PROFESSOR: Trees are about the most basic data structure that you're ever going to come across. They pervade computer science and other subjects. So let's talk about them.

And the simplest definition of a tree is that a tree is a connected graph with no cycles. In this setting we're talking about simple graphs, and trees with undirected edges. Well, in order to make sense of that, we better have a definition of a cycle. There's a picture of a typical tree, but to be precise, what's a cycle in a simple graph? Well, it's a closed walk of length greater than 2 that doesn't cross itself.

So the not crossing itself is the standard definition of a cycle that we were using in a directed graph. It simply means that it's a path, except that the beginning and end vertex are the same. So it looks like you start someplace at v, and then you go around to a and to w, and it's all distinct vertices as you go around in this path. Except that the path ends where it starts at v, which is what keeps it from being a path and makes it a cycle.

Now, the length greater than 2 is what is the difference between the definition of cycle [? between ?] simple graphs and directed graphs. And in a directed graph, it's perfectly possible to have a self-loop of length, 1, that is an interesting and important kind of cycle to have. But we forbid them in simple graphs, because there's no way to avoid having a cycle of length 2, because you always have the ability to go back and forth across an edge-- that's not interesting.

And so we don't consider that to be a cycle. A cycle, then, has to be of length greater than 2. It also rules out the cycle of length 0, which you get by taking a vertex all by itself.

OK, with that technical definition, we now know what a cycle is in the simple graph, and we understand the definition of tree. Here are some more pictures of trees-- simple graphs with no cycles.

Now, they really come up all the time. And why is that? Well, there are family trees, which you may be familiar with-- where you're drawing a picture of the descendants of a given person, and they keep branching out in a tree structure, as traditionally displayed. There are search trees, which come up all the time in computer science, where you branch on the answer to some question, which tells you which way to search next.

There are game trees, which we've already discussed in this class, which are used to define games and strategies. There are parse trees, that come up in compiler technology, and in language theory. And there are spanning trees, which we're going to be talking about some today.

Now, in addition to these places where trees come up, there are a lot of different kinds of trees. There's rooted trees, where there's some designated vertex called the root, and you think of getting to all the other vertices from the root. There are ordered trees, where when you're at a given vertex, there's a distinct order in which the exit edges from a vertex-- there's a first one, and a second one, and a third one, or a left one, and a next, left, most, and so on, so that there is an order in which you can choose to leave the vertex.

There are a binary trees, in which each vertex has two ways out exactly-- or no ways out if it's a so-called leaf. And then there are a complete trees, whose definition is not important to us, because we're not going to consider any of these. There's also, by the way, directed trees in which edges have a direction, as in a [? di-graph ?] tree.

But we're not considering any of these. We're going to focus on so-called pure trees, which are unordered, unrooted, undirected, and that's what we're talking about. So let's examine some more properties of trees and equivalent definitions of trees. It will be important for theoretical reasons and convenience to know lots of different characterizations of trees.

So we're starting off with a definition which says, it's a connected simple graph with no cycles, but there's other ways to characterize it. So an edge in a simple graph is called a cut edge, if, when you remove it from the graph, two vertices that used to be connected-- that is used to have a path between them-- cease to have a path between them. So here's a simple graph illustration.

And that edge, e, is a cut edge, because if I delete it, then there are now two components. There used to be two vertices-- actually any of the vertices here used to be connected to any of the vertices there via that edge. But once I've deleted that edge, all of those vertices here and there that used to be connected no longer are. So that makes e a cut edge.

f is not a cut edge. Because even if I delete f, there is, in fact, still a path from every vertex to every other vertex here, so that f is not disconnecting anything. And as I say, it's still connected after you delete it.

So now we get a simple way to characterize trees in terms of cut edges-- because an edge is not a cut edge if and only if it's on a cycle. If you think about that, if it's on a cycle, and you cut an edge out of a cycle, then everything on the cycle is still connected by going the other way around the cycle that doesn't use that edge. And if it's not on a cycle, then, in fact, you can think through that deleting it means that there's not going to be two paths between two things at its endpoints. And so it will separate them.

OK, so another way, then, to define a tree is to say a tree is a connected graph where every edge is a cut edge-- that is as soon as you cut any edge out of a tree it stops being connected. That yields another way to say that something is a tree-- a tree is a simple graph that is connected and is edge-minimal, which, again, means that if you remove any edge, it stops having that property of being connected. So its an edge-minimal connected graph.

That's kind of the reason why trees are so important, because if you're trying to figure out a way to get a whole bunch of things-- a whole bunch of vertices-- connected, a tree is going to have the minimum number of edges that are sufficient to get them all connected. If you think about different nodes in a network that need to communicate with each other, and you want to know how many direct connections that there have to be between these communication centers in order for everybody to talk to everybody else, the answer is-- it's got to be a tree on n vertices. And a tree on n vertices is going to have exactly n minus 1 edges.

So that gives you still another equivalent definition of a tree. A tree is a connected graph that has n vertices, and n minus 1 edges.

A kind of dual way to think about it is that a tree is an acyclic graph that has as many edges as it possibly could without having any cycles. So typically, an acyclic graph might not be connected, but as long as it's not connected, you can keep adding edges that will connect things up without creating cycles. But the minute you get a tree, so that everything is connected, you can't add another edge.

So an edge maximal acyclic graph is still another way to characterize trees. And maybe the most useful way is to say that a graph, in which there is a unique path between any two vertices, is a tree. So of course, if there is a unique path-- in particular, there's a path, so all the vertices have to be connected. But what makes it a tree is that there aren't two different ways to connect between two vertices, because as soon as there were there would be a cycle.

And those are some of the basic ways that trees can be formulated equivalently. And in fact,

there's lots more, but this is enough for today.