2. Trees

A *tree* is a connected graph with no cycles. A *forest* is a graph with no cycles.

Trees of one sort or another arise in a variety of situations. The directory structure of a computer is a tree, and trees arise in computer science in many other ways. Some of you will know about binary search trees. Statistics students will also be familiar with decision trees. Gardeners know about oak trees. There are family trees and more generally evolutionary trees.

When e is an edge of the graph G, recall that $G\backslash e$ ("G delete e") denotes the graph obtained by removing e and leaving all other edges and vertices intact.

An edge e of a connected graph G is a *bridge* if $G\backslash e$ is disconnected.

Remember how we said that graph theory terminology is not standardised? A bridge is sometimes known as a "cut edge" or (even more illustratively) an "isthmus". But in this course, we'll stick with the name "bridge".

Theorem 2.1. *Let G be a connected graph, and let e be an edge of G. The edge e is a bridge if and only if it is not contained in a cycle.*

Proof. Suppose e is a bridge of G. Then $G\backslash e$ is disconnected, so there exist vertices w and z such that there is no walk between w and z in $G\backslash e$. This shows that every walk from w to z in G contains the edge e. Now consider one such walk W (such a walk exists since G is connected). If e is contained in some cycle of G, then we can replace e in the walk W by the rest of the cycle, thereby obtaining a walk from w to z in $G\backslash e$. Since there are no walks from w to z in $G\backslash e$, this shows that e is not contained in any cycle of G.

Now, for the converse, suppose that e is not contained in any cycle of G. Let u and v be the ends of e, and note that $u \neq v$. If there is a walk between u and v in $G\backslash e$, then by Lemma 1.1 there is a path between u and v in $G\backslash e$, and, by adding e to this path, we see that G has a cycle containing e. But G has no such cycles, so there is no walk between u and v in $G\backslash e$. Thus u and v are in different components of $G\backslash e$, and in particular $G\backslash e$ is disconnected. Hence e is a bridge of G. □

With Theorem 2.1 in hand we can now give a *characterisation* of trees.

Corollary 2.2. *A graph is a tree if and only if it is connected and every edge is a bridge.*

Proof. Suppose G is a tree. Then G is connected and has no cycles. Since no edge of G is in a cycle, Theorem 2.1 implies that every edge of G is a bridge.

Conversely, suppose G is a connected graph where every edge is a bridge. Then, by Theorem 2.1, each edge is not in a cycle, so G has no cycles. Hence G is a tree. □

→ What's a *characterisation*? We use the word a lot in mathematics, and it's important. A characterisation is a theorem that tells us that a mathematical object that is defined by one property is really just the same as one that is defined by a different property. Thus, a tree is defined to be a connected graph with no cycles. But the characterisation of Corollary 2.2 tells us that we could just as well have defined a tree to be a connected graph all of whose edges are bridges.

A *leaf* of a graph is a vertex of degree one. Here is another useful fact about trees, but it is not a characterisation. Why?

Lemma 2.3. *Let T be a tree with at least two vertices. Then T has at least two leaves.*

Proof. Since T is a tree, it is connected, and since it has at least two vertices, it has at least one edge. Choose a path $P = v_1, e_1, v_2, e_2, \ldots, e_{n-1}, v_n$ in the tree T of maximum length. Since T has at least one edge, $n \geq 2$, so $v_1 \neq v_n$ and e_{n-1} is one edge that is incident to v_n. Assume that v_n has degree at least 2. Then there is an edge e incident with v_n that is distinct from the edge e_{n-1}. Let w be the other end of e, so $e = v_n w$. Now $w \in \{v_1, v_2, \ldots v_n\}$, as otherwise $v_1, e_1, v_2, e_2, \ldots, e_{n-1}, v_n, e, w$ is a path that is longer than P. Choose i so that $w = v_i$. Then $v_i, e_i, \ldots, e_{n-1}, v_n, e, v_i$ is a cycle in G, contradicting the fact that T is a tree. From this contradiction we deduce that v_n has degree one, that is, it is a leaf. Similarly, v_1 has degree one, so v_1 and v_n are distinct leaves of T. □

There is a bewildering diversity of trees with, say, 1000 vertices. Behind this diversity they all have one thing in common — they all have exactly 999 edges — and we can prove it!

By the way, how many non-isomorphic trees are there with 1000 vertices?

First we require a lemma. If you draw pictures, this lemma may seem obvious. But how to prove it? "Proof: obvious from pictures" is not satisfactory.

Lemma 2.4. *Let G be a connected graph. If e is a bridge of G, then $G\backslash e$ has exactly two components.*

Proof. Let $e = uv$ be a bridge of G. By definition, $G\backslash e$ has at least two components. Towards a contradiction, assume that $G\backslash e$ has at least three components. Then there is a component C of $G\backslash e$ such that neither u nor v is a vertex of C. Let c be a vertex of C. Since c and u are in different components of $G\backslash e$, there is no path from c to u in $G\backslash e$. But G is connected, so there is a path P from c to u in G. Hence P uses the edge e, so P is of the form $c, \ldots, v, e, u$. Now the subpath of P from c to v (i.e. the part $c, \ldots, v$) does not contain e, so it is a path in $G\backslash e$. But this contradicts that c and v are in different components of $G\backslash e$. We deduce that our initial assumption was false; that is, $G\backslash e$ has at most two components. So $G\backslash e$ has exactly two components, as required. □

Theorem 2.5. *Let G be a tree with n vertices, where $n \geq 1$. Then G has $n-1$ edges.*

Proof. We use strong induction on n, the number of vertices of G, to prove that G has $n - 1$ edges.

If $n = 1$, then G certainly has no edges. Now suppose $n > 1$ and that the result holds for all trees with fewer than n vertices (but at least one vertex). Let e be an edge of G and consider $G\backslash e$. By Lemma 2.4, $G\backslash e$ has two components, and each component is a tree with fewer than n vertices (but at least one vertex). Let T_1 and T_2 be these two components of $G\backslash e$, and say they have n_1 and n_2 vertices respectively. Then $n_1 + n_2 = n$. By the induction assumption, T_1 has $n_1 - 1$ edges and T_2 has $n_2 - 1$ edges. Now the edges of G are either edges of T_1, edges of T_2, or e. Hence G has

$$(n_1 - 1) + (n_2 - 1) + 1 = n_1 + n_2 - 1 = n - 1$$

edges. □

Theorem 2.6. *Let G be a forest with n vertices and c components, where $n \geq 1$. Then G has $n - c$ edges.*

Exercise 2.7. *Prove Theorem 2.6.*

Spanning trees. Let G be a graph on vertex set V. A *spanning tree* of G is a subgraph H on the vertex set V such that H is a tree.

(Note: it is important here that H has the same vertex set as G. A spanning tree retains all the vertices of G — we just potentially throw away some edges.)

Theorem 2.8. *Let G be a graph. The graph G is connected if and only if G has a spanning tree.*

Proof. ($\Leftarrow$) Suppose G has a spanning tree H. Then H is connected, so there is a path between every pair of vertices in H. But H is a subgraph of G with $V(H) = V(G)$, so there is a path between every pair of vertices in G. Thus G is connected.

($\Rightarrow$) Suppose G is a connected graph. If it is a tree, then G itself is a spanning tree of G, in which case G has a spanning tree as required. So assume that G is not a tree. Then G has an edge e that is *not* a bridge, by Corollary 2.2. Remove this edge and obtain a connected graph with fewer edges than G. This process can be repeated until we obtain a connected graph G' for which every edge of G' is a bridge. Since G' has the same vertex set as G, the subgraph G' is a spanning tree of G, as required. □

Using the last two theorems, we can obtain another characterisation of trees.

Corollary 2.9. *Let G be a graph with n vertices, where $n \geq 1$. The graph G is a tree if and only if G is connected and has $n - 1$ edges.*

Proof. If G is a tree, then it is connected and, by Theorem 2.5, has $n - 1$ edges.

For the converse, suppose G is not a tree, but G is connected. Then, by Theorem 2.8, G has a spanning tree G', where G' has n vertices (since G and G' have the same vertex set). By Theorem 2.5, G' has $n - 1$ edges. Now G has at least $n - 1$ edges; in fact, since it is not a tree, it has more edges than G', so it has at least n edges. Thus we have shown that if G is not a tree, then either G is not connected or G has more than $n - 1$ edges. □

→ Note the proof technique of Corollary 2.9. We want to prove A holds if and only if B holds. The standard way is to show that A implies B and also to show that B implies A. In Corollary 2.9 we used the technique of showing A implies B, and not A implies not B (the latter is the *contrapositive* of B implies A). It is an easy exercise in logic to show that these are logically equivalent.

In practice, finding spanning trees of graphs is very important. Essentially, this is because a spanning tree is a minimal connected subgraph containing all the vertices. Typically one seeks a spanning tree of a certain type. Computer science students may be familiar with *depth-first search*, which constructs a spanning tree that is as path-like as possible, and *breadth-first search*, which constructs a spanning tree that is as "star-like" as possible.

A graph may have many different spanning trees. How many spanning trees does K_n have?

Lemma 2.10. *Let G be a connected graph, and let e be an edge of G that is not a loop. Then there is a spanning tree of G that contains e.*

Proof. We prove this by induction on the number of edges of G. Let m be the number of edges of G. Clearly the lemma holds if $m = 1$. Now assume the lemma holds when $m = k$, for some positive integer k, and consider when $m = k + 1$. If G is a tree, then G itself is a spanning tree containing e. So assume that G is not a tree. Then G has a cycle C. As e is not a loop, there is an edge f of C such that $f \neq e$. Since f is in a cycle, it is not a bridge (by Theorem 2.1), so $G\backslash f$ is connected, and has k edges. Thus, by the induction assumption, $G\backslash f$ has a spanning tree containing e, and this is also a spanning tree for G. □

→ The proof above uses induction, but there is nothing scary about it: the induction is used to formalise the idea that we could repeatedly find an edge $f \neq e$ and delete it until we have only a spanning tree left.

The following is a characterisation of a spanning tree.

Proposition 2.11. *Let G be a graph, and let H be a subgraph of G. Then H is a spanning tree of G if and only if H is a forest with $|V(G)| - 1$ edges.*

Proof. Let H be a spanning tree of G. Then clearly H is a forest, and, by Theorem 2.5, H has $|V(H)| - 1 = |V(G)| - 1$ edges.

Now suppose H is a forest with $|V(G)| - 1$ edges. Let c be the number of components of H, so $c \geq 1$. Then $|V(G)| - 1 = |E(H)| = |V(H)| - c$, by Theorem 2.6. But $V(H) \subseteq V(G)$, so $|V(G)| \geq |V(H)|$, which implies $c \leq 1$. Thus H has one component, so H is a tree and $|V(H)| = |V(G)|$; that is, H is a spanning tree of G. □

Rooted trees. In many applications, trees come with a special vertex, usually called the *root*. The root typically represents some sort of starting place. In a family tree, the root would be the original ancestor. In a unix computer system, to get to the root of the directory structure you have to log in as the *root* user. Here we're really only mentioning rooted trees here to prevent confusion. In other contexts, trees are sometimes presented as if they are always rooted and students who have been exposed to this sometimes develop the expectation that

all trees must be rooted. This is not true. None of the trees considered in this section have been rooted.

Contraction revisited. Now that we have recalled the definitions of a forest and a spanning tree, we consider the effect of contracting an edge on these structures.

Lemma 2.12. *Let $G = (V, E)$ be a graph with a non-loop edge e, and let $F \subseteq E \setminus \{e\}$. Then F is the edge set of a forest of G/e if and only if $F \cup \{e\}$ is the edge set of a forest of G.*

Proof. Say F is not the set of edges of a forest of G/e. Then there is a cycle C of G/e whose edges are contained in F. By Lemma 1.12, either C or $C \cup \{e\}$ is a cycle of G. In either case, $F \cup \{e\}$ contains a cycle of G, so $F \cup \{e\}$ is not the set of edges of a forest of G.

Conversely, say $F \cup \{e\}$ is not the set of edges of a forest of G. Then there is a cycle of G whose edges are contained in $F \cup \{e\}$. We consider two cases. First, suppose there is such a cycle C containing e. Then, by Lemma 1.12, $C \setminus \{e\}$ is the set of edges of a cycle in G/e. Now $C \setminus \{e\} \subseteq F$, so F is not the edge set of a forest of G/e. Second, suppose that there is no cycle containing e and contained in $F \cup \{e\}$. Then e is a bridge of $G[F \cup \{e\}]$, by Theorem 2.1. Thus, any cycle contained in $F \cup \{e\}$ must belong entirely to one of the components we obtain when we delete e. Such a cycle contains at most one of u and v. Hence, again by Lemma 1.12, there is a cycle of G/e with edges contained in F, so F is not the edge set of a forest of G/e. □

→ Note that here we proved "A if and only if B" by proving "not A implies not B" and "not B implies not A".

Recall that we can unambiguously describe a tree, or a cycle, by the set of edges it contains. For a forest, however, it could contain vertices with degree zero, and there is no way of knowing this by looking only at the edge set.

Lemma 2.13. *Let G be a connected graph, let e be a non-loop edge of G, and let $T \subseteq E(G) \setminus \{e\}$. Then T is the edge set of a spanning tree of G/e if and only if $T \cup \{e\}$ is the edge set of a spanning tree of G.*

Proof. Say $T \cup \{e\}$ is the edge set of a spanning tree of G. Then, by Lemma 2.12, T is the edge set of a forest in G/e. Also, by Theorem 2.5, $|T \cup \{e\}| = |V(G)| - 1$,

so $|T| = |V(G)| - 2 = |V(G/e)| - 1$. Hence, by Theorem 2.6, T is the edge set of a spanning tree of G/e.

Say T is the edge set of a spanning tree in G/e. By Lemma 2.12, $T \cup \{e\}$ is the edge set of a forest in G. But, using Theorem 2.5 again, $|T| = |V(G/e)| - 1 = |V(G)| - 2$, so $|T \cup \{e\}| = |V(G)| - 1$, and hence $T \cup \{e\}$ is the edge set of a spanning tree of G, by Theorem 2.6. □

Cut vertices. The remainder of this section is not actually related to trees, but concerns a vertex analogue of bridges. We also will finally prove Theorem 1.7, first stated in Section 1.

Earlier we defined a bridge to be an edge whose deletion disconnects the graph. But what if a graph is disconnected? We say that an edge e in a (connected or disconnected) graph G is a *bridge* if $G \backslash e$ has more components than G. Equivalently, an edge e is a bridge if $C \backslash e$ is disconnected where C is the component of G containing e. (Are these really equivalent? The next exercise asks you to prove that they are.)

Exercise 2.14. *Let G be a graph with an edge e, and let C be the component of G containing e. Prove that e is a bridge of C if and only if e is a bridge of G.*

Recall that, if v is a vertex of G, then $G - v$ is the graph obtained from G by removing v and all edges incident with v.

→ Note that we use '$\backslash$' when deleting an edge, and '$-$' when deleting a vertex. This is a deliberate choice of notation, since deleting a vertex does something different to deleting an edge! We also use '$\backslash$' in one other situation however: when X and Y are sets, we use $X \setminus Y$ to denote "set difference", i.e. the set of elements in X that are not also in Y. In the upcoming proof of Lemma 2.16, we use '$-$', and '$\backslash$' in these two different ways... can you spot all three of these?

We have seen that a bridge in a connected graph shows that the graph is only connected in a fragile way. Certain vertices can have a similar property. In a connected graph G, a vertex v is a *cut vertex* if $G - v$ is not connected.

More generally: a vertex v in a graph G is a *cut vertex* if $G - v$ has more components than G. Equivalently, v is a cut vertex if $C - v$ is disconnected where C is the component containing C.

Exercise 2.15. *Let G be a graph with a vertex v, and let C be the component of G containing v. Prove that v is a cut vertex of C if and only if v is a cut vertex of G.*

From pictures it seems that if $e = uv$ is a bridge, then u and v are likely to be cut vertices, but this is not always the case.

Recall that a *leaf* of a graph is a vertex of degree 1. A *pendant edge* is an edge incident with a leaf.

Lemma 2.16. *Let G be a graph, and let v be a leaf of G. Both of the following hold:*

(i) *The vertex v is not a cut vertex.*
(ii) *The pendant edge incident with v is a bridge.*

Proof. Let e be the pendant edge incident with v. To begin with, we assume that G is connected.

Consider (i). As v is a leaf of G, any path of G that contains v must begin or end at v. Let s and t be vertices in $V(G) \setminus \{v\}$. As G is connected, there is a path P from s to t. By the previous observation, this path does not contain v. Therefore P is also a path of $G - v$. Since, there is a path between any pair of vertices in the graph $G - v$, this shows that $G - v$ is connected, so v is not a cut vertex.

Consider (ii). Let $e = uv$ (so u is the other vertex that e is incident to). The vertex v has degree 0 in $G\backslash e$, so there is no path from u to v in $G\backslash e$. As u and v are in different components of $G\backslash e$, this tells us that $G\backslash e$ is disconnected. So e is a bridge of G.

We have shown that (i) and (ii) hold when G is connected. When G is not connected, we can use the same argument on the component C of G containing v. Since v is a cut vertex in C if and only if it is a cut vertex in G (see Exercise 2.15), and e is a bridge of C if and only if it is a bridge of G (see Exercise 2.14), this completes the proof. □

You might be tempted to conjecture that if $e = uv$ is a bridge, then both u and v are cut vertices unless e is a pendant edge. But this is not quite correct.

Exercise 2.17. *Describe exactly when a vertex incident with a bridge is not a cut vertex.*

Lemma 2.18. *Let G be a connected graph with at least three vertices, and let $e = uv$ be a bridge of G. Then at least one of u and v is a cut vertex.*

Proof. It follows from the solution to Exercise 2.17 that, if neither u nor v is a cut vertex, then the two components of $G \backslash e$ each contain only a single vertex. But then G has only two vertices. Thus, if G has at least three vertices, then at least one of u and v must be a cut vertex. □

Above we used the notation $e = uv$. But note that if there is more than one edge joining vertices u and v, the notation is ambiguous, and the edge e is not well-defined. This is not a problem above, as e is a bridge, but it is something to be aware of. Long story short: the notation is convenient, but use it with care.

The next lemma illustrates an important property of cut vertices.

Lemma 2.19. *Let H be a connected graph, and let v be a vertex of H. Then v is a cut vertex if and only if there is a partition $\{A, B\}$ of $V(H) \setminus \{v\}$ such that each path in H from a vertex in A to a vertex in B passes through v.*

Proof. ($\Rightarrow$) Suppose that v is a cut vertex. When we delete the vertex v from H, the resulting graph $H - v$ is not connected. Choose one component C of the graph $H - v$, and let A be the set of vertices in C, and let B be the set of vertices not in C. Then $\{A, B\}$ is a partition of $V(H) \setminus \{v\}$. Moreover, for each vertex $a \in A$ and for each vertex $b \in B$, there is a path from a to b in H (since H is connected), but this path contains v (since the path does not exist in $H - v$).

($\Leftarrow$) Now suppose that there is a partition $\{A, B\}$ of $V(H) \setminus \{v\}$ such that each path in H from a vertex in A to a vertex in B passes through v. Arbitrarily pick some $a \in A$ and some $b \in B$. Since any path from a to b in H passes through v, there is no no path from a to b in $H - v$. Thus a and b are in different components of H, so v is a cut vertex. □

Recall that a *partition* of a set X is a set of non-empty subsets of X, called *parts*, such that each $x \in X$ is in precisely one of the parts. Note that each part of the partition must not be empty. For example, in Lemma 2.19, the sets A and B have size at least one.

We next prove Theorem 1.7, first stated in Section 1. We restate it here for convenience.

Theorem 1.7. A graph G is bipartite if and only if G has no cycles of odd length.

It's helpful to describe the sets of a partition using the language of colours; in other words, we will talk about the *red* set of vertices and the *blue* set of vertices. We say a 2-*colouring* of G is a partition of $V(G)$ into two parts, the red vertices and the blue vertices, such that each edge of G has one end red and one end blue. We say G is 2-*colourable* if it has a 2-colouring. If we can show that G is 2-colourable, then we have shown that G is bipartite.

Proof of Theorem 1.7. Say that G has a cycle C of odd length. Let the vertices of C be $v_1, v_2, \ldots, v_{2t+1}$. We now attempt to find a 2-colouring of C. If we colour v_1 red, then v_2 must be blue, v_3 must be red, v_4 must be blue and so on. Indeed, all the odd numbered vertices must be red and the even numbered vertices must be blue. Hence v_{2t+1} is red. That is, both v_1 and v_{2t+1} are red. But there is an edge joining v_1 and v_{2t+1}. We have shown that C is not 2-colourable, and hence G is not 2-colourable.

Now consider the converse. Assume that G has no cycles of odd length. The proof is by induction on the number of edges of G. We may assume that G is connected, since a graph is bipartite if and only if each of its components is bipartite.

If G has one edge (which cannot be a loop, as this would be a cycle of odd length), then the result clearly holds. Assume that G has n edges and that the result holds for connected graphs with no odd cycles and $n-1$ edges.

There are two cases. For the first case, assume that G is a tree. By Lemma 2.3, G has a leaf v. By Lemma 2.16(i), v is not a cut vertex. Hence the graph $G-v$ (obtained by deleting v and the single pendant edge $e = vw$ incident with v) is connected. By the induction hypothesis, $G-v$ is bipartite, so it is 2-colourable. It is now easy to extend a 2-colouring of $G-v$ to a 2-colouring of G: if w is a red vertex then we colour v blue, whereas if w is blue then we colour v red. Thus G is bipartite.

For the second case, assume that G is not a tree. Then G has an edge $e = uv$ in a cycle C. Consider the graph $G\backslash e$. By Theorem 2.1, $G\backslash e$ is connected. As G has no odd cycles neither does $G\backslash e$. By the induction assumption $G\backslash e$ is bipartite, so it has a 2-colouring. We need to show this 2-colouring extends to G. All will be good if we can demonstrate that u and v must have different colours.

As $G\backslash e$ is 2-connected, there is a path P from u to v in $G\backslash e$. As e joins the first and last vertex of this path, we obtain a cycle when we add e. This cycle must have even length. Therefore there are an even number of vertices in P. Arguing as before we see that the first and last vertex of P, that is u and v, must have different colours — as otherwise we would not have a 2-colouring of $G\backslash e$. Now the edge e joins vertices with different colours and we have shown that G is bipartite. □