Trees

Michael L. Littman

CS 22 2020

April 20 ★, 2020
Overview

Forests & Trees (11.11)
  Leaves, Parents & Children (11.11.1)
  Properties (11.11.2)
  Spanning Trees (11.11.3)
Acyclic graph

Cycle is a path $p$ where the first and last vertices are the same.
Acyclic graph

Cycle is a path $p$ where the first and last vertices are the same.

**Definition:** A *simple cycle* is a length $k \geq 3$ cycle with no repeated vertices (except the first and last).
Acyclic graph

Cycle is a path $p$ where the first and last vertices are the same.

**Definition:** A *simple cycle* is a length $k \geq 3$ cycle with no repeated vertices (except the first and last).

**Definition:** A graph is *acyclic* if it contains no simple cycles.
Acyclic graph

Cycle is a path $p$ where the first and last vertices are the same.

**Definition**: A *simple cycle* is a length $k \geq 3$ cycle with no repeated vertices (except the first and last).

**Definition**: A graph is *acyclic* if it contains no simple cycles.

In an acyclic graph, if there is a simple path from $u$ to $v$, there is only one such path.
Acyclic graph

Cycle is a path $p$ where the first and last vertices are the same.

**Definition:** A *simple cycle* is a length $k \geq 3$ cycle with no repeated vertices (except the first and last).

**Definition:** A graph is *acyclic* if it contains no simple cycles.

In an acyclic graph, if there is a simple path from $u$ to $v$, there is only one such path. If there were more than one such path, we could use it to build a simple cycle.
Tree

**Definition:** A *tree* is a connected acyclic graph.
Tree

**Definition:** A *tree* is a connected acyclic graph.
Tree

Definition: A tree is a connected acyclic graph.

Define the “connected to” relation in a graph $G$ on a pair of vertices $u$ and $v$ as whether there is a path from $u$ to $v$ or $u = v$. 
Tree

**Definition:** A *tree* is a connected acyclic graph.

Define the “connected to” relation in a graph $G$ on a pair of vertices $u$ and $v$ as whether there is a path from $u$ to $v$ or $u = v$. Symmetric?
**Tree**

**Definition:** A tree is a connected acyclic graph.

Define the “connected to” relation in a graph $G$ on a pair of vertices $u$ and $v$ as whether there is a path from $u$ to $v$ or $u = v$. Symmetric? Yes.
**Tree**

**Definition:** A *tree* is a connected acyclic graph.

Define the “connected to” relation in a graph $G$ on a pair of vertices $u$ and $v$ as whether there is a path from $u$ to $v$ or $u = v$. Symmetric? Yes. Reflexive?
Tree

Definition: A tree is a connected acyclic graph.

Define the “connected to” relation in a graph $G$ on a pair of vertices $u$ and $v$ as whether there is a path from $u$ to $v$ or $u = v$. Symmetric? Yes. Reflexive? Yes.
**Definition:** A tree is a connected acyclic graph.

Define the “connected to” relation in a graph $G$ on a pair of vertices $u$ and $v$ as whether there is a path from $u$ to $v$ or $u = v$. Symmetric? Yes. Reflexive? Yes. Transitive?
Tree

**Definition:** A *tree* is a connected acyclic graph.

Define the “connected to” relation in a graph $G$ on a pair of vertices $u$ and $v$ as whether there is a path from $u$ to $v$ or $u = v$. Symmetric? Yes. Reflexive? Yes. Transitive? Yes.
**Tree**

**Definition:** A *tree* is a connected acyclic graph.

Define the “connected to” relation in a graph $G$ on a pair of vertices $u$ and $v$ as whether there is a path from $u$ to $v$ or $u = v$. Symmetric? Yes. Reflexive? Yes. Transitive? Yes. Equivalence relation!
Tree

**Definition:** A tree is a connected acyclic graph.

Define the “connected to” relation in a graph $G$ on a pair of vertices $u$ and $v$ as whether there is a path from $u$ to $v$ or $u = v$. Symmetric? Yes. Reflexive? Yes. Transitive? Yes. Equivalence relation! Thus, “connected to” partitions the graph...
**Tree**

**Definition:** A *tree* is a connected acyclic graph.

Define the “connected to” relation in a graph $G$ on a pair of vertices $u$ and $v$ as whether there is a path from $u$ to $v$ or $u = v$. Symmetric? Yes. Reflexive? Yes. Transitive? Yes. Equivalence relation! Thus, “connected to” partitions the graph into “connected components”.
**Tree**

**Definition:** A *tree* is a connected acyclic graph.

Define the “connected to” relation in a graph $G$ on a pair of vertices $u$ and $v$ as whether there is a path from $u$ to $v$ or $u = v$. Symmetric? Yes. Reflexive? Yes. Transitive? Yes. Equivalence relation! Thus, “connected to” partitions the graph into “connected components”.

Since an acyclic graph can be decomposed into a set of trees (connected, acyclic), it can be called a forest.
Leaves

**Definition**: A *leaf* is a vertex $v$ in a tree such that $\deg(v) = 1$. 

Diagram: 
```
    a
   /|
  /  |
 /    |
f-----c-----d
   b
```

$21 / 105$
Leaves

**Definition:** A *leaf* is a vertex \( v \) in a tree such that \( \text{deg}(v) = 1 \).

An *n*-vertex star graph is a tree with one central vertex and \( n - 1 \) leaves.
Leaves

**Definition:** A *leaf* is a vertex $v$ in a tree such that $\deg(v) = 1$.

An $n$-vertex star graph is a tree with one central vertex and $n - 1$ leaves. An $n$-vertex chain is a simple path with 2 leaves.
Depth and parents

We can define any vertex of a tree to be its root.
Depth and parents

We can define any vertex of a tree to be its root.

Definition: Given a tree \( G \) and a choice of root \( r \in V(G) \), the depth of \( u \in V(G) \), \( \text{dep}_r(u) \) is the length of the simple path from \( r \) to \( u \).
Depth and parents

We can define any vertex of a tree to be its root.

![Tree Diagram](image)

**Definition**: Given a tree $G$ and a choice of root $r \in V(G)$, the depth of $u \in V(G)$, $\text{dep}_r(u)$ is the length of the simple path from $r$ to $u$.

Depth is well defined because every pair of nodes in a tree has a unique simple path between them.
Depth and parents

We can define any vertex of a tree to be its root.

Definition: Given a tree $G$ and a choice of root $r \in V(G)$, the depth of $u \in V(G)$, $\text{dep}_r(u)$ is the length of the simple path from $r$ to $u$.

Depth is well defined because every pair of nodes in a tree has a unique simple path between them.

Definition: Given a tree $G$ and a choice of root $r \in V(G)$, $u$ is the parent of $v$ if $(u, v) \in E(G)$ and $\text{dep}_r(u) = \text{dep}_r(v) - 1$. 
All my children

In a tree $G$ with root $r$, if $(u, v) \in E(G)$, $|\text{dep}_r(u) - \text{dep}_r(v)| = 1$. 
All my children

In a tree $G$ with root $r$, if $(u, v) \in E(G)$, $|\text{dep}_r(u) - \text{dep}_r(v)| = 1$.

Proof?
All my children

In a tree $G$ with root $r$, if $(u, v) \in E(G)$, $|\text{dep}_r(u) - \text{dep}_r(v)| = 1$.

Proof?

In a tree, every vertex (except the root) has exactly one parent.
All my children

In a tree $G$ with root $r$, if $(u, v) \in E(G)$, $|\text{dep}_r(u) - \text{dep}_r(v)| = 1$.

Proof?

In a tree, every vertex (except the root) has exactly one parent.

Proof?
All my children

In a tree $G$ with root $r$, if $(u, v) \in E(G)$, $|\text{dep}_r(u) - \text{dep}_r(v)| = 1$. Proof?

In a tree, every vertex (except the root) has exactly one parent. Proof?

**Definition:** If $u$ is the parent of $v$, we call $v$ a *child* of $u$. 
All my children

In a tree $G$ with root $r$, if $(u, v) \in E(G)$, $|\text{dep}_r(u) - \text{dep}_r(v)| = 1$.
Proof?

In a tree, every vertex (except the root) has exactly one parent.
Proof?

**Definition:** If $u$ is the parent of $v$, we call $v$ a *child* of $u$.

A (non-root) leaf has no children. Other vertices have one or more children.
Subgraph

Let $G$ be a graph.

Let $G'$ be a graph.

A subgraph $G'$ of $G$ is defined so that:
- $V(G') \subseteq V(G)$
- $E(G') \subseteq E(G)$
- $\forall (u, v) \in E(G'), u \in V(G')$ and $v \in V(G')$

Example facts:
- If $u$ is connected to $v$ in $G'$, then $u$ is connected to $v$ in $G$.
- All $n$-node graphs are subgraphs of a complete graph $K_n$.
- Every subgraph of an acyclic graph is acyclic.
Subgraph

Let $G$ be a graph. A subgraph $G'$ of $G$ is defined so that
Subgraph

Let $G$ be a graph. A subgraph $G'$ of $G$ is defined so that

$\forall (u,v) \in E(G')$, $u \in V(G')$ and $v \in V(G')$
Subgraph

Let $G$ be a graph. A subgraph $G'$ of $G$ is defined so that

- $V(G') \subseteq V(G)$
- $E(G') \subseteq E(G)$
Subgraph

Let $G$ be a graph. A subgraph $G'$ of $G$ is defined so that

- $V(G') \subseteq V(G)$
- $E(G') \subseteq E(G)$
- $\forall (u, v) \in E(G')$, $u \in V(G')$ and $v \in V(G')$
Subgraph

Let $G$ be a graph. A subgraph $G'$ of $G$ is defined so that

- $V(G') \subseteq V(G)$
- $E(G') \subseteq E(G)$
- $\forall (u, v) \in E(G'), u \in V(G')$ and $v \in V(G')$

Example facts:
Subgraph

Let $G$ be a graph. A subgraph $G'$ of $G$ is defined so that

- $V(G') \subseteq V(G)$
- $E(G') \subseteq E(G)$
- $\forall (u, v) \in E(G'), u \in V(G')$ and $v \in V(G')$

Example facts:
- If $u$ is connected to $v$ in $G'$, then $u$ is connected to $v$ in $G$. 
Subgraph

Let $G$ be a graph. A subgraph $G'$ of $G$ is defined so that

- $V(G') \subseteq V(G)$
- $E(G') \subseteq E(G)$
- $\forall (u, v) \in E(G'), u \in V(G')$ and $v \in V(G')$

Example facts:

- If $u$ is connected to $v$ in $G'$, then $u$ is connected to $v$ in $G$.
- All $n$-node graphs are subgraphs of a complete graph $K_n$. 
Subgraph

Let $G$ be a graph. A subgraph $G'$ of $G$ is defined so that

- $V(G') \subseteq V(G)$
- $E(G') \subseteq E(G)$
- $\forall (u, v) \in E(G'), u \in V(G')$ and $v \in V(G')$

Example facts:
- If $u$ is connected to $v$ in $G'$, then $u$ is connected to $v$ in $G$.
- All $n$-node graphs are subgraphs of a complete graph $K_n$.
- Every subgraph of an acyclic graph is acyclic.
Properties of trees

1. *Every connected subgraph is a tree.*
Properties of trees

1. *Every connected subgraph is a tree.* Proof by contradiction. If subgraph is not a tree, it has simple cycles.
Properties of trees

1. *Every connected subgraph is a tree.* Proof by contradiction. If subgraph is not a tree, it has simple cycles. But, then so must the original tree.
Properties of trees

1. *Every connected subgraph is a tree.* Proof by contradiction. If subgraph is not a tree, it has simple cycles. But, then so must the original tree.

2. *There is a unique path between every pair of vertices.*
Properties of trees

1. *Every connected subgraph is a tree.* Proof by contradiction. If subgraph is not a tree, it has simple cycles. But, then so must the original tree.

2. *There is a unique path between every pair of vertices.* Otherwise, we can make a simple cycle.
Properties of trees

1. *Every connected subgraph is a tree.* Proof by contradiction. If subgraph is not a tree, it has simple cycles. But, then so must the original tree.

2. *There is a unique path between every pair of vertices.* Otherwise, we can make a simple cycle.

3. *Adding an edge between nonadjacent vertices in a tree creates a graph with a simple cycle.*
Properties of trees

1. *Every connected subgraph is a tree.* Proof by contradiction. If subgraph is not a tree, it has simple cycles. But, then so must the original tree.

2. *There is a unique path between every pair of vertices.* Otherwise, we can make a simple cycle.

3. *Adding an edge between nonadjacent vertices in a tree creates a graph with a simple cycle.* All pairs of vertices have a unique (simple) path, so adding that edge makes a simple cycle.
Properties of trees

1. *Every connected subgraph is a tree.* Proof by contradiction. If subgraph is not a tree, it has simple cycles. But, then so must the original tree.

2. *There is a unique path between every pair of vertices.* Otherwise, we can make a simple cycle.

3. *Adding an edge between nonadjacent vertices in a tree creates a graph with a simple cycle.* All pairs of vertices have a unique (simple) path, so adding that edge makes a simple cycle.

4. *Removing any edge disconnects the graph.*
Properties of trees

1. *Every connected subgraph is a tree.* Proof by contradiction. If subgraph is not a tree, it has simple cycles. But, then so must the original tree.

2. *There is a unique path between every pair of vertices.* Otherwise, we can make a simple cycle.

3. *Adding an edge between nonadjacent vertices in a tree creates a graph with a simple cycle.* All pairs of vertices have a unique (simple) path, so adding that edge makes a simple cycle.

4. *Removing any edge disconnects the graph.* There is no longer a path between the endpoints of the edge.
Properties of trees

1. *Every connected subgraph is a tree.* Proof by contradiction. If subgraph is not a tree, it has simple cycles. But, then so must the original tree.

2. *There is a unique path between every pair of vertices.* Otherwise, we can make a simple cycle.

3. *Adding an edge between nonadjacent vertices in a tree creates a graph with a simple cycle.* All pairs of vertices have a unique (simple) path, so adding that edge makes a simple cycle.

4. *Removing any edge disconnects the graph.* There is no longer a path between the endpoints of the edge.

5. *If the tree has at least two vertices, then it has at least two leaves.*
Properties of trees

1. *Every connected subgraph is a tree.* Proof by contradiction. If subgraph is not a tree, it has simple cycles. But, then so must the original tree.

2. *There is a unique path between every pair of vertices.* Otherwise, we can make a simple cycle.

3. *Adding an edge between nonadjacent vertices in a tree creates a graph with a simple cycle.* All pairs of vertices have a unique (simple) path, so adding that edge makes a simple cycle.

4. *Removing any edge disconnects the graph.* There is no longer a path between the endpoints of the edge.

5. *If the tree has at least two vertices, then it has at least two leaves.* Chains are trees and have the smallest number of leaves.
Properties of trees

1. *Every connected subgraph is a tree.* Proof by contradiction. If subgraph is not a tree, it has simple cycles. But, then so must the original tree.

2. *There is a unique path between every pair of vertices.* Otherwise, we can make a simple cycle.

3. *Adding an edge between nonadjacent vertices in a tree creates a graph with a simple cycle.* All pairs of vertices have a unique (simple) path, so adding that edge makes a simple cycle.

4. *Removing any edge disconnects the graph.* There is no longer a path between the endpoints of the edge.

5. *If the tree has at least two vertices, then it has at least two leaves.* Chains are trees and have the smallest number of leaves.

6. *The number of vertices in a tree is one larger than the number of edges.*
Properties of trees

1. *Every connected subgraph is a tree.* Proof by contradiction. If subgraph is not a tree, it has simple cycles. But, then so must the original tree.

2. *There is a unique path between every pair of vertices.* Otherwise, we can make a simple cycle.

3. *Adding an edge between nonadjacent vertices in a tree creates a graph with a simple cycle.* All pairs of vertices have a unique (simple) path, so adding that edge makes a simple cycle.

4. *Removing any edge disconnects the graph.* There is no longer a path between the endpoints of the edge.

5. *If the tree has at least two vertices, then it has at least two leaves.* Chains are trees and have the smallest number of leaves.

6. *The number of vertices in a tree is one larger than the number of edges.* Can prove by induction.
Induction on graphs

There are subtleties when using induction on graphs.
Induction on graphs

There are subtleties when using induction on graphs. In particular, there’s the step where we say “assume $P(n)$ holds, we can show that $P(n+1)$ holds”.
Induction on graphs

There are subtleties when using induction on graphs. In particular, there’s the step where we say “assume $P(n)$ holds, we can show that $P(n + 1)$ holds”.

When we’re doing induction on graphs of type $X$, it’s tempting to define $P(n)$ to be “a property holds for all graphs of type $X$ with $n$ vertices.”
Induction on graphs

There are subtleties when using induction on graphs. In particular, there’s the step where we say “assume $P(n)$ holds, we can show that $P(n + 1)$ holds”.

When we’re doing induction on graphs of type X, it’s tempting to define $P(n)$ to be “a property holds for all graphs of type X with $n$ vertices.” But, that means we need to show that any graph of type X with $n + 1$ vertices can be constructed from some graph of type X with $n$ vertices using the operation in our proof.
Induction on graphs

There are subtleties when using induction on graphs. In particular, there’s the step where we say “assume $P(n)$ holds, we can show that $P(n + 1)$ holds”.

When we’re doing induction on graphs of type $X$, it’s tempting to define $P(n)$ to be “a property holds for all graphs of type $X$ with $n$ vertices.” But, that means we need to show that any graph of type $X$ with $n + 1$ vertices can be constructed from some graph of type $X$ with $n$ vertices using the operation in our proof. Sometimes that’s true, sometimes it isn’t.
Induction on graphs

There are subtleties when using induction on graphs. In particular, there’s the step where we say “assume \( P(n) \) holds, we can show that \( P(n + 1) \) holds”.

When we’re doing induction on graphs of type X, it’s tempting to define \( P(n) \) to be “a property holds for all graphs of type X with \( n \) vertices.” But, that means we need to show that any graph of type X with \( n + 1 \) vertices can be constructed from some graph of type X with \( n \) vertices using the operation in our proof. Sometimes that’s true, sometimes it isn’t.

To show it is true, we need a step in the proof that says
Induction on graphs

There are subtleties when using induction on graphs. In particular, there’s the step where we say “assume $P(n)$ holds, we can show that $P(n + 1)$ holds”.

When we’re doing induction on graphs of type $X$, it’s tempting to define $P(n)$ to be “a property holds for all graphs of type $X$ with $n$ vertices.” But, that means we need to show that any graph of type $X$ with $n + 1$ vertices can be constructed from some graph of type $X$ with $n$ vertices using the operation in our proof. Sometimes that’s true, sometimes it isn’t.

To show it is true, we need a step in the proof that says “Let $G$ be an arbitrary $n$-vertex graph of type $X$.”
Induction on graphs

There are subtleties when using induction on graphs. In particular, there’s the step where we say “assume $P(n)$ holds, we can show that $P(n + 1)$ holds”.

When we’re doing induction on graphs of type X, it’s tempting to define $P(n)$ to be “a property holds for all graphs of type X with $n$ vertices.” But, that means we need to show that any graph of type X with $n + 1$ vertices can be constructed from some graph of type X with $n$ vertices using the operation in our proof. Sometimes that’s true, sometimes it isn’t.

To show it is true, we need a step in the proof that says “Let $G$ be an arbitrary $n$-vertex graph of type X.” We perform an operation that turns our graph into an $n - 1$-vertex graph also of type X.
Induction on graphs

There are subtleties when using induction on graphs. In particular, there’s the step where we say “assume $P(n)$ holds, we can show that $P(n+1)$ holds”.

When we’re doing induction on graphs of type $X$, it’s tempting to define $P(n)$ to be “a property holds for all graphs of type $X$ with $n$ vertices.” But, that means we need to show that any graph of type $X$ with $n+1$ vertices can be constructed from some graph of type $X$ with $n$ vertices using the operation in our proof. Sometimes that’s true, sometimes it isn’t.

To show it is true, we need a step in the proof that says “Let $G$ be an arbitrary $n$-vertex graph of type $X$.” We perform an operation that turns our graph into an $n-1$-vertex graph also of type $X$. We then use the opposite of that operation in our induction step.
Induction on graphs

There are subtleties when using induction on graphs. In particular, there’s the step where we say “assume \( P(n) \) holds, we can show that \( P(n + 1) \) holds”.

When we’re doing induction on graphs of type \( X \), it’s tempting to define \( P(n) \) to be “a property holds for all graphs of type \( X \) with \( n \) vertices.” But, that means we need to show that any graph of type \( X \) with \( n + 1 \) vertices can be constructed from some graph of type \( X \) with \( n \) vertices using the operation in our proof. Sometimes that’s true, sometimes it isn’t.

To show it is true, we need a step in the proof that says “Let \( G \) be an arbitrary \( n \)-vertex graph of type \( X \).” We perform an operation that turns our graph into an \( n - 1 \)-vertex graph also of type \( X \). We then use the opposite of that operation in our induction step.

We call this argument “build-down induction.”
Induction on graphs

There are subtleties when using induction on graphs. In particular, there’s the step where we say “assume \( P(n) \) holds, we can show that \( P(n+1) \) holds”.

When we’re doing induction on graphs of type \( X \), it’s tempting to define \( P(n) \) to be “a property holds for all graphs of type \( X \) with \( n \) vertices.” But, that means we need to show that any graph of type \( X \) with \( n+1 \) vertices can be constructed from some graph of type \( X \) with \( n \) vertices using the operation in our proof. Sometimes that’s true, sometimes it isn’t.

To show it is true, we need a step in the proof that says “Let \( G \) be an arbitrary \( n \)-vertex graph of type \( X \).” We perform an operation that turns our graph into an \( n-1 \)-vertex graph also of type \( X \). We then use the opposite of that operation in our induction step.

We call this argument “build-down induction.” At least in CS22.
Good example of build-down induction

We use induction on the proposition
Good example of build-down induction

We use induction on the proposition

\[ P(n) ::= \text{there are } n - 1 \text{ edges in any } n\text{-vertex tree.} \]
Good example of build-down induction

We use induction on the proposition

\[ P(n) ::= \text{there are } n - 1 \text{ edges in any } n\text{-vertex tree.} \]

**Base case** \((n = 1)\): \(P(1)\) is true since a tree with 1 node has 0 edges and \(1 - 1 = 0\).
Good example of build-down induction

We use induction on the proposition

\[ P(n) ::= \text{there are } n - 1 \text{ edges in any } n\text{-vertex tree.} \]

**Base case** \((n = 1): \) \(P(1)\) is true since a tree with 1 node has 0 edges and \(1 - 1 = 0\).

**Inductive step:** Now, suppose that \(P(n)\) is true and consider an \((n + 1)\)-vertex tree \(T\).
Good example of build-down induction

We use induction on the proposition

\[ P(n) ::= \text{there are } n - 1 \text{ edges in any } n\text{-vertex tree.} \]

**Base case** \((n = 1)\): \(P(1)\) is true since a tree with 1 node has 0 edges and \(1 - 1 = 0\).

**Inductive step**: Now, suppose that \(P(n)\) is true and consider an \((n + 1)\)-vertex tree \(T\). Let \(v\) be a leaf of the tree.
Good example of build-down induction

We use induction on the proposition

\[ P(n) ::= \text{there are } n - 1 \text{ edges in any } n\text{-vertex tree.} \]

**Base case** \((n = 1)\): \(P(1)\) is true since a tree with 1 node has 0 edges and \(1 - 1 = 0\).

**Inductive step**: Now, suppose that \(P(n)\) is true and consider an \((n + 1)\)-vertex tree \(T\). Let \(v\) be a leaf of the tree. *Deleting a vertex of degree 1 (and its edge) from any connected graph leaves a connected subgraph, so it’s a smaller tree*, and this smaller tree has \(n - 1\) edges by induction.
Good example of build-down induction

We use induction on the proposition

\[ P(n) ::= \text{there are } n - 1 \text{ edges in any } n\text{-vertex tree.} \]

**Base case** \((n = 1)\): \(P(1)\) is true since a tree with 1 node has 0 edges and \(1 - 1 = 0\).

**Inductive step**: Now, suppose that \(P(n)\) is true and consider an \((n + 1)\)-vertex tree \(T\). Let \(v\) be a leaf of the tree. Deleting a vertex of degree 1 (and its edge) from any connected graph leaves a connected subgraph, so it’s a smaller tree, and this smaller tree has \(n - 1\) edges by induction. If we reattach the vertex \(v\) and its edge, we find that \(T\) has \(n = (n + 1) - 1\) edges.
Good example of build-down induction

We use induction on the proposition

\[ P(n) ::= \text{there are } n - 1 \text{ edges in any } n\text{-vertex tree.} \]

**Base case** \((n = 1)\): \(P(1)\) is true since a tree with 1 node has 0 edges and \(1 - 1 = 0\).

**Inductive step**: Now, suppose that \(P(n)\) is true and consider an \((n + 1)\)-vertex tree \(T\). Let \(v\) be a leaf of the tree. Deleting a vertex of degree 1 (and its edge) from any connected graph leaves a connected subgraph, so it’s a smaller tree, and this smaller tree has \(n - 1\) edges by induction. If we reattach the vertex \(v\) and its edge, we find that \(T\) has \(n = (n + 1) - 1\) edges. Hence, \(P(n + 1)\) is true, and the induction proof is complete.
Failed example without build-down induction

False statement: If every vertex in a graph has positive degree, then the graph is connected.
Failed example without build-down induction

False statement: If every vertex in a graph has positive degree, then the graph is connected.

Let $P(n)$ be the proposition that if every vertex in an $n$-vertex graph has positive degree, then the graph is connected.
Failed example without build-down induction

False statement: If every vertex in a graph has positive degree, then the graph is connected.

Let $P(n)$ be the proposition that if every vertex in an $n$-vertex graph has positive degree, then the graph is connected.

**Base cases** ($n \leq 2$): In a graph with 1 vertex, that vertex cannot have positive degree, so $P(1)$ holds vacuously.
Failed example without build-down induction

False statement: If every vertex in a graph has positive degree, then the graph is connected.

Let $P(n)$ be the proposition that if every vertex in an $n$-vertex graph has positive degree, then the graph is connected.

**Base cases** ($n \leq 2$): In a graph with 1 vertex, that vertex cannot have positive degree, so $P(1)$ holds vacuously. $P(2)$ holds because there is only one graph with two vertices of positive degree, namely, the graph with an edge between the vertices, and this graph is connected.
Failed induction step

**Inductive step:** We must show that $P(n)$ implies $P(n + 1)$ for all $n \geq 2$. 
Failed induction step

**Inductive step:** We must show that $P(n)$ implies $P(n + 1)$ for all $n \geq 2$. Consider an $n$-vertex graph in which every vertex has positive degree.
Failed induction step

**Inductive step:** We must show that $P(n)$ implies $P(n + 1)$ for all $n \geq 2$. Consider an $n$-vertex graph in which every vertex has positive degree. *. By the assumption $P(n)$, this graph is connected; that is, there is a path between every pair of vertices.
Failed induction step

**Inductive step**: We must show that $P(n)$ implies $P(n + 1)$ for all $n \geq 2$. Consider an $n$-vertex graph in which every vertex has positive degree. *. By the assumption $P(n)$, this graph is connected; that is, there is a path between every pair of vertices. Now, we add one more vertex $x$ to obtain an $(n + 1)$-vertex graph.
Failed induction step

**Inductive step:** We must show that $P(n)$ implies $P(n + 1)$ for all $n \geq 2$. Consider an $n$-vertex graph in which every vertex has positive degree. *. By the assumption $P(n)$, this graph is connected; that is, there is a path between every pair of vertices. Now, we add one more vertex $x$ to obtain an $(n + 1)$-vertex graph. All that remains is to check that there is a path from $x$ to every other vertex $z$. 

*Missing step:* 
Deleting a vertex of degree 1 (and its edge) from any graph with positive degree leaves a graph with positive degree. False.
Failed induction step

**Inductive step**: We must show that $P(n)$ implies $P(n + 1)$ for all $n \geq 2$. Consider an $n$-vertex graph in which every vertex has positive degree. By the assumption $P(n)$, this graph is connected; that is, there is a path between every pair of vertices. Now, we add one more vertex $x$ to obtain an $(n + 1)$-vertex graph. All that remains is to check that there is a path from $x$ to every other vertex $z$. Since $x$ has positive degree, there is an edge from $x$ to some other vertex, $y$. 

*Missing step:* 
Deleting a vertex of degree 1 (and its edge) from any graph with positive degree leaves a graph with positive degree. False.
Failed induction step

**Inductive step:** We must show that $P(n)$ implies $P(n + 1)$ for all $n \geq 2$. Consider an $n$-vertex graph in which every vertex has positive degree. By the assumption $P(n)$, this graph is connected; that is, there is a path between every pair of vertices. Now, we add one more vertex $x$ to obtain an $(n + 1)$-vertex graph. All that remains is to check that there is a path from $x$ to every other vertex $z$. Since $x$ has positive degree, there is an edge from $x$ to some other vertex, $y$. Thus, we can obtain a path from $x$ to $z$ by going from $x$ to $y$ and then following the path from $y$ to $z$, proving $P(n + 1)$. 
Failed induction step

**Inductive step:** We must show that $P(n)$ implies $P(n + 1)$ for all $n \geq 2$. Consider an $n$-vertex graph in which every vertex has positive degree. *By the assumption $P(n)$, this graph is connected; that is, there is a path between every pair of vertices.* Now, we add one more vertex $x$ to obtain an $(n + 1)$-vertex graph. All that remains is to check that there is a path from $x$ to every other vertex $z$. Since $x$ has positive degree, there is an edge from $x$ to some other vertex, $y$. Thus, we can obtain a path from $x$ to $z$ by going from $x$ to $y$ and then following the path from $y$ to $z$, proving $P(n + 1)$. By the principle of induction, $P(n)$ is true for all $n \geq 0$. 
Failed induction step

**Inductive step**: We must show that $P(n)$ implies $P(n + 1)$ for all $n \geq 2$. Consider an $n$-vertex graph in which every vertex has positive degree. * By the assumption $P(n)$, this graph is connected; that is, there is a path between every pair of vertices. Now, we add one more vertex $x$ to obtain an $(n + 1)$-vertex graph. All that remains is to check that there is a path from $x$ to every other vertex $z$. Since $x$ has positive degree, there is an edge from $x$ to some other vertex, $y$. Thus, we can obtain a path from $x$ to $z$ by going from $x$ to $y$ and then following the path from $y$ to $z$, proving $P(n + 1)$. By the principle of induction, $P(n)$ is true for all $n \geq 0$.

Missing step: * $=$ *Deleting a vertex of degree 1 (and its edge) from any graph with positive degree leaves a graph with positive degree*. False.
### Comparing the examples

<table>
<thead>
<tr>
<th></th>
<th>good</th>
<th>bad</th>
</tr>
</thead>
<tbody>
<tr>
<td>tree</td>
<td></td>
<td></td>
</tr>
<tr>
<td>graph</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Before we can apply inductive hypothesis, need to show we can make any graph type $X$ of using the graph operation from a smaller graph of type $X$.
## Comparing the examples

<table>
<thead>
<tr>
<th></th>
<th>good</th>
<th>bad</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X$</td>
<td>tree</td>
<td>graph with positive degrees</td>
</tr>
</tbody>
</table>
Comparing the examples

<table>
<thead>
<tr>
<th>$X$</th>
<th>$P(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>good</td>
<td>tree has $n - 1$ edges</td>
</tr>
<tr>
<td>bad</td>
<td>graph with positive degrees is connected</td>
</tr>
</tbody>
</table>
## Comparing the examples

<table>
<thead>
<tr>
<th>X</th>
<th>good</th>
<th>bad</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(n)$</td>
<td>tree has $n - 1$ edges</td>
<td>graph with positive degrees is connected</td>
</tr>
<tr>
<td>graph operation</td>
<td>add a vertex and...</td>
<td>edge to existing vertex</td>
</tr>
</tbody>
</table>
# Comparing the examples

<table>
<thead>
<tr>
<th>$X$</th>
<th>good</th>
<th>bad</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(n)$</td>
<td>tree</td>
<td>graph with positive degrees</td>
</tr>
<tr>
<td>graph operation</td>
<td>has $n - 1$ edges</td>
<td>is connected</td>
</tr>
<tr>
<td></td>
<td>add a vertex and...</td>
<td>edge to existing vertex</td>
</tr>
</tbody>
</table>

Before we can apply inductive hypothesis, need to show we can make any graph type $X$ of using the graph operation from a smaller graph of type $X$. 
Comparing the examples

<table>
<thead>
<tr>
<th>X</th>
<th>good</th>
<th>bad</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(n)</td>
<td>tree graph with positive degrees</td>
<td>any other connected graph</td>
</tr>
<tr>
<td>graph operation</td>
<td>has n – 1 edges</td>
<td>is connected</td>
</tr>
<tr>
<td></td>
<td>add a vertex and...</td>
<td>edge to existing vertex</td>
</tr>
</tbody>
</table>

Before we can apply inductive hypothesis, need to show we can make any graph type X of using the graph operation from a smaller graph of type X.

Do: Use induction to prove things about graphs.
Comparing the examples

<table>
<thead>
<tr>
<th>$X$</th>
<th>good</th>
<th>bad</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(n)$ graph operation</td>
<td>tree has $n - 1$ edges</td>
<td>graph with positive degrees</td>
</tr>
<tr>
<td></td>
<td>add a vertex and edge to existing vertex</td>
<td>is connected</td>
</tr>
</tbody>
</table>

Before we can apply inductive hypothesis, need to show we can make any graph type $X$ of using the graph operation from a smaller graph of type $X$.

Do: Use induction to prove things about graphs.
Don’t: Skip this step!
Definition

**Definition:** A *spanning tree* of a graph $G$ is a subgraph of $G$ that (1) is a tree and (2) includes all of the vertices of $G$. 
Definition

Definition: A **spanning tree** of a graph $G$ is a subgraph of $G$ that (1) is a tree and (2) includes all of the vertices of $G$. 
Crouching graph hidden tree

Theorem: Every connected graph contains a spanning tree.
Crouching graph hidden tree

**Theorem:** Every connected graph contains a spanning tree.

**Proof:** $G$ is a subgraph of $G$ that is connected and includes all of the vertices of $G$. 
Crouching graph hidden tree

**Theorem:** Every connected graph contains a spanning tree.

**Proof:** $G$ is a subgraph of $G$ that is connected and includes all of the vertices of $G$. It has $m = |E(G)|$ edges.
Crouching graph hidden tree

**Theorem:** Every connected graph contains a spanning tree.

**Proof:** $G$ is a subgraph of $G$ that is connected and includes all of the vertices of $G$. It has $m = |E(G)|$ edges. By the well-ordering principle, there must be a *smallest* graph $T$ with this property.
Crouching graph hidden tree

**Theorem:** Every connected graph contains a spanning tree.

**Proof:** $G$ is a subgraph of $G$ that is connected and includes all of the vertices of $G$. It has $m = |E(G)|$ edges. By the well-ordering principle, there must be a *smallest* graph $T$ with this property. $T$ must be a spanning tree.
Crouching graph hidden tree

**Theorem:** Every connected graph contains a spanning tree.

**Proof:** $G$ is a subgraph of $G$ that is connected and includes all of the vertices of $G$. It has $m = |E(G)|$ edges. By the well-ordering principle, there must be a *smallest* graph $T$ with this property.

$T$ must be a spanning tree. Since $T$ is a connected graph that includes all of the vertices of $G$, all we have to show is that $T$ is acyclic.
Crouching graph hidden tree

**Theorem:** Every connected graph contains a spanning tree.

**Proof:** $G$ is a subgraph of $G$ that is connected and includes all of the vertices of $G$. It has $m = |E(G)|$ edges. By the well-ordering principle, there must be a *smallest* graph $T$ with this property.

$T$ must be a spanning tree. Since $T$ is a connected graph that includes all of the vertices of $G$, all we have to show is that $T$ is acyclic.

Suppose to the contrary that $T$ contains a simple cycle $C$. 
Crouching graph hidden tree

**Theorem:** Every connected graph contains a spanning tree.

**Proof:** $G$ is a subgraph of $G$ that is connected and includes all of the vertices of $G$. It has $m = |E(G)|$ edges. By the well-ordering principle, there must be a *smallest* graph $T$ with this property.

$T$ must be a spanning tree. Since $T$ is a connected graph that includes all of the vertices of $G$, all we have to show is that $T$ is acyclic.

Suppose to the contrary that $T$ contains a simple cycle $C$. Removing any edge of the cycle results in a graph $T'$ that still includes all of the vertices of $G$ and is still connected.
Theorem: Every connected graph contains a spanning tree.

Proof: $G$ is a subgraph of $G$ that is connected and includes all of the vertices of $G$. It has $m = |E(G)|$ edges. By the well-ordering principle, there must be a smallest graph $T$ with this property.

$T$ must be a spanning tree. Since $T$ is a connected graph that includes all of the vertices of $G$, all we have to show is that $T$ is acyclic.

Suppose to the contrary that $T$ contains a simple cycle $C$. Removing any edge of the cycle results in a graph $T'$ that still includes all of the vertices of $G$ and is still connected. But, that violates the definition of $T$. 