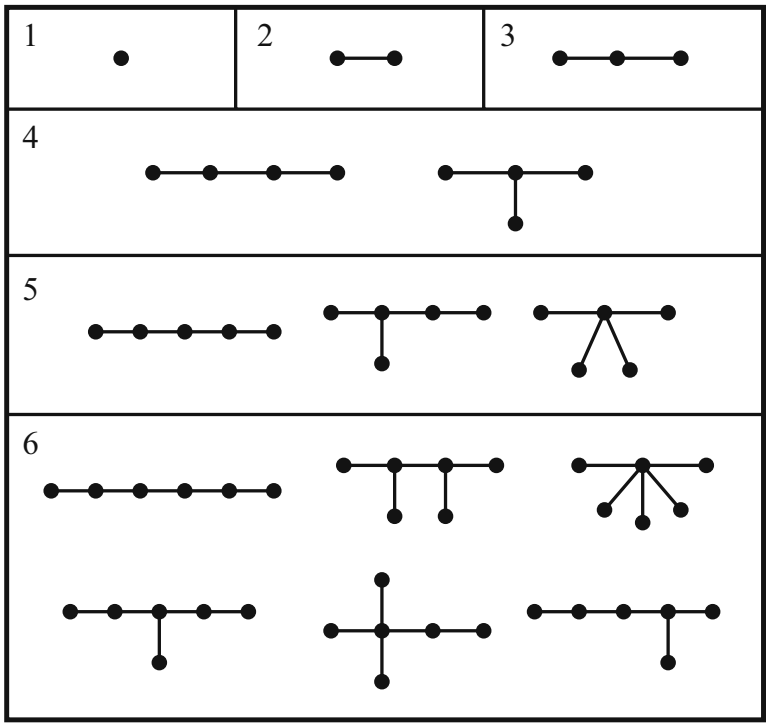


FIGURE 1.31. Trees of order 6 or less.

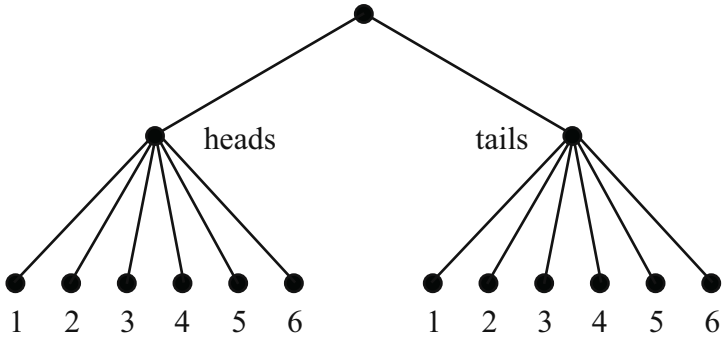


FIGURE 1.32. Outcomes of a coin/die experiment.

2. Programmers often use tree structures to facilitate searches and sorts and to model the logic of algorithms. For instance, the logic for a program that finds the maximum of four numbers (w, x, y, z) can be represented by the tree shown in Figure 1.33. This type of tree is a *binary decision tree*.

3. Chemists can use trees to represent, among other things, saturated hydrocarbons—chemical compounds of the form C_nH_{2n+2} (propane, for example). The bonds between the carbon and hydrogen atoms are depicted in the trees of Figure 1.34. The vertices of degree 4 are the carbon atoms, and the leaves represent the hydrogen atoms.

4. College basketball fans will recognize the tree in Figure 1.35. It displays final results for the "Sweet 16" portion of the 2008 NCAA men's basketball tournament. Each vertex represents a single game.

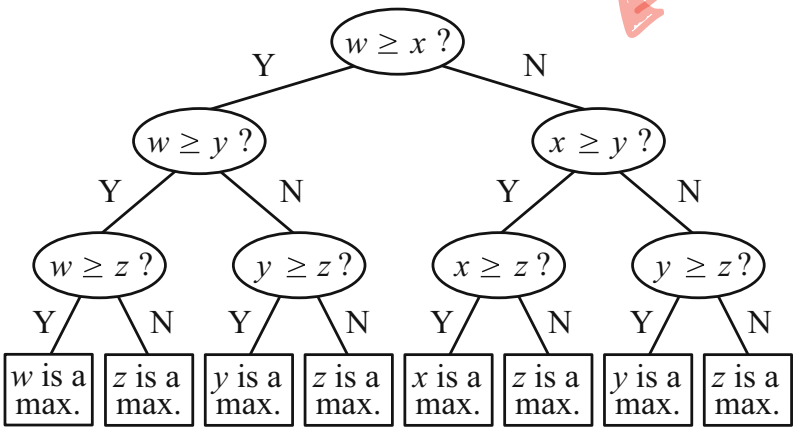


FIGURE 1.33. Logic of a program.

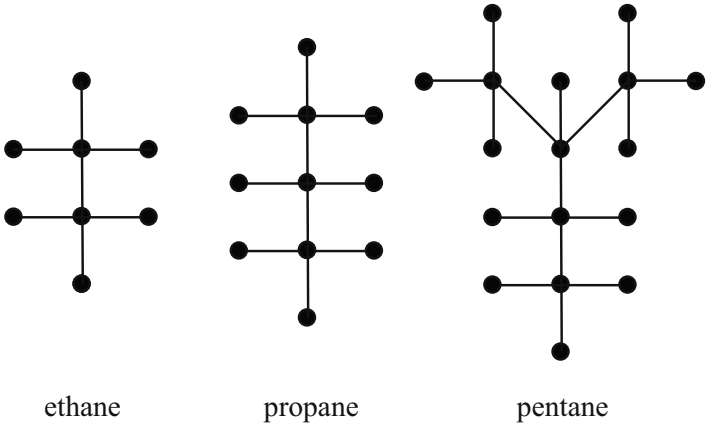


FIGURE 1.34. A few saturated hydrocarbons.

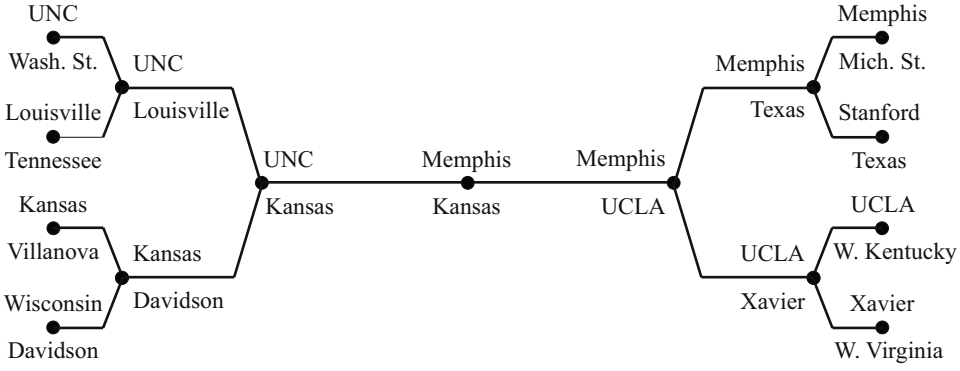


FIGURE 1.35. The 2008 Men's Sweet 16.

Exercises

Sec 1.3.1 Definitions and Examples

1. Draw all unlabeled trees of order 7. Hint: There are a prime number of them.

2. Draw all unlabeled forests of order 6.

3. Let T be a tree of order $n \geq 2$. Prove that T is bipartite.

4. Graphs of the form $K_{1,n}$ are called *stars*. Prove that if $K_{r,s}$ is a tree, then it must be a star.

5. Match the graphs in Figure 1.36 with appropriate names: a palm tree, autumn, a path through a forest, tea leaves.

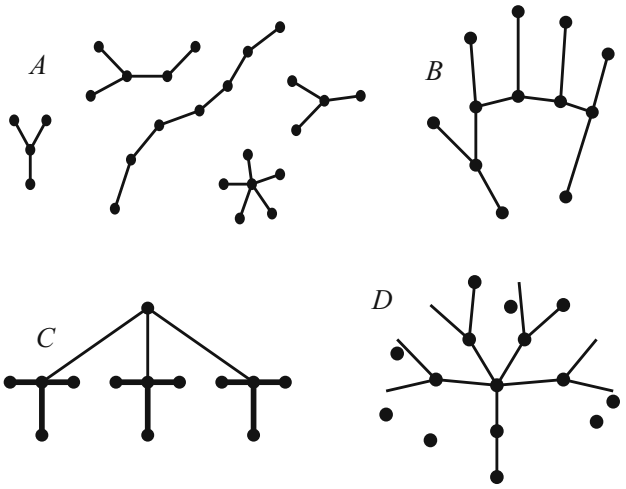
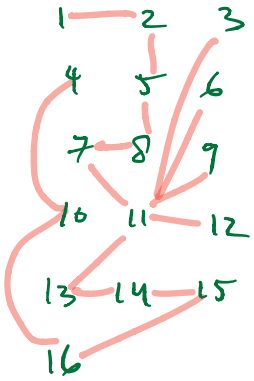


FIGURE 1.36. What would you name these graphs?

1.3.2 Properties of Trees

And the tree was happy.

— Shel Silverstein, *The Giving Tree*



Let us try an experiment. On a piece of scratch paper, draw a tree of order 16. Got one? Now count the number of edges in the tree. We are going to go out on a limb here and predict that there are 15. Since there are nearly 20,000 different trees of order 16, it may seem surprising that our prediction was correct. The next theorem gives away our secret.

Theorem 1.10. *If T is a tree of order n, then T has $n-1$ edges.*

Proof. We induct on the order of T. For $n = 1$ the only tree is the stump (K_1), and it of course has 0 edges. Assume that the result is true for all trees of order less than k, and let T be a tree of order k.

Choose some edge of T and call it e. Since T is a tree, it must be that $T - e$ is disconnected (see Exercise 7) with two connected components that are trees themselves (see Figure 1.37). Say that these two components of $T - e$ are T_1 and T_2, with orders k_1 and k_2, respectively. Thus, k_1 and k_2 are less than k and $k_1 + k_2 = k$.

Since $k_1 < k$, the theorem is true for T_1. Thus T_1 has $k_1 - 1$ edges. Similarly, T_2 has $k_2 - 1$ edges. Now, since $E(T)$ is the disjoint union of $E(T_1)$, $E(T_2)$, and $\{e\}$, we have $|E(T)| = (k_1 - 1) + (k_2 - 1) + 1 = k_1 + k_2 - 1 = k - 1$. This completes the induction. □

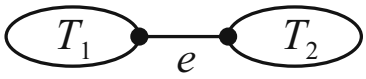


FIGURE 1.37.

The next theorem extends the preceding result to forests. The proof is similar and appears as Exercise 4.

Theorem 1.11. *If F is a forest of order n containing k connected components, then F contains $n - k$ edges.*

The next two theorems give alternative methods for defining trees. Two other methods are given in Exercises 5 and 6.

Theorem 1.12. *A graph of order n is a tree if and only if it is connected and contains $n - 1$ edges.*

Proof. The forward direction of this theorem is immediate from the definition of trees and Theorem 1.10. For the reverse direction, suppose a graph G of order n is connected and contains $n - 1$ edges. We need to show that G is acyclic. If G did have a cycle, we could remove an edge from the cycle and the resulting graph would still be connected. In fact, we could keep removing edges (one at a time) from existing cycles, each time maintaining connectivity. The resulting graph would be connected and acyclic and would thus be a tree. But this tree would have fewer than $n - 1$ edges, and this is impossible by Theorem 1.10. Therefore, G has no cycles, so G is a tree. □

Theorem 1.13. *A graph of order n is a tree if and only if it is acyclic and contains $n - 1$ edges.*

Proof. Again the forward direction of this theorem follows from the definition of trees and from Theorem 1.10. So suppose that G is acyclic and has $n - 1$ edges. To show that G is a tree we need to show only that it is connected. Let us say that the connected components of G are $G_1, G_2, \ldots, G_k$. Since G is acyclic, each of these components is a tree, and so G is a forest. Theorem 1.11 tells us that G has $n - k$ edges, implying that $k = 1$. Thus G has only one connected component, implying that G is a tree. □

It is not uncommon to look out a window and see leafless trees. In graph theory, though, leafless trees are rare indeed. In fact, the stump (K_1) is the only such tree, and every other tree has at least two leaves. Take note of the proof technique of the following theorem. It is a standard graph theory induction argument.

Theorem 1.14. *Let T be the tree of order $n \geq 2$. Then T has at least two leaves.*

Proof. Again we induct on the order. The result is certainly true if $n = 2$, since $T = K_2$ in this case. Suppose the result is true for all orders from 2 to $n - 1$, and consider a tree T of order $n \geq 3$. We know that T has $n - 1$ edges, and since we can assume $n \geq 3$, T has at least 2 edges. If every edge of T is incident with

skip

a leaf, then T has at least two leaves, and the proof is complete. So assume that there is some edge of T that is not incident with a leaf, and let us say that this edge is $e = uv$. The graph $T - e$ is a pair of trees, T_1 and T_2, each of order less than n. Let us say, without loss of generality, that $u \in V(T_1)$, $v \in V(T_2)$, $|V(T_1)| = n_1$, and $|V(T_2)| = n_2$ (see Figure 1.38). Since e is not incident with any leaves of T,

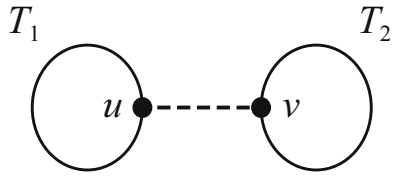


FIGURE 1.38.

we know that n_1 and n_2 are both at least 2, so the induction hypothesis applies to each of T_1 and T_2. Thus, each of T_1 and T_2 has two leaves. This means that each of T_1 and T_2 has at least one leaf that is not incident with the edge e. Thus the graph $(T - e) + e = T$ has at least two leaves. □

We saw in the previous section that the center of a graph is the set of vertices with minimum eccentricity. The next theorem, due to Jordan [170], shows that for trees, there are only two possibilities for the center.

Theorem 1.15. *In any tree, the center is either a single vertex or a pair of adjacent vertices.*

Proof. Given a tree T, we form a sequence of trees as follows. Let $T_0 = T$. Let T_1 be the graph obtained from T_0 by deleting all of its leaves. Note here that T_1 is also a tree. Let T_2 be the tree obtained from T_1 by deleting all of the leaves of T_1. In general, for as long as it is possible, let T_j be the tree obtained by deleting all of the leaves of T_{j-1}. Since T is finite, there must be an integer r such that T_r is either K_1 or K_2.

Consider now a consecutive pair T_i, T_{i+1} of trees from the sequence $T = T_0$, $T_1, \ldots, T_r$. Let v be a non-leaf of T_i. In T_i, the vertices that are at the greatest distance from v are leaves (of T_i). This means that the eccentricity of v in T_{i+1} is one less than the eccentricity of v in T_i. Since this is true for all non-leaves of T_i, it must be that the center of T_{i+1} is exactly the same as the center of T_i.

Therefore, the center of T_r is the center of T_{r-1}, which is the center of T_{r-2}, $\ldots$, which is the center of $T_0 = T$. Since (the center of) T_r is either K_1 or K_2, the proof is complete. □

We conclude this section with an interesting result about trees as subgraphs.

Theorem 1.16. *Let T be a tree with k edges. If G is a graph whose minimum degree satisfies $\delta(G) \geq k$, then G contains T as a subgraph. Alternatively, G contains every tree of order at most $\delta(G) + 1$ as a subgraph.*

skip

Proof. We induct on k. If $k = 0$, then $T = K_1$, and it is clear that K_1 is a subgraph of any graph. Further, if $k = 1$, then $T = K_2$, and K_2 is a subgraph of any graph whose minimum degree is 1. Assume that the result is true for all trees with $k - 1$ edges ($k \geq 2$), and consider a tree T with exactly k edges. We know from Theorem 1.14 that T contains at least two leaves. Let v be one of them, and let w be the vertex that is adjacent to v. Consider the graph $T - v$. Since $T - v$

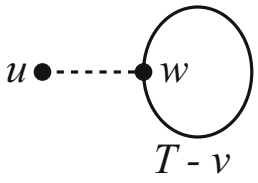


FIGURE 1.39.

has $k - 1$ edges, the induction hypothesis applies, so $T - v$ is a subgraph of G. We can think of $T - v$ as actually sitting inside of G (meaning w is a vertex of G, too). Now, since G contains at least $k + 1$ vertices and $T - v$ contains k vertices, there exist vertices of G that are not a part of the subgraph $T - v$. Further, since the degree in G of w is at least k, there must be a vertex u not in $T - v$ that is adjacent to w (Figure 1.40). The subgraph $T - v$ together with u forms the tree T

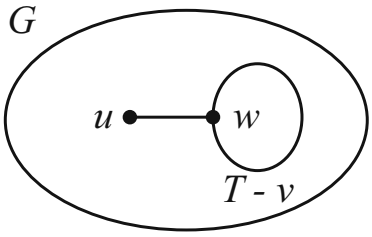


FIGURE 1.40. A copy of T inside G.

as a subgraph of G. □

Exercises

1. Draw each of the following, if you can. If you cannot, explain the reason.
   (a) A 10-vertex forest with exactly 12 edges
   (b) A 12-vertex forest with exactly 10 edges
   (c) A 14-vertex forest with exactly 14 edges
   (d) A 14-vertex forest with exactly 13 edges
   (e) A 14-vertex forest with exactly 12 edges
2. Suppose a tree T has an even number of edges. Show that at least one vertex must have even degree.
3. Let T be a tree with max degree Δ. Prove that T has at least Δ leaves.

4. Let F be a forest of order n containing k connected components. Prove that F contains $n - k$ edges.

5. Prove that a graph G is a tree if and only if for every pair of vertices u, v, there is exactly one path from u to v.

6. Prove that T is a tree if and only if T contains no cycles, and for any new edge e, the graph $T + e$ has exactly one cycle.

7. Show that every edge in a tree is a bridge.

8. Show that every nonleaf in a tree is a cut vertex.

9. Find a shorter proof to Theorem 1.14. Hint: Start by considering a longest path in T.

10. Let T be a tree of order $n > 1$. Show that the number of leaves is

$$2 + \sum_{\deg(v_i) \geq 3} (\deg(v_i) - 2),$$

where the sum is over all vertices of degree 3 or more.

11. For a graph G, define the average degree of G to be

$$\text{avgdeg}(G) = \frac{\sum_{v \in V(G)} \deg(v)}{|V(G)|}.$$

If T is a tree and $\text{avgdeg}(T) = a$, then find an expression for the number of vertices in T in terms of a.

12. Let T be a tree such that every vertex adjacent to a leaf has degree at least 3. Prove that some pair of leaves in T has a common neighbor.

1.3.3 Spanning Trees

Under the spreading chestnut tree ...
— Henry W. Longfellow, *The Village Blacksmith*

The North Carolina Department of Transportation (NCDOT) has decided to implement a rapid rail system to serve eight cities in the western part of the state. Some of the cities are currently joined by roads or highways, and the state plans to lay the track right along these roads. Due to the mountainous terrain, some of the roads are steep and curvy; and so laying track along these roads would be difficult and expensive. The NCDOT hired a consultant to study the roads and to assign difficulty ratings to each one. The rating accounted for length, grade, and curviness of the roads; and higher ratings correspond to greater cost. The graph

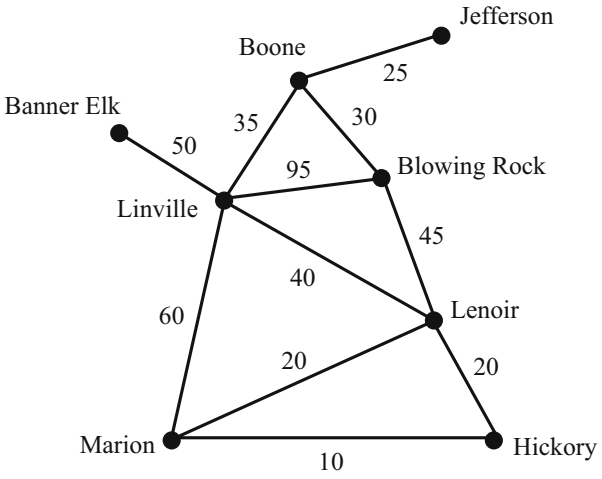
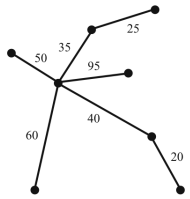
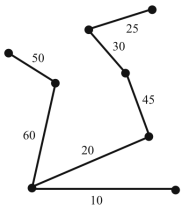


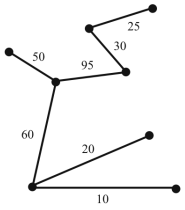
FIGURE 1.41. The city graph.



Total Weight = 325



Total Weight = 240



Total Weight = 290

FIGURE 1.42. Several spanning trees. of the city graph

in Figure 1.41, call it the "city graph," shows the result of the consultant's investigation. The number on each edge represents the difficulty rating assigned to the existing road.

The state wants to be able to make each city accessible (but not necessarily directly accessible) from every other city. One obvious way to do this is to lay track along every one of the existing roads. But the state wants to minimize cost, so this solution is certainly not the best, since it would result in a large amount of unnecessary track. In fact, the best solution will not include a cycle of track anywhere, since a cycle would mean at least one segment of wasted track.

The situation above motivates a definition. Given a graph G and a subgraph T, we say that T is a *spanning tree* of G if T is a tree that contains every vertex of G.

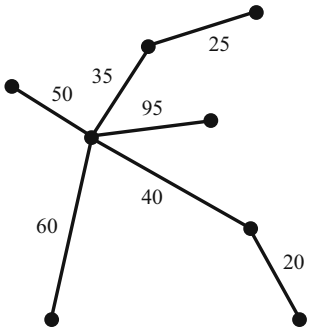
So it looks as though the DOT just needs to find a spanning tree of the city graph, and they would like to find one whose overall rating is as small as possible. Figure 1.42 shows several attempts at a solution.

Of the solutions in the figure, the one in the upper right has the least total weight—but is it the best one overall? Try to find a better one. We will come back to this problem soon.

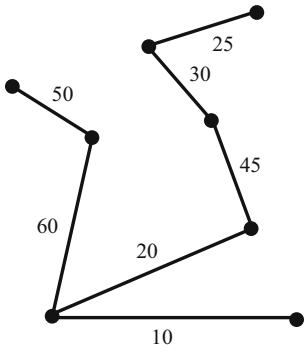
Given a graph G, a *weight function* is a function W that maps the edges of G to the nonnegative real numbers. The graph G together with a weight function is called a *weighted graph* . The graph in Figure 1.41 is a simple example of a weighted graph. Although one might encounter situations where negative valued weights would be appropriate, we will stick with nonnegative weights in our discussion.

It should be fairly clear that every connected graph has at least one spanning tree. In fact, it is not uncommon for a graph to have many different spanning trees. Figure 1.42 displays three different spanning trees of the city graph.

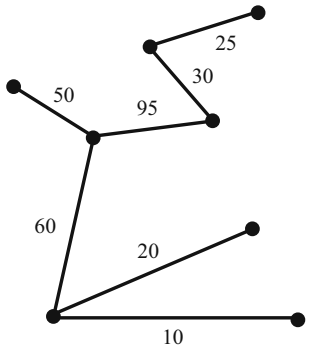
Given a connected, weighted graph G, a spanning tree T is called a *minimum weight spanning tree* if the sum of the weights of the edges of T is no more than the sum for any other spanning tree of G.



Total Weight = 325



Total Weight = 240



Total Weight = 290

FIGURE 1.42. Several spanning trees.

There are a number of fairly simple algorithms for finding minimum weight spanning trees. Perhaps the best known is Kruskal's algorithm.

Kruskal's Algorithm

Given: A connected, weighted graph G.

i. Find an edge of minimum weight and mark it.

ii. Among all of the unmarked edges that do not form a cycle with any of the marked edges, choose an edge of minimum weight and mark it.

iii. If the set of marked edges forms a spanning tree of G, then stop. If not, repeat step ii.

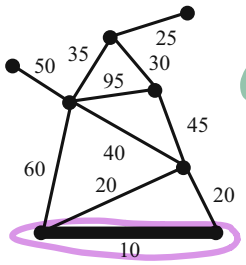
Figure 1.43 demonstrates Kruskal's algorithm applied to the city graph. The minimum weight is 210.

It is certainly possible for different trees to result from two different applications of Kruskal's algorithm. For instance, in the second step we could have chosen the edge between Marion and Lenoir instead of the one that was chosen. Even so, the total weight of resulting trees is the same, and each such tree is a minimum weight spanning tree.

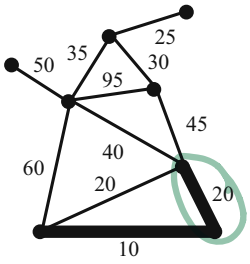
Kruskal's Algorithm Note: This is an example of a greedy algorithm

Given: A connected, weighted graph G.

i. Find an edge of minimum weight and mark it.



ii. Among all of the unmarked edges that do not form a cycle with any of the marked edges, choose an edge of minimum weight and mark it.



iii. If the set of marked edges forms a spanning tree of G, then stop. If not, repeat step ii.

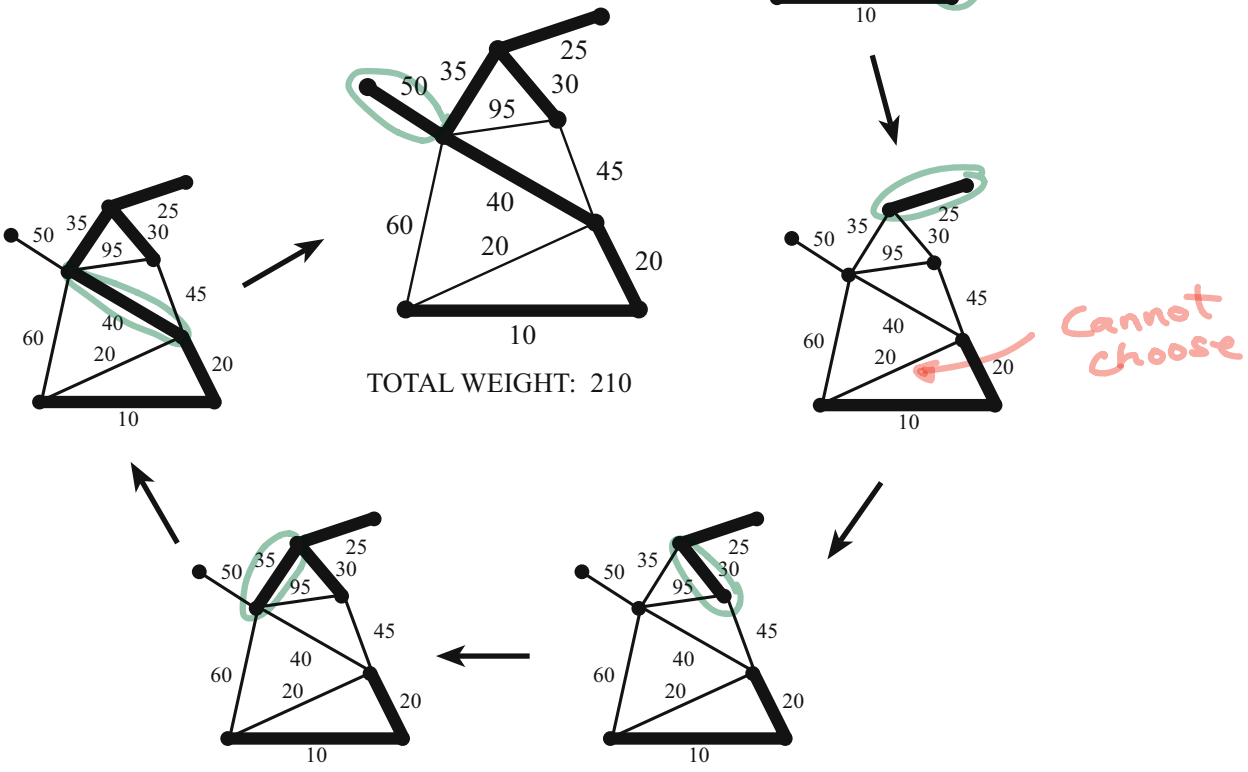


FIGURE 1.43. The stages of Kruskal's algorithm.

It should be clear from the algorithm itself that the subgraph built is in fact a spanning tree of G. How can we be sure, though, that it has minimum total weight? The following theorem answers our question [183].

Theorem 1.17. *Kruskal's algorithm produces a spanning tree of minimum total weight.*

Proof. Let G be a connected, weighted graph of order n, and let T be a spanning tree obtained by applying Kruskal's algorithm to G. As we have seen, Kruskal's algorithm builds spanning trees by adding one edge at a time until a tree is formed. Let us say that the edges added for T were (in order) $e_1, e_2, \ldots, e_{n-1}$. Suppose T is *not* a minimum weight spanning tree. Among all minimum weight spanning trees of G, choose T' to be a minimum weight spanning tree that agrees with the construction of T for the longest time (i.e., for the most initial steps). This

means that there exists some k such that T' contains $e_1, \ldots, e_k$, and no minimum weight spanning tree contains all of $e_1, \ldots, e_k, e_{k+1}$ (notice that since T is not of minimum weight, $k < n - 1$).

Since T' is a spanning tree, it must be that $T' + e_{k+1}$ contains a cycle C, and since T contains no cycles, C must contain some edge, say e', that is *not* in T. If we remove the edge e' from $T' + e_{k+1}$, then the cycle C is broken and what remains is a spanning tree of G. Thus, $T' + e_{k+1} - e'$ is a spanning tree of G, and it contains edges $e_1, \ldots, e_k, e_{k+1}$. Furthermore, since the edge e' must have been available to be chosen when e_{k+1} was chosen by the algorithm, it must be that $w(e_{k+1}) \leq w(e')$. This means that $T' + e_{k+1} - e'$ is a spanning tree with weight no more than T' that contains edges $e_1, \ldots, e_{k+1}$, contradicting our assumptions. Therefore, it must be that T is a minimum weight spanning tree. □

Exercises Sec 1.3.3 Spanning Trees

1. Prove that every connected graph contains at least one spanning tree.

2. Prove that a graph is a tree if and only if it is connected and has exactly one spanning tree.

3. Let G be a connected graph with n vertices and at least n edges. Let C be a cycle of G. Prove that if T is a spanning tree of G, then $\overline{T}$, the complement of T, contains at least one edge of C.

4. Let G be connected, and let e be an edge of G. Prove that e is a bridge if and only if it is in every spanning tree of G.

5. Using Kruskal's algorithm, find a minimum weight spanning tree of the graphs in Figure 1.44. In each case, determine (with proof) whether the minimum weight spanning tree is unique.

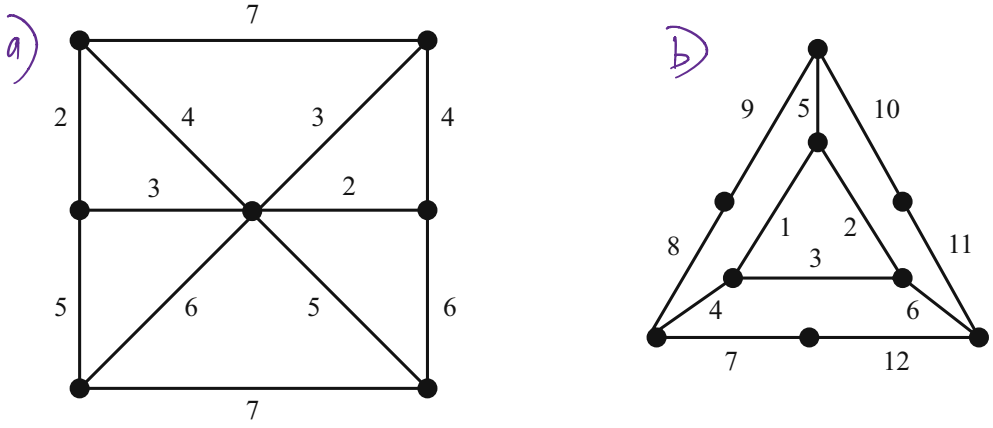


FIGURE 1.44. Two weighted graphs.

6. Prim's algorithm (from [228]) provides another method for finding minimum weight spanning trees.

Prim's Algorithm

Given: A connected, weighted graph G.

i. Choose a vertex v, and mark it.

ii. From among all edges that have one marked end vertex and one unmarked end vertex, choose an edge e of minimum weight. Mark the edge e, and also mark its unmarked end vertex.

iii. If every vertex of G is marked, then the set of marked edges forms a minimum weight spanning tree. If not, repeat step ii.

Use Prim's algorithm to find minimum weight spanning trees for the graphs in Figure 1.44. As you work, compare the stages to those of Kruskal's algorithm.

7. Give an example of a connected, weighted graph G having (i) a cycle with two identical weights, which is neither the smallest nor the largest weight in the graph, and (ii) a unique minimum weight spanning tree which contains exactly one of these two identical weights.

Textbook: "Combinatorics and Graph Theory" by Harris, Hirst, Mossinghoff

1.3.4 Counting Trees

As for everything else, so for a mathematical theory: beauty can be perceived but not explained.

— Arthur Cayley [214]

In this section we discuss two beautiful results on counting the number of spanning trees in a graph. The next chapter studies general techniques for counting arrangements of objects, so these results are a sneak preview.

Cayley's Tree Formula

Cayley's Tree Formula gives us a way to count the number of different labeled trees on n vertices. In this problem we think of the vertices as being fixed, and we consider all the ways to draw a tree on those fixed vertices. Figure 1.45 shows three different labeled trees on three vertices, and in fact, these are the only three.

There are 16 different labeled trees on four vertices, and they are shown in Figure 1.46.

As an exercise, the ambitious student should try drawing all of the labeled trees on five vertices. The cautious ambitious student might wish to look ahead at Cayley's formula before embarking on such a task.

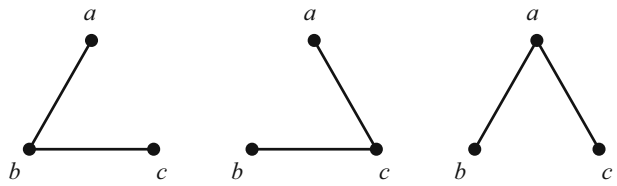


FIGURE 1.45. Labeled trees on three vertices.

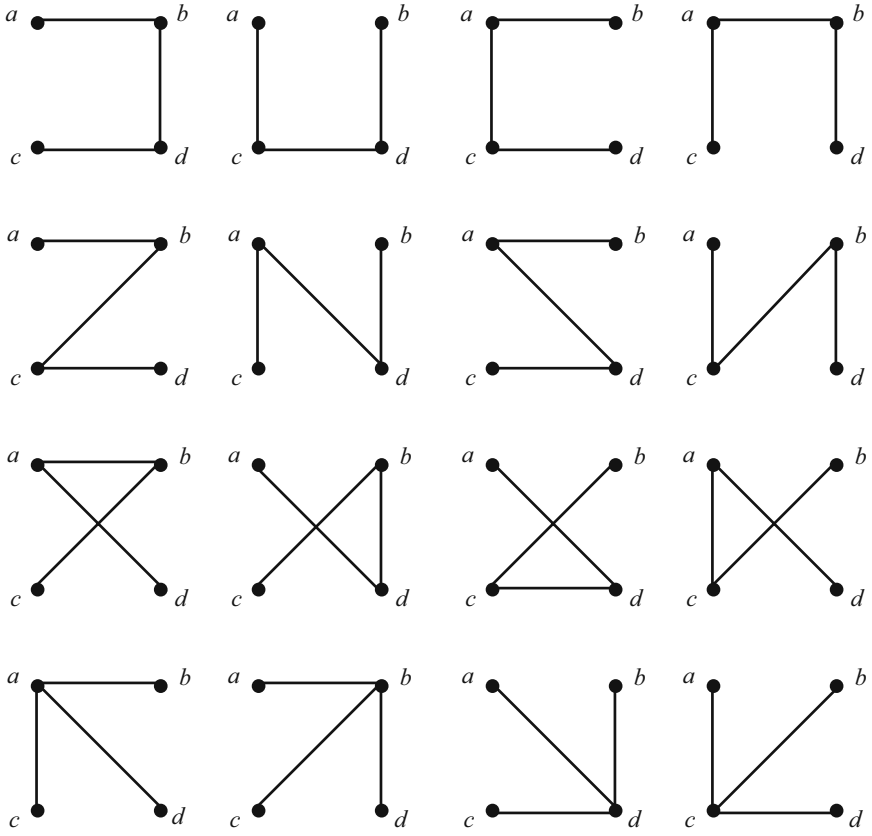


FIGURE 1.46. Labeled trees on four vertices.

Cayley proved the following theorem in 1889 [50]. The proof technique that we will describe here is due to Prüfer[7] [229]. Prüfer's method is almost as noteworthy as the result itself. He counted the labeled trees by placing them in one-to-one correspondence with a set whose size is easy to determine—the set of all sequences of length $n-2$ whose entries come from the set $\{1, \ldots, n\}$. There are n^{n-2} such sequences.

Theorem 1.18 (Cayley's Tree Formula)**.** *There are n^{n-2} distinct labeled trees of order n.*

The algorithm below gives the steps that Prüfer used to assign a particular sequence to a given tree, T, whose vertices are labeled $1, \ldots, n$. Each labeled tree is assigned a unique sequence.

[7] With a name like that he was destined for mathematical greatness!

Prüfer's Method for Assigning a Sequence to a Labeled Tree

Given: A tree T, with vertices labeled $1, \ldots, n$.

1. Let $i = 0$, and let $T_0 = T$.
2. Find the leaf on T_i with the smallest label and call it v.
3. Record in the sequence the label of v's neighbor.
4. Remove v from T_i to create a new tree T_{i+1}.
5. If $T_{i+1} = K_2$, then stop. Otherwise, increment i by 1 and go back to step 2.

See next page, p. 46, on this note

Let us run through this algorithm with a particular graph. In Figure 1.47, tree $T = T_0$ has 7 vertices, labeled as shown. The first step is finding the leaf with smallest label: This would be 2. The neighbor of vertex 2 is the vertex labeled 4. Therefore, 4 is the first entry in the sequence. Removing vertex 2 produces tree T_1. The leaf with smallest label in T_1 is 4, and its neighbor is 3. Therefore, we put 3 in the sequence and delete 4 from T_1. Vertex 5 is the smallest leaf in tree $T_2 = T_1 - \{4\}$, and its neighbor is 1. So our sequence so far is 4, 3, 1. In $T_3 = T_2 - \{5\}$ the smallest leaf is vertex 6, whose neighbor is 3. In $T_4 = T_3 - \{6\}$, the smallest leaf is vertex 3, whose neighbor is 1. Since $T_5 = K_2$, we stop here. Our resulting sequence is 4, 3, 1, 3, 1.

Notice that in the previous example, none of the leaves of the original tree T appears in the sequence. More generally, each vertex v appears in the sequence exactly $\deg(v) - 1$ times. This is not a coincidence (see Exercise 1). We now present Prüfer's algorithm for assigning trees to sequences. Each sequence gets assigned a unique tree.

Prüfer's Method for Assigning a Labeled Tree to a Sequence

Given: A sequence $\sigma = a_1, a_2, \ldots, a_k$ of entries from the set $\{1, \ldots, k+2\}$.

1. Draw $k+2$ vertices; label them $v_1, v_2, \ldots, v_{k+2}$. Let $S = \{1, 2, \ldots, k+2\}$.
2. Let $i = 0$, let $\sigma_0 = \sigma$, and let $S_0 = S$.
3. Let j be the smallest number in S_i that does not appear in the sequence σ_i.
4. Place an edge between vertex v_j and the vertex whose subscript appears first in the sequence σ_i.
5. Remove the first number in the sequence σ_i to create a new sequence σ_{i+1}. Remove the element j from the set S_i to create a new set S_{i+1}.
6. If the sequence σ_{i+1} is empty, place an edge between the two vertices whose subscripts are in S_{i+1}, and stop. Otherwise, increment i by 1 and return to step 3.

See p. 47 on this note

Prüfer's Method for Assigning a Sequence to a Labeled Tree

Given: A tree T, with vertices labeled $1, \ldots, n$.

1. Let $i = 0$, and let $T_0 = T$.
2. Find the leaf on T_i with the smallest label and call it v.
3. Record in the sequence the label of v's neighbor.
4. Remove v from T_i to create a new tree T_{i+1}.
5. If $T_{i+1} = K_2$, then stop. Otherwise, increment i by 1 and go back to step 2.

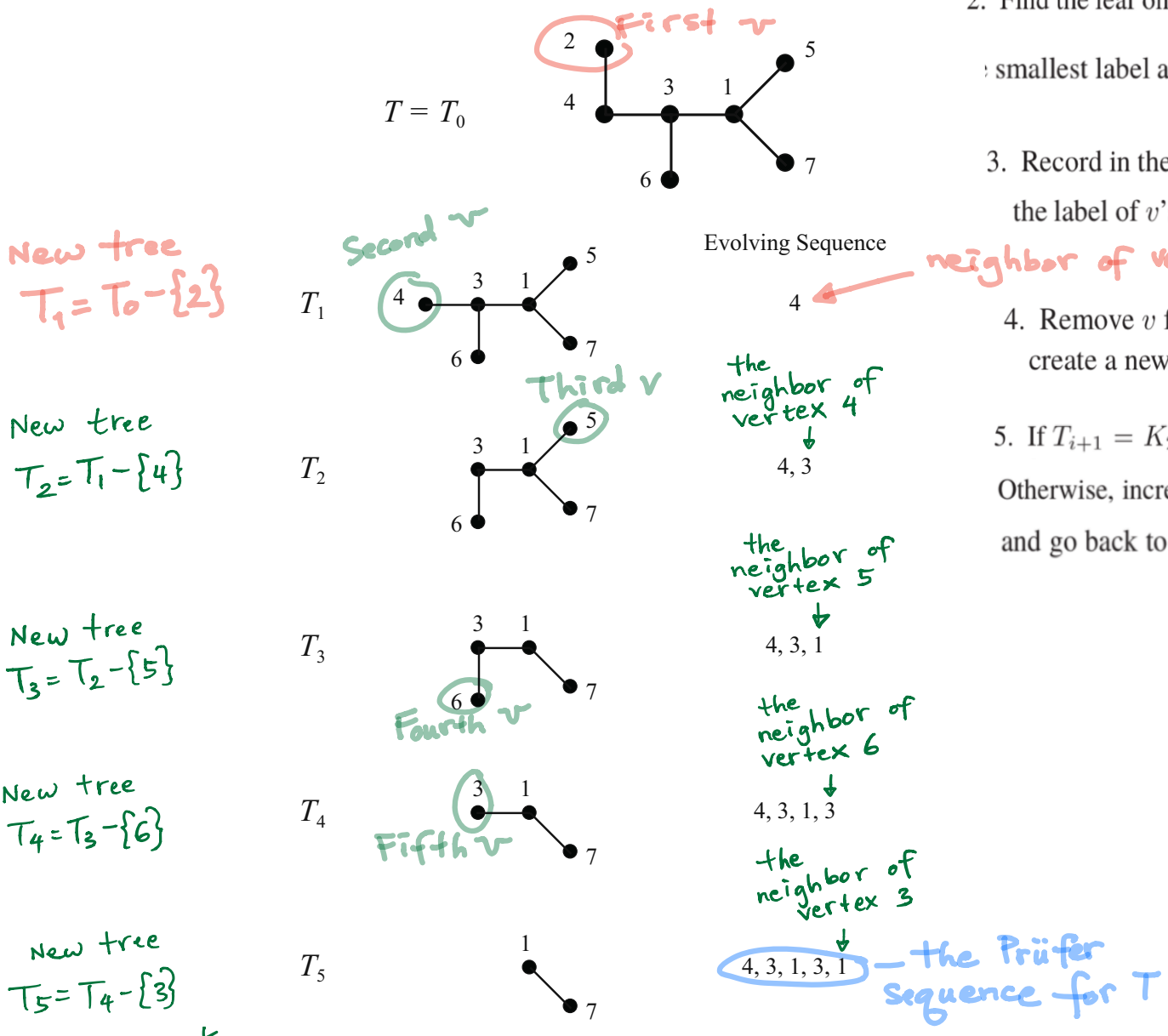
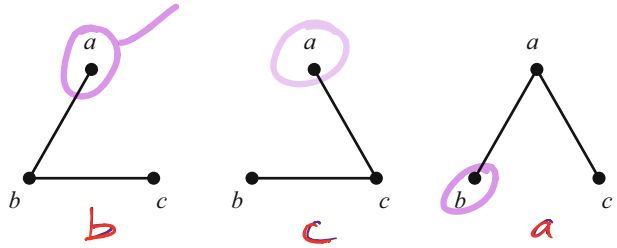


FIGURE 1.47. Creating a Prüfer sequence.

Let us apply this algorithm to a particular example. Let $\sigma = 4, 3, 1, 3, 1$ be our initial sequence to which we wish to assign a particular labeled tree. Since there are five terms in the sequence, our labels will come from the set $S = \{1, 2, 3, 4, 5, 6, 7\}$. After drawing the seven vertices, we look in the set $S = S_0$ to find the smallest subscript that does not appear in the sequence $\sigma = \sigma_0$. Subscript 2 is the one, and so we place an edge between vertices v_2 and v_4, the first subscript in the sequence. We now remove the first term from the sequence and the label v_2 from the set, forming a new sequence $\sigma_1 = 3, 1, 3, 1$ and a new set $S_1 = \{1, 3, 4, 5, 6, 7\}$. The remaining steps in the process are shown in Figure 1.48.

Examples of computing Prüfer sequence

Leaf w/ smallest label



Tree T

Prüfer sequence

FIGURE 1.45. Labeled trees on three vertices.

Circle the leaf w/ smallest label

After removing the circled leaf, box the leaf w/ smallest label

Prüfer sequence
first entry in red
Second entry in green

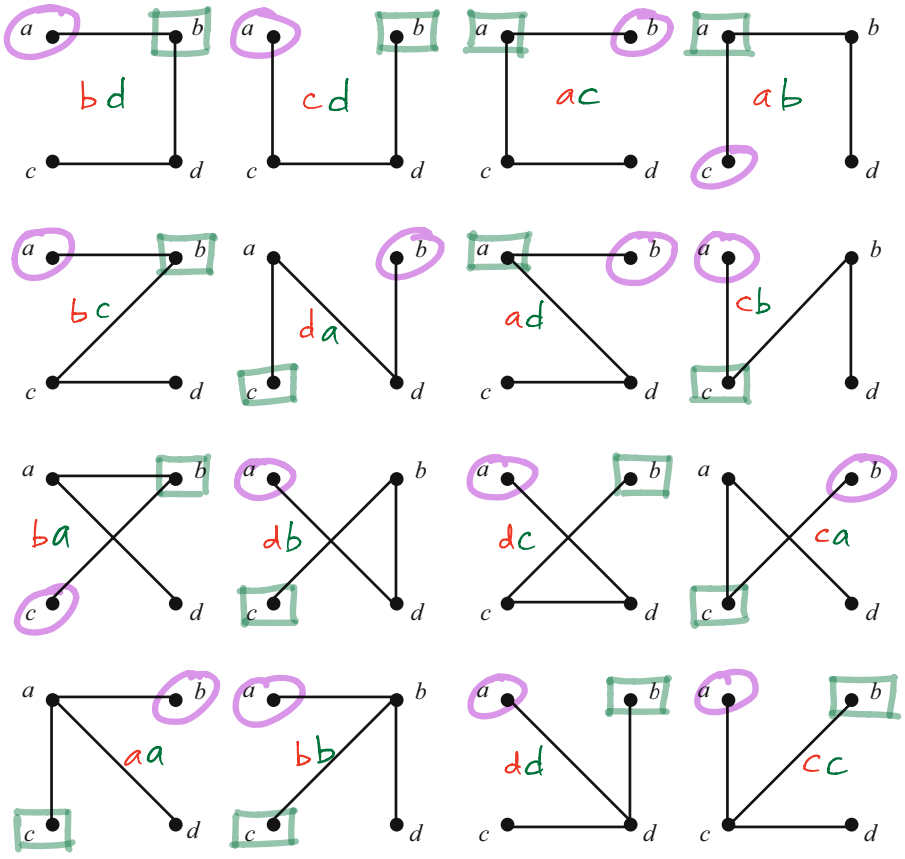


FIGURE 1.46. Labeled trees on four vertices.

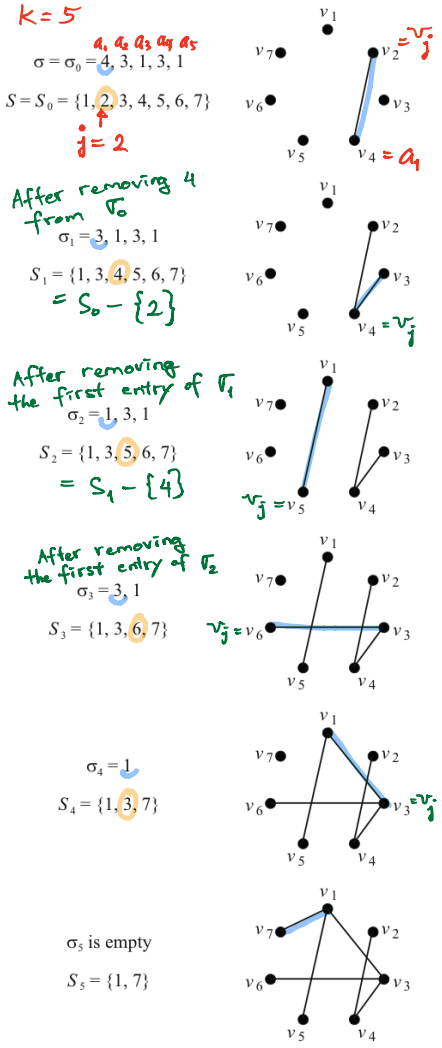


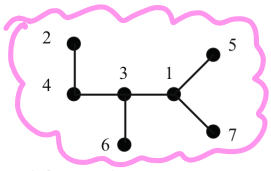
FIGURE 1.48. Building a labeled tree.

Prüfer's Method for Assigning a Labeled Tree to a Sequence

Given: A sequence $\sigma = a_1, a_2, \ldots, a_k$ of entries from the set $\{1, \ldots, k+2\}$.

1. Draw $k+2$ vertices; label them $v_1, v_2, \ldots, v_{k+2}$. Let $S = \{1, 2, \ldots, k+2\}$.
2. Let $i = 0$, let $\sigma_0 = \sigma$, and let $S_0 = S$.
3. Let j be the smallest number in S_i that does not appear in the sequence σ_i.
4. Place an edge between vertex v_j and the vertex whose subscript appears first in the sequence σ_i.
5. Remove the first number in the sequence σ_i to create a new sequence σ_{i+1}. Remove the element j from the set S_i to create a new set S_{i+1}.
6. If the sequence σ_{i+1} is empty, place an edge between the two vertices whose subscripts are in S_{i+1}, and stop. Otherwise, increment i by 1 and return to step 3.

6. If the sequence σ_{i+1} is empty, place an edge between the two vertices whose subscripts are in S_{i+1}, and stop.



You will notice that the tree that was created from the sequence σ in the second example is the very same tree that created the sequence σ in the first example. Score one for Prüfer!

Matrix Tree Theorem

The second major result that we present in this section is the Matrix Tree Theorem, and like Cayley's Theorem, it provides a way of counting spanning trees of labeled graphs. While Cayley's Theorem in essence gives us a count on the number of spanning trees of complete labeled graphs, the Matrix Tree Theorem applies to labeled graphs in general. The theorem was proved in 1847 by Kirchhoff [175], and it demonstrates a wonderful connection between spanning trees and matrices.

The theorem involves two special matrices. One is the adjacency matrix (defined back in Section 1.2.2), and the other is defined as follows. Let G be a graph with vertices $v_1, v_2, \ldots v_n$. The *degree matrix* of G is the $n \times n$ matrix D whose (i, j) entry, denoted by $[D]_{i,j}$, is defined by

$$[D]_{i,j} = \begin{cases} \deg(v_i) & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

So, the diagonal entries of D are the vertex degrees, and the off-diagonal entries are all zero.

Given an $n \times n$ matrix M, the i, j *cofactor* of M is defined to be

$$(-1)^{i+j} \det(M(i|j)),$$

where $\det(M(i|j))$ represents the determinant of the $(n-1) \times (n-1)$ matrix formed by deleting row i and column j from M.

We are now ready to state the Matrix Tree Theorem, due to Kirchhoff. The proof that we give imitates those presented in [148] and [52].

Theorem 1.19 (Matrix Tree Theorem)**.** *If G is a connected labeled graph with adjacency matrix A and degree matrix D, then the number of unique spanning trees of G is equal to the value of any cofactor of the matrix $D - A$.*

Proof. Suppose G has n vertices $(v_1, \ldots, v_n)$ and k edges $(f_1, \ldots, f_k)$. Since G is connected, we know that k is at least $n - 1$. Let N be the $n \times k$ matrix whose (i, j) entry is defined by

$$[N]_{i,j} = \begin{cases} 1 & \text{if } v_i \text{ and } f_j \text{ are incident,} \\ 0 & \text{otherwise.} \end{cases}$$

N is called the *incidence matrix* of G. Since every edge of G is incident with exactly two vertices of G, each column of N contains two 1's and $n - 2$ zeros. Let M be the $n \times k$ matrix that results from changing the topmost 1 in each column to -1. To prove the result, we first need to establish two facts, which we call Claim A and Claim B.

Claim A. $MM^T = D - A$ (where M^T denotes the transpose of M).

First, notice that the (i, j) entry of $D - A$ is

$$[D - A]_{i,j} = \begin{cases} \deg(v_i) & \text{if } i = j, \\ -1 & \text{if } i \neq j \text{ and } v_i v_j \in E(G), \\ 0 & \text{if } i \neq j \text{ and } v_i v_j \notin E(G). \end{cases}$$

Now, what about the (i, j) entry of MM^T? The rules of matrix multiplication tell us that this entry is the dot product of row i of M and column j of M^T. That is,

$$\begin{aligned} [MM^T]_{i,j} &= ([M]_{i,1}, [M]_{i,2}, \ldots, [M]_{i,k}) \cdot \left([M^T]_{1,j}, [M^T]_{2,j}, \ldots, [M^T]_{k,j}\right) \\ &= ([M]_{i,1}, [M]_{i,2}, \ldots, [M]_{i,k}) \cdot ([M]_{j,1}, [M]_{j,2}, \ldots, [M]_{j,k}) \\ &= \sum_{r=1}^{k} [M]_{i,r}[M]_{j,r}. \end{aligned}$$

 Skip

If $i = j$, then this sum counts one for every nonzero entry in row i; that is, it counts the degree of v_i. If $i \neq j$ and $v_i v_j \notin E(G)$, then there is no column of M in which both the row i and row j entries are nonzero. Hence the value of the sum in this case is 0. If $i \neq j$ and $v_i v_j \in E(G)$, then the only column in which both the row i and the row j entries are nonzero is the column that represents the edge $v_i v_j$. Since one of these entries is 1 and the other is -1, the value of the sum is -1. We have shown that the (i, j) entry of MM^T is the same as the (i, j) entry of $D - A$, and thus Claim A is proved.

Let H be a subgraph of G with n vertices and $n-1$ edges. Let p be an arbitrary integer between 1 and n, and let M' be the $(n-1) \times (n-1)$ submatrix of M formed by all rows of M except row p and the columns that correspond to the edges in H.

Claim B. If H is a tree, then $|\det(M')| = 1$. Otherwise, $\det(M') = 0$.

First suppose that H is not a tree. Since H has n vertices and $n-1$ edges, we know from earlier work that H must be disconnected. Let H_1 be a connected component of H that does not contain the vertex v_p. Let M'' be the $|V(H_1)| \times (n-1)$ submatrix of M' formed by eliminating all rows other than the ones corresponding to vertices of H_1. Each column of M'' contains exactly two nonzero entries: 1 and -1. Therefore, the sum of all of the row vectors of M'' is the zero vector, so the rows of M'' are linearly dependent. Since these rows are also rows of M', we see that $\det(M') = 0$.

Now suppose that H is a tree. Choose some leaf of H that is not v_p (Theorem 1.14 lets us know that we can do this), and call it u_1. Let us also say that e_1 is the edge of H that is incident with u_1. In the tree $H - u_1$, choose u_2 to be some leaf other than v_p. Let e_2 be the edge of $H - u_1$ incident with u_2. Keep removing leaves in this fashion until v_p is the only vertex left. Having established the list of vertices $u_1, u_2, \ldots, u_{n-1}$, we now create a new $(n-1) \times (n-1)$ matrix M^* by rearranging the rows of M' in the following way: row i of M^* will be the row of M' that corresponds to the vertex u_i.

An important (i.e., useful!) property of the matrix M^* is that it is lower triangular (we know this because for each i, vertex u_i is *not* incident with any of $e_{i+1}, e_{i+2}, \ldots, e_{n-1}$). Thus, the determinant of M^* is equal to the product of the main diagonal entries, which are either 1 or -1, since every u_i is incident with e_i. Thus, $|\det(M^*)| = 1$, and so $|\det(M')| = 1$. This proves Claim B.

We are now ready to investigate the cofactors of $D - A = MM^T$. It is a fact from matrix theory that if the row sums and column sums of a matrix are all 0, then the cofactors all have the same value. (It would be a nice exercise—and a nice review of matrix skills—for you to try to prove this.) Since the matrix MM^T satisfies this condition, we need to consider only one of its cofactors. We might as well choose i and j such that $i + j$ is even—let us choose $i = 1$ and $j = 1$. So, the $(1, 1)$ cofactor of $D - A$ is

$$\begin{aligned} \det\left((D - A)(1|1)\right) &= \det\left(MM^T(1|1)\right) \\ &= \det(M_1 M_1^T) \end{aligned}$$

where M_1 is the matrix obtained by deleting the first row of $D - A$.

At this point we make use of the Cauchy–Binet Formula, which says that the determinant above is equal to the sum of the determinants of $(n-1) \times (n-1)$ submatrices of M_1 (for a more thorough discussion of the Cauchy–Binet Formula, see [40]). We have already seen (in Claim B) that any $(n-1) \times (n-1)$ submatrix that corresponds to a spanning tree of G will contribute 1 to the sum, while all others contribute 0. This tells us that the value of $\det(D - A) = \det(MM^T)$ is precisely the number of spanning trees of G. □

Figure 1.49 shows a labeled graph G and each of its eight spanning trees.

Example:

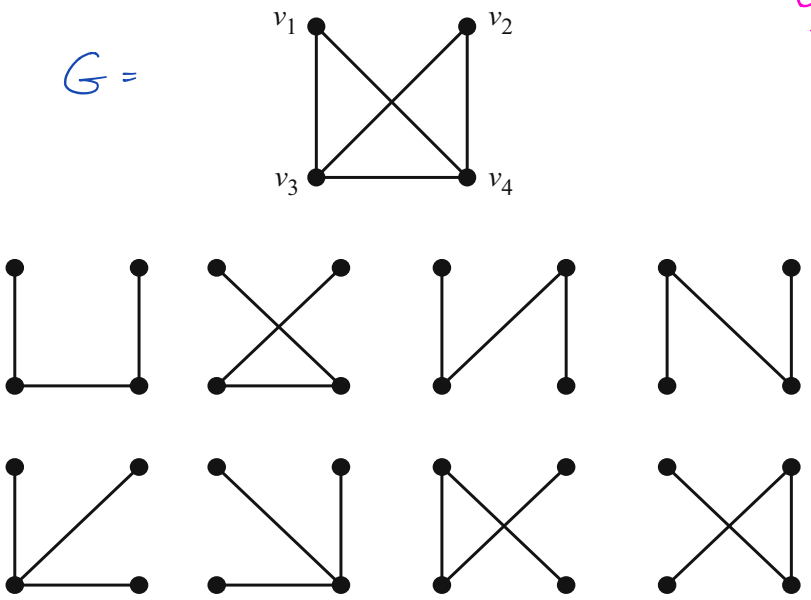


FIGURE 1.49. A labeled graph and its spanning trees.

The degree matrix D and adjacency matrix A of G are

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}, \quad A = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix},$$

and so

$$D - A = \begin{bmatrix} 2 & 0 & -1 & -1 \\ 0 & 2 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}.$$

The $(1,1)$ cofactor of $D - A$ is $(-1)^{1+1} \det((D-A) \text{ with row 1 \& column 1 removed})$

$$\det \begin{bmatrix} 2 & -1 & -1 \\ -1 & 3 & -1 \\ -1 & -1 & 3 \end{bmatrix} = 8.$$

Score one for Kirchhoff!

Theorem 1.19 (Matrix Tree Theorem)**.** *If G is a connected labeled graph with adjacency matrix A and degree matrix D, then the number of unique spanning trees of G is equal to the value of any cofactor of the matrix $D - A$.*

Exercises Sec 1.3,4 Counting Trees

1. Let T be a labeled tree. Prove that the Prüfer sequence of T will not contain any of the leaves' labels. Also prove that each vertex v will appear in the sequence exactly $\deg(v) - 1$ times.

2. Determine the Prüfer sequence for the trees in Figure 1.50.

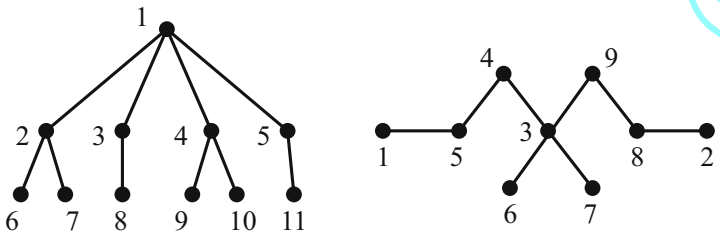


FIGURE 1.50. Two labeled trees.

3. Draw and label a tree whose Prüfer sequence is

$$5, 4, 3, 5, 4, 3, 5, 4, 3.$$

4. Which trees have constant Prüfer sequences?

5. Which trees have Prüfer sequences with distinct terms?

6. Let e be an edge of K_n. Use Cayley's Theorem to prove that $K_n - e$ has $(n-2)n^{n-3}$ spanning trees.

7. Use the Matrix Tree Theorem to prove Cayley's Theorem. Hint: Look back at the discussion prior to the statement of the Matrix Tree Theorem.

1.4 Trails, Circuits, Paths, and Cycles

Takes a real salesman, I can tell you that. Anvils have a limited appeal, you know.

— Charlie Cowell, anvil salesman, *The Music Man*

Charlie Cowell was a door to door anvil salesman, and he dragged his heavy wares down every single street in each town he visited. Not surprisingly, Charlie became quite proficient at designing routes that did not repeat many streets. He certainly did not want to drag the anvils any farther than necessary, and he especially liked it when he could cover every street in the town without repeating a single one.

After several years of unsuccessful sales (he saw more closed doors than closed deals), the Acme Anvil Company did the natural thing — they promoted him. Charlie moved from salesman to regional supplier. This meant that Charlie would