Queen's University
School of Computing

CISC 203: Discrete Mathematics for Computing II
Lecture 6: Trees
Winter 2019

1 An Introduction to Dendrology

Dendrology is the scientific study of trees. Although dendrology may seem strangely out of place in the realm of computing, the two fields share a common interest. Both dendrologists and computer scientists care a lot about trees; the particular type of tree each group cares about, however, may differ.

In these notes, we will take our knowledge of graph theory and use it to investigate a particular class of graphs called **trees**. Unlike what we find outdoors, the trees we will investigate do not come in pine, birch, or spruce variants. However, that does not necessarily mean the trees we will investigate are not as diverse and unique as those found in nature.

2 Trees

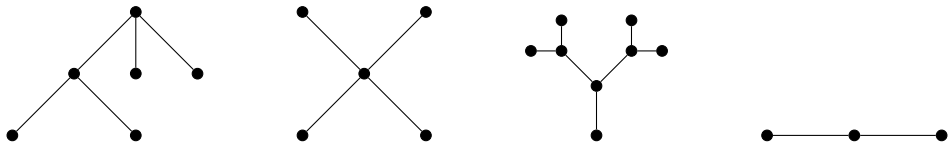
Before we proceed, recall that a graph is connected if there exists some path between every pair of vertices within the graph. Further recall that a graph contains a circuit (also called a cycle) if there exists a path within the graph that begins and ends at the same vertex. If no such path exists, then we say that the graph is **acyclic**.

Combining the two notions of connectedness and acyclicity is all that is needed in order to define the main object of these notes.

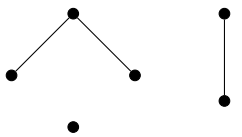
Definition 1 (Tree)**.** A tree is a connected acyclic graph.

In keeping with the arboreal nature of our terminology, a graph consisting of more than one connected component, where each connected component is a tree, is called a **forest**.

Example 2. Each of the following graphs are trees.



The following graph, containing three connected components, is a forest.

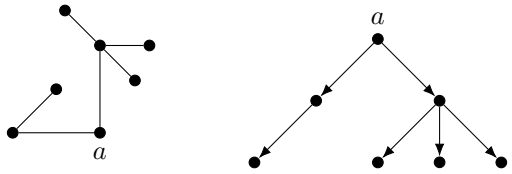


Our terse definition of a tree doesn't leave us much to work with if we wish to make our study of trees consistent in some way. For instance, the definition alone gives us no way to standardize the method in which we draw a tree. If we want to do so, we must create a more precise definition by placing an additional constraint on the tree that distinguishes a particular vertex. We can then use this distinguished vertex as a starting point from which we can draw the rest of the tree.

Definition 3 (Rooted tree)**.** A rooted tree is a tree with one vertex designated as the root. Every other vertex in the tree is directed away from the root vertex.

Going forward, when we refer to a "tree", we will almost always mean "rooted tree". We simply omit the word "rooted" to save space. We will often omit the directions on each edge of a rooted tree as well, since both the root and the edge directions can be inferred from the way a tree is drawn.

Example 4. The graph at left is a tree. The graph at right is the equivalent rooted tree produced after choosing vertex a to be the root.



Finally, since we are on the topic of modifying definitions, it is possible to define a tree recursively in terms of smaller instances of trees. In this context, "smaller" is a measure on the number of vertices of the tree. In some cases, a recursive definition can be useful to prove certain results (especially results relating to the structure of a tree), so we state it here.

Definition 5 (Tree—recursive)**.** A tree is either

- a null graph; or
- a graph consisting of one vertex adjacent to zero or more smaller trees.

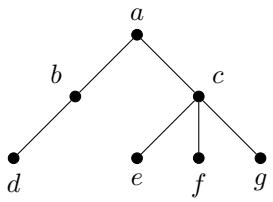
2.1 Terminology

Since a tree is nothing more than a certain type of graph, all of the usual graph terminology applies to trees as well, such as "vertex" and "edge". However, in addition to these terms, there exists a wealth of other terms relating specifically to trees.

We begin by summarizing some important terms relating to vertices and edges.

- The **root** of a tree is the unique vertex in a rooted tree with indegree zero.
- A **leaf** of a tree (also called an **external vertex**) is a vertex in a rooted tree with outdegree zero.
- An **internal vertex** is a vertex in a rooted tree with outdegree greater than zero.
- Given a vertex v, the **parent** of v is the unique vertex u incident to the directed edge (u, v).
- Given a vertex v, a **child** of v is any vertex w that is adjacent to v by some directed edge (v, w).
- Vertices that share a common parent vertex are called **siblings**.
- The **ancestors** of a vertex v are all vertices on the path from the root to v.
- The **descendants** of a vertex v are all vertices that have v as an ancestor.
- The **degree** of a vertex v is equal to the number of children of v.

Example 6. Consider the tree T at left.

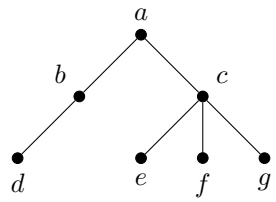


- The root of T is a.
- The leaves of T are d, e, f, and g.
- The internal vertices of T are a, b, and c.
- The parent of c is a. The children of c are e, f, and g. Each of these vertices are siblings.
- The ancestors of d are a and b.
- The only descendant of b is d.

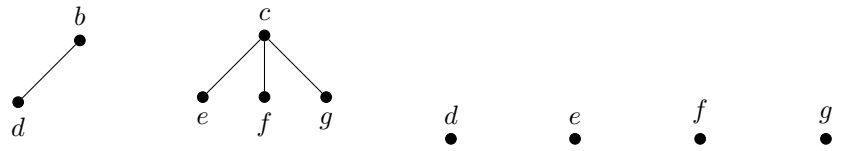
We can also talk about the structure of the overall tree using specific terminology.

- Given a vertex v in a tree T, the **subtree** of T rooted at v is the subgraph containing v and its descendants.
- A tree T is an ***m*-ary tree** if each internal vertex of T has at most m children.
- An m-ary tree T is **full** if every internal vertex of T has exactly m children.

Example 7. Consider again the tree T from Example 6.



This tree is a 3-ary tree (also known as a **ternary tree**), since each internal vertex has at most three children. The tree is not full, however, because vertices a and b have fewer than three children. The following subtrees are contained within T:

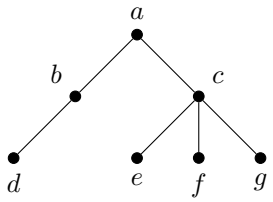


The most common type of m-ary tree we will see is a **binary tree**, where each vertex of the tree has at most two children. Binary trees are prolific throughout computer science, so our notes will return to these trees from time to time.

Finally, there exist a few terms relating to the position of vertices within a tree.

- The **level** of a vertex v (also called the **depth** of v) is the length of the path from the root to v.
- The **height of a vertex** v is the length of the longest path from v to a leaf.
- The **height of a tree** T is the length of the longest path from the root of T to a leaf.

Example 8. Consider once more the tree T from Examples 6 and 7.



- Vertex a is on level 0 of T.
- Vertices b and c are on level 1 of T.
- Vertices d, e, f, and g are on level 2 of T.
- The height of vertex a is 2.
- The height of vertices b and c is 1.
- The height of T is 2.

2.2 Properties of Trees

Immediately from the definition of a tree, we can discover a number of properties about trees. For instance, all trees are bipartite graphs; we can illustrate this by colouring all vertices on odd levels of the tree with one colour and colouring all vertices on even levels of the tree with another colour. Moreover, all trees are planar graphs (that is, as long as the tree contains a countably infinite number of vertices, but we won't worry about infinities here).

Here, we prove some arguably more interesting properties of trees.

Given the difficulties we previously established about the vagueness of the definition of a tree, we might sometimes want to have alternative methods of identifying when a graph is a tree. Fortunately, it is rather straightforward to come up with a number of equivalent characterizations of a tree.

Theorem 9. *Given a graph T, the following statements are equivalent.*

1. *T is a tree.*
2. *There exists exactly one path between any pair of vertices in T.*
3. *T is connected and every edge of T is a cut edge.*
4. *T is acyclic and the addition of any edge to T creates a circuit.*

Proof. To prove the equivalence of all of these statements, we will show that one statement implies the next statement in a chain of proofs.

$1 \Rightarrow 2$: Let T be a tree, and suppose that there exists some pair of vertices u and v that are connected by two distinct paths, say p_1 and p_2. We can reach v from u by following path p_1, and we can return to u from v by following path p_2. However, this implies that T contains a circuit, which contradicts our assumption that T is a tree. Therefore, no two vertices of T can be connected by more than one path.

$2 \Rightarrow 3$: Let T be a graph where every pair of vertices is connected by exactly one path. Then T is connected. Let e be an arbitrary edge of T, and let $T - e$ be the tree produced by removing e from T. If $T - e$ is connected, then there exists a path between the vertices that were previously incident to e. Since e itself was also a path between these vertices, then there must have existed more than one path between some pair of vertices in T, which contradicts our assumption. Therefore, $T - e$ is not connected and, since e is an arbitrary edge, any edge removed from T is a cut edge.

$3 \Rightarrow 4$: Let T be a connected graph where every edge is a cut edge. Then every pair of vertices in T is connected by exactly one path, so T is acyclic. Suppose we add an edge e to T between two arbitrary vertices u and v. Since T is connected, a path already exists between u and v, so the addition of e to T creates a circuit between u and v. Since u and v are arbitrary vertices, the addition of any edge to T creates a circuit.

$4 \Rightarrow 1$: Let T be an acyclic graph where the addition of any edge to T creates a circuit. To show that T is a tree, all we need to do is show that T is connected. Suppose that T is not connected. Then there exists some pair of vertices u and v such that no path exists between these vertices. However, this means that we can add an edge between u and v without creating a circuit, which contradicts our assumption. Therefore, T must be connected, and thus T is a tree. □

There exist other equivalent characterizations of a tree beyond the four characterizations we proved here. For example, it is possible to show that a graph T is a tree if and only if T is connected *or* acyclic and T contains n vertices and $n - 1$ edges. Here, we will prove a simpler version of that statement on its own. First, however, we require a small intermediate result.

Lemma 10. *If T is a tree and v is a leaf of T, then $T - v$ is a tree.*

Proof. To show that $T - v$ is a tree, we must show that $T - v$ is both connected and acyclic.

We can see immediately that $T - v$ is acyclic, since if it were not, then our original tree T would contain a circuit. Since T is acyclic, $T - v$ must also be acyclic.

Next, consider two arbitrary vertices u and w from $T - v$. If $T - v$ is connected, then there exists a path between u and w. Moreover, this path cannot contain the removed vertex v; if it did, then v would have to be located somewhere between u and w on the path, meaning that v would have degree 2. This contradicts our assumption that v is a leaf. Therefore, for any pair of arbitrary vertices in $T - v$, there exists a path between those vertices that does not include v, and so $T - v$ is connected. □

With this lemma, we can prove the aforementioned statement.

Theorem 11. *A tree T with n vertices contains $n-1$ edges.*

Proof. We will prove the claim using the principle of mathematical induction on the number of vertices of T. Let $P(n)$ be the statement "a tree T with n vertices contains $n-1$ edges".

When $n = 1$, the tree contains one vertex and thus has no edges. Therefore, $P(1)$ is true.

Assume that $P(k)$ is true for some $k \in \mathbb{N}$.

We now show that $P(k+1)$ is true; that is, a tree with $k+1$ vertices contains k edges. Suppose v is a leaf of T. By Lemma 10, removing v from T produces a tree $T - v$ with k vertices. By our inductive hypothesis, $T - v$ contains $k-1$ edges. It follows that T must contain k edges when we add v back into the tree.

Therefore, $P(k+1)$ is true. By the principle of mathematical induction, $P(n)$ is true for all $n \in \mathbb{N}$. □

Let us now shift from general trees to binary trees for a moment. Since binary trees are so popular in computer science, many results about these types of trees are known. For instance, given the height of a binary tree, we can determine certain measures such as the maximum number of leaves in that tree.

Theorem 12. *A binary tree T of height h contains at most 2^h leaves.*

Proof. We will prove the claim using the principle of mathematical induction on the height of T. Let $P(h)$ be the statement "a binary tree T of height h contains at most 2^h leaves".

When $h = 1$, T consists of a root with at most 2 children, so T contains at most $2^1 = 2$ leaves. Therefore, $P(1)$ is true.

Assume that $P(h')$ is true for some $h' \in \mathbb{N}$.

We now show that $P(h'+1)$ is true; that is, a binary tree of height $h'+1$ contains at most $2^{h'+1}$ leaves. Let T be a binary tree of height $h'+1$. Then T consists of a root with at most 2 children, both of which are the roots of subtrees of height h'. By our inductive hypothesis, both of these subtrees contain at most $2^{h'}$ leaves. Thus, T must contain at most $2 \times 2^{h'} = 2^{h'+1}$ leaves.

Therefore, $P(h'+1)$ is true. By the principle of mathematical induction, $P(h)$ is true for all $h \in \mathbb{N}$. □

Using a different proof technique, we can count the maximum number of vertices in a binary tree.

Theorem 13. *A binary tree T of height h contains at most $2^{h+1} - 1$ vertices.*

Proof. To prove this theorem, we count the maximum number of vertices at each level of the tree. Since we are dealing with a binary tree, the number of vertices at any given level may be at most twice the number of vertices at the previous level.

Suppose we have a binary tree T of height h. Then the zeroth level of T contains at most one vertex: the root. The first level contains at most twice the number of vertices at the zeroth level, so the first level contains at most two vertices. The second level contains at most four vertices, and so on.

Summing together each of these values, we get

$$\begin{aligned}\sum_{i=0}^{h} 2^i &= 2^0 + 2^1 + 2^2 + \cdots + 2^h \\ &= \frac{1 - 2^{h+1}}{1 - 2} \\ &= 2^{h+1} - 1.\end{aligned}$$

Thus, a binary tree of height h contains at most $2^{h+1} - 1$ vertices. □

Both Theorems 12 and 13 can be generalized from binary trees to m-ary trees with the appropriate modifications. We will not present the generalizations here, but you should attempt to discover them yourself!