4. Introduction to Trees

4.1. Definition of a Tree.

DEFINITION 4.1.1. *A* **tree** *is a connected, undirected graph with no simple circuits.*

Discussion

For the rest of this chapter, unless specified, a *graph* will be understood to be undirected and simple. Recall that a simple circuit is also called a cycle. A graph is **acyclic** if it does not contain any cycles. A tree imposes two conditions on a (simple) graph: that is be connected and acyclic.

4.2. Examples.

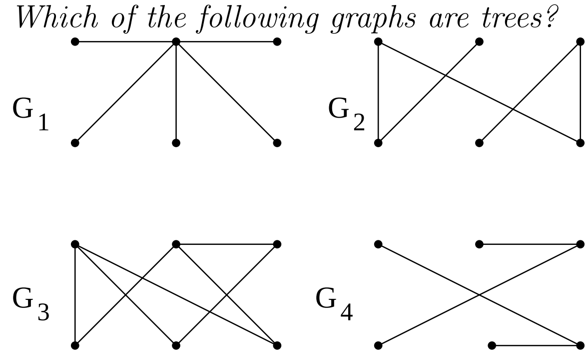
EXAMPLE 4.2.1. *The following are examples of trees*

- *Family tree*
- *File/directory tree*
- *Decision tree*
- *Organizational charts*

Discussion

You have most likely encountered examples of trees: your family tree, the directory or folder tree in your computer. You have likely encountered trees in other courses as well. When you covered counting principals and probability in precalculus you probably used trees to demonstrate the possible outcomes of some experiment such as a coin toss.

EXERCISE 4.2.1. *Which of the following graphs are trees?*



THEOREM 4.2.1. *A graph is a tree iff there is a unique simple path between any two of its vertices.*

PROOF. Suppose T is a tree and suppose u and v are distinct vertices in T. T is connected since it is a tree, and so there is a simple path between u and v. Suppose there are two different simple paths between u and v, say

$$P_1 : u = u_0, u_1, u_2, \dots u_m = v$$

and

$$P_2 : u = v_0, v_1, v_2, \dots v_n = v.$$

(Which type of proof do you think we are planning to use?)

Since the paths are different and since P_2 is a simple path, P_1 must contain an edge that isn't in P_2. Let $j \geq 1$ be the first index for which the edge $\{u_{j-1}, u_j\}$ of P_1 is not an edge of P_2. Then $u_{j-1} = v_{j-1}$. Let u_k be the first vertex in the path P_1 after u_{j-1} (that is, $k \geq j$) that is in the path P_2. Then $u_k = v_\ell$ for some $\ell \geq j$. We now have two simple paths, $Q_1 : u_{j-1}, ..., u_k$ using edges from P_1 and $Q_2 : v_{j-1}, ..., v_\ell$ using edges from P_2, between $u_{j-1} = v_{j-1}$ and $u_k = v_\ell$. The paths Q_1 and Q_2 have no vertices in common, other than the first and last, and no edges in common. Thus, the path from u_{j-1} to u_k along Q_1 followed by the path from v_ℓ to v_{j-1} along the reverse of Q_2 is a simple circuit in T, which contradicts the assumption that T is a tree. Thus, the path from u to v must be unique proving a tree has a unique path between any pair of vertices.

Conversely, assume G is not a tree.

(What kind of proof are we setting up for the reverse direction?)

Then either (a) G is not connected, so there is no path between some pair of vertices, or (b) G contains a simple circuit.

(a) Suppose G is not connected. Then there are two vertices u and v that can not be joined by a path, hence, by a simple path.
(b) Suppose G contains a simple circuit $C : v_0, ..., v_n$, where $v_0 = v_n$. If $n = 1$, then C would be a loop which is not possible since G is simple. Thus we have $n \geq 2$. But, since $n \geq 2$, $v_1 \neq v_0$, and so we have two different simple paths from v_0 to v_1: one containing the single edge $\{v_0, v_1\}$, and the other the part of the reverse of C from $v_n = v_0$ back to v_1.

Thus we have proved the statement "If a graph G is not a tree, then either there is no simple path between some pair of vertices of G or there is more than one simple path between some pair of vertices of G." This, is the contrapositive of the statement "If there is a unique simple path between any two vertices of a graph G, then G is a tree."

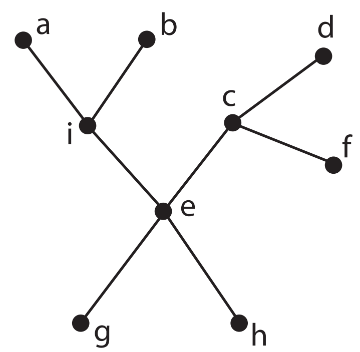
□

4.3. Roots.

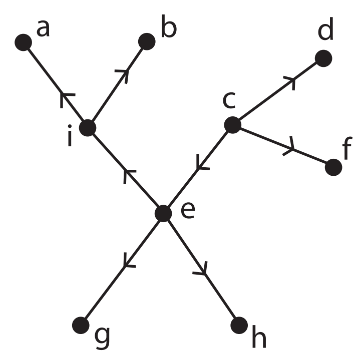
DEFINITION 4.3.1. *A* **rooted tree** *is a tree T together with a particular vertex designated as it* **root**. *Any vertex may a priori serve as the root. A rooted tree provides each edge with a direction by traveling away from the root.*

4.4. Example 4.4.1.

EXAMPLE 4.4.1. *Consider the tree below.*



We choose c to be the root. Then we have the directed tree:



Discussion

A rooted tree has a natural direction on its edges: given the root, v, an edge $e = \{x, y\}$ lies in the unique simple path from either v to x or from v to y, but *not both*. Say e is in the simple path from v to y. Then we can direct e from x to y.

4.5. Isomorphism of Directed Graphs.

DEFINITION 4.5.1. *Let G and H be a directed graphs. G and H are* **isomorphic** *if there is a bijection $f : V(G) \to V(H)$ such that (u, v) is an edge in G if and only if $(f(u), f(v))$ is an edge in H. We call the map f and isomorphism and write $G \simeq H$*

Discussion

Notice the notation for *ordered pairs* is used for the edges in a directed graph and that the isomorphism must preserve the direction of the edges.

EXERCISE 4.5.1. *Let $\mathcal{G}$ be a set of directed graphs. Prove that isomorphism of directed graphs defines an equivalence relation on $\mathcal{G}$.*

4.6. Isomorphism of Rooted Trees.

DEFINITION 4.6.1. *Rooted trees T_1 and T_2 are* **isomorphic** *if they are isomorphic as directed graphs.*

Discussion

EXERCISE 4.6.1. *Give an example of rooted trees that are isomorphic as (undirected) simple graphs, but not isomorphic as rooted trees.*

EXERCISE 4.6.2. *How many different isomorphism types are there of (a) trees with four vertices? (b) rooted trees with four vertices?*

4.7. Terminology for Rooted Trees.

DEFINITION 4.7.1.

1. *If $e = (u, v)$ is a directed edge then u is the* **parent** *of v and v is the* **child** *of u. The root has no parent and some of the vertices do not have a child.*
2. *The vertices with no children are called the* **leaves** *of the (rooted) tree.*
3. *If two vertices have the same parent, they are* **siblings**.
4. *If there is a directed, simple path from u to v then u is an* **ancestor** *of v and v is a* **descendant** *of u.*
5. *Vertices that have children are* **internal vertices**.

Discussion

Much of the terminology you see on this slide comes directly from a family tree. There are a few exceptions. For example, on a family tree you probably would not say cousin Freddy, who has never had kids, is a "leaf."

4.8. m-ary Tree.

DEFINITION 4.8.1.
1. *An m-**ary tree** is one in which every internal vertex has no more than m children.*
2. *A **full** m-**ary tree** is a tree in which every internal vertex has exactly m children.*

Discussion

Notice the distinction between an m-ary tree and a full m-ary tree. The first may have fewer than m children off of some internal vertex, but a latter must have exactly m children off of each internal vertex. A full m-ary tree is always an m-ary tree. More generally, if $k \leq m$ then every k-ary tree is also an m-ary tree. Caution: terminology regarding m-ary trees differs among authors.

4.9. Counting the Elements in a Tree.

THEOREM 4.9.1. *A tree with n vertices has $n-1$ edges.*

THEOREM 4.9.2.
1. *A full m-ary tree with i internal vertices has $n = mi + 1$ vertices.*
2. *If a full m-ary tree has n vertices, i internal vertices, and L leaves then*
   (a) $i = (n-1)/m$
   (b) $L = [n(m-1)+1]/m$
   (c) $L = i(m-1)+1$
   (d) $n = (mL-1)/(m-1)$
   (e) $i = (L-1)/(m-1)$

Discussion

PROOF OF THEOREM 4.9.1. Select a root v and direct the edges away from v. Then there are as many edges as there are terminal vertices of the edges and every vertex except v is a terminal vertex of some edge. □

PROOF OF THEOREM 4.9.2 PART 1. Every internal vertex has m children; hence, there are mi children. Only the root is not counted, since it is the only vertex that is not a child. □

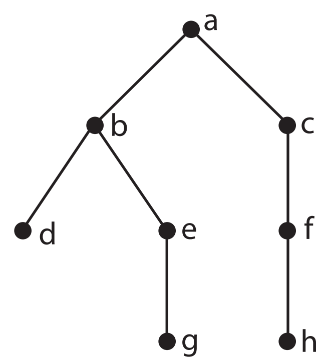
EXERCISE 4.9.1. *Prove Theorem 4.9.2 Part 2 [Hint: What is $L + i$?]*

4.10. Level.

DEFINITION 4.10.1.
1. *The* **level** *of a vertex in a rooted tree is the length of the shortest path to the root. The root has level 0.*
2. *The* **height** *of a rooted tree is the maximal level of any vertex.*
3. *A rooted tree of height h is* **balanced** *if each leaf has level h or $h-1$.*

Discussion

EXAMPLE 4.10.1. *Consider the rooted tree below*



If a is the root of the tree above, then

- *the level of b and c is 1.*
- *The level of g and h is 3.*
- *The height of the tree is 3.*
- *This is also a balanced tree since the level of every leaf is either 2 or 3.*

4.11. Number of Leaves.

THEOREM 4.11.1. *An m-ary tree of height h has at most m^h leaves.*

COROLLARY 4.11.1.1.
1. *If T is an m-ary tree with L leaves and height h, then*

$$h \geq \lceil \log_m L \rceil.$$

2. *If T is full and balanced, then*

$$h = \lceil \log_m L \rceil.$$

Discussion

Notice the *at most* in Theorem 4.11.1.

Proof of Theorem 4.11.1. We prove the theorem by induction on the height h, $h \geq 0$.

Basis: If the height of the tree is 0, then the tree consists of a single vertex, which is also a leaf, and $m^0 = 1$.

Induction hypothesis: Assume any m-ary tree of height h has at most m^h leaves for some positive integer m.

Induction step: Prove that an m-ary tree of height $h+1$ has at most m^{h+1} leaves.

Let T be an m-ary tree of height $h+1$, $h \geq 0$. Remove *all* the leaves of T to get a tree T'. T' is an m-ary tree of height h, and so by the induction hypothesis it has at most m^h leaves. We recover T from T' by adding back the deleted edges, and there are at most m edges added to each leaf of T'. Thus, T has at most $m \cdot m^h = m^{h+1}$ leaves, since no leaf of T' is a leaf of T.

By the principle of mathematical induction any m-ary tree with height h has at most m^h leaves for any positive integers m and h.

□

Exercise 4.11.1. *Prove Corollary 4.11.1.1.*

4.12. Characterizations of a Tree.

Theorem 4.12.1. *Let G be a graph with at least two vertices. Then the following statements are equivalent.*

1. *G is a tree*
2. *For each pair of distinct vertices in G, there is a unique simple path between the vertices.*
3. *G is connected, but if one edge is removed the resulting graph is disconnected.*
4. *G is acyclic, but if an edge is added between two vertices in G the resulting graph contains a cycle.*
5. *G is connected and the number of edges is one less than the number of vertices.*
6. *G is acyclic and the number of edges is one less than the number of vertices.*

Discussion

Theorem 4.12.1 gives us many different tools for recognizing trees. Once we prove this Theorem, it is enough to prove a graph satisfies any one of the statements to show the graph is a tree. Equivalently we could show a graph fails to satisfy any of the conditions to show it is not a tree.

The equivalence 1⇔2 is actually Theorem 4.2.1 which we have already proven. The equivalence 1⇔3 will be part of your graded assignment.

PROOF 1⇔4. First we show 1⇒4.

Let G be a tree with at least two vertices. By the definition of a tree G is acyclic, so what we must show is if an edge is added between two vertices of G then resulting graph contains a cycle. Let u and v be two vertices in G and add an edge, e, between these vertices. Let G' be the resulting graph. Note that G' is not necessarily simple.

By part 2 of this Theorem there must be a unique simple path between v and u in G. Let $e_1, e_2, e_3, \ldots, e_k$ be the edge sequence defining this path. The edge sequence $e_1, e_2, e_3, \ldots, e_k, e$ will define a path from v to itself in G'. Moreover, this is a simple circuit since $e_1, e_2, e_3, \ldots, e_k$ defined a simple path in G and e is not an edge in G. This shows G' contains a cycle.

Now we show 4⇒1.

Let G be an acyclic simple graph that satisfies the property given in part 4 of Theorem 4.12.1. Since we already know G is acyclic, what we need to show to complete the proof that G is a tree is that it is connected. To show a graph is connected we show two arbitrary vertices are connected by a path in G.

Let u and v be vertices in G. Add an edge, $e = \{u, v\}$, to G and call the resulting graph G'. By our assumption, there must be a cycle in G' now. In fact, the edge e must be part of the cycle because without this edge we would not have a cycle. Suppose $e, e_1, e_2, \ldots, e_k$ is the cycle that begins and ends at u. Notice that k must be at least 1 for this edges sequence to define a cycle. Moreover, the edge sequence $e_1, e_2, \ldots, e_k$ defines a path between v and u in G since all of these edges must be in G. This shows G is connected and so is a tree. □

EXERCISE 4.12.1. *Prove 1⇔5 in Theorem 4.12.1.*

EXERCISE 4.12.2. *Prove 1⇔6 in Theorem 4.12.1.*