

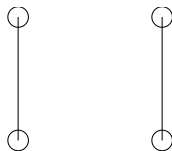
## Graph Theory III

### 1 Build-Up Error

Here is a false proof about connectivity. It exposes a very common flaw made on proofs by induction on graphs – it even has a name – it is known as “build-up error”.

**False Claim.** *If every vertex in a graph has degree at least 1, then the graph is connected.*

There are many counterexamples; here is one:

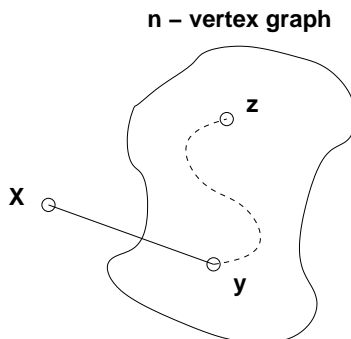


Since the claim is false, there must be at least one error in the following “proof”.

*Proof.* We use induction. Let  $P(n)$  be the proposition that if every vertex in an  $n$ -vertex graph has degree at least 1, then the graph is connected.

*Base case:* There is only one graph with a single vertex and it has degree 0. Therefore,  $P(1)$  is vacuously true, since the if-part is false.

*Inductive step:* We must show that  $P(n)$  implies  $P(n+1)$  for all  $n \geq 1$ . Consider an  $n$ -vertex graph in which every vertex has degree at least 1. 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 degree at least one, there is an edge from  $x$  to some other vertex; call it  $y$ . Thus, we can obtain a path from  $x$  to  $z$  by adjoining the edge  $x$ — $y$  to the path from  $y$  to  $z$ . This proves  $P(n + 1)$ .

By the principle of induction  $P(n)$  is true for all  $n \geq 1$ , which proves the theorem.  $\square$

That looks fine! Where is the bug? It turns out that faulty assumption underlying this argument is that *every*  $(n + 1)$ -vertex graph with minimum degree 1 can be obtained from an  $n$ -vertex graph with minimum degree 1 by adding 1 more vertex. Instead of starting by considering an arbitrary  $(n + 1)$ -node graph, this proof only considered an  $(n + 1)$ -node graph that you can make by starting with an  $n$ -node graph with minimum degree 1.

The counterexample above shows that this assumption is false; there is no way to build that 4-vertex graph from a 3-vertex graph with minimum degree 1. Thus the first error in the proof is the statement “This proves  $P(n + 1)$ ”.

More generally, this is an example of “build-up error”. Generally, this arises from a faulty assumption that every size  $n + 1$  graph with some property can be “built up” from a size  $n$  graph with the same property. (This assumption is correct for some properties, but incorrect for others— such as the one in the argument above.)

One way to avoid an accidental build-up error is to use a “shrink down, grow back” process in the inductive step: start with a size  $n + 1$  graph, remove a vertex (or edge), apply the inductive hypothesis  $P(n)$  to the smaller graph, and then add back the vertex (or edge) and argue that  $P(n + 1)$  holds. Let’s see what would have happened if we’d tried to prove the claim above by this method:

*Inductive step:* We must show that  $P(n)$  implies  $P(n + 1)$  for all  $n \geq 1$ . Consider an  $(n + 1)$ -vertex graph  $G$  in which every vertex has degree at least 1. Remove an arbitrary vertex  $v$ , leaving an  $n$ -vertex graph  $G'$  in which every vertex has degree... uh-oh!

The reduced graph  $G'$  might contain a vertex of degree 0, making the inductive hypothesis  $P(n)$  inapplicable! We are stuck— and properly so, since the claim is false!

## 2 Trees

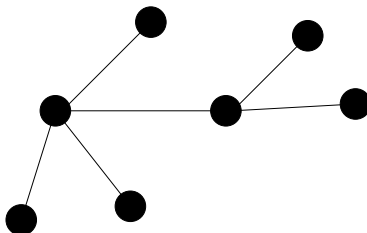
Today we’ll talk about a very special class of graphs called *trees*. Trees arise in all sorts of applications and you’ll see them in just about every computer science class that you’ll take at MIT. There are at least half a dozen ways to define a tree, but the simplest is the following.

**Definition.** A tree is a simple<sup>1</sup>, connected, acyclic graph.

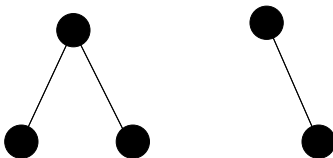
---

<sup>1</sup>Recall that we only consider simple graphs in this class, that is, graphs without loops or multiple edges.

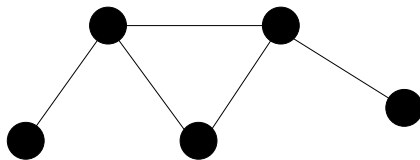
For example, a tree might look like this.



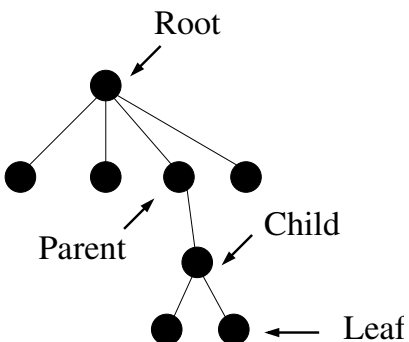
On the other hand, this is not a tree



and neither is this



Sometimes we'll draw trees in a leveled fashion, in which case we can identify the top node as the root, and every edge joints a "parent" to a "child".



The nodes at the bottom of degree 1 are called leaves.

**Definition.** A leaf is a node in a tree with degree 1.

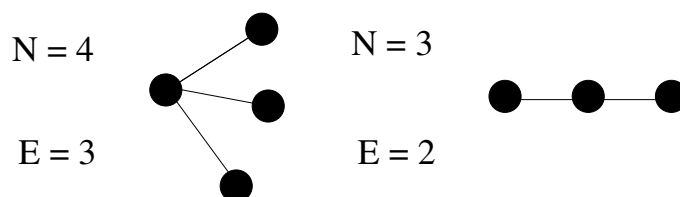
For example, in the tree above there are 5 leaves. It turns out that no matter how we draw a tree, every tree with more than 1 node always has some leaves.

**Theorem 1.** Every tree with more than 1 node has at least one leaf.

*Proof.* We prove the theorem by contradiction. Suppose it is not true. Then there exists a tree  $T = (V, E)$  where every node has degree at least 2. Let  $P = v_1, v_2, \dots, v_m$  be a path of maximum length in  $T$ , i.e., there is no path with  $m + 1$  or more nodes in  $T$  (recall that all nodes in a path are distinct by definition). Note that we are implicitly using the well-ordering principle here.

Since  $|V| \geq 2$  and  $T$  is connected, such a  $P$  exists and we have  $2 \leq m \leq n$ . Since  $T$  has minimum degree at least 2 by assumption,  $v_m$  is joined to at least 2 nodes. In particular, there is a node  $x \neq v_{m-1}$  such that  $\{v_m, x\} \in E$ . Since  $P$  has maximum length,  $v_1, v_2, \dots, v_m, x$  is not a path, which means  $x = v_i$  for some  $1 \leq i < m-1$ . But then  $v_i, v_{i+1}, \dots, v_{m-1}, v_m, v_i$  is a cycle in  $T$ . But  $T$  is a tree, and so by definition contains no cycle. This is a contradiction.  $\square$

As it turns out, every tree has at least 2 leaves, which you'll prove in the problem sets. There is also a close correlation between the number of nodes and number of edges in a tree. In the previous example we saw that  $N = 7$  and  $E = 6$ . We also have the following examples.

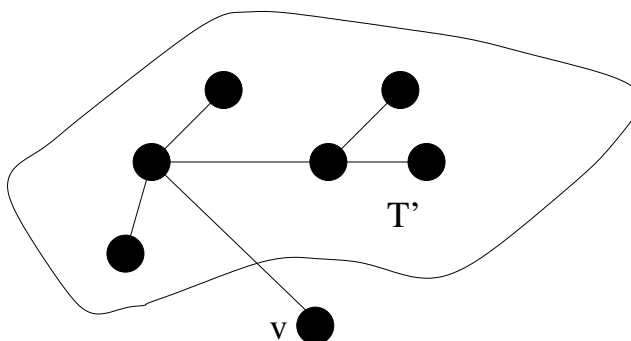


Naturally this leads us to guess,

**Theorem 2.** For any tree  $T = (V, E)$ ,  $|E| = |V| - 1$ .

*Proof.* We prove the theorem by induction on the number of nodes  $N$ . Our inductive hypothesis  $P(N)$  is that every  $N$ -node tree has exactly  $N - 1$  edges. For the base case, i.e., to show  $P(1)$ , we just note that every 1 node graph has no edges. Now assume that  $P(N)$  holds for some  $N \geq 1$ , and let's show that  $P(N + 1)$  holds.

Consider any  $(N + 1)$ -node tree  $T$ . Let  $v$  be a leaf of  $T$ . We know by the previous theorem that such a leaf  $v$  exists. Let  $T' = (V', E')$  be the tree formed by removing  $v$  and its incident edge from  $T$ .



We first claim that  $T'$  is in fact a tree. Since  $T$  is connected, so is  $T'$  since we have only removed a leaf. Moreover, removing vertices or edges cannot create any new cycles, so  $T'$  is also acyclic. Thus  $T'$  is a tree. By our inductive hypothesis, it follows that  $|E'| = |V'| - 1$ .

Now, we formed  $E'$  by removing one edge of  $E$ . Thus  $|E'| = |E| - 1$ . Similarly,  $|V'| = |V| - 1$ . Since  $|E'| = |V'| - 1$ , this means

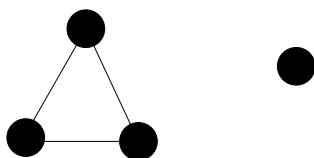
$$|E| - 1 = |V| - 1 - 1,$$

or  $|E| = |V| - 1$ , which shows  $P(N + 1)$ , and completes the proof.  $\square$

In fact, in the problems sets you will show the converse:

**Theorem 3.** *Any connected,  $N$ -node graph with  $N - 1$  edges is a tree.*

Note that we need to assume the graph is connected, as otherwise the following graph would be a counterexample.

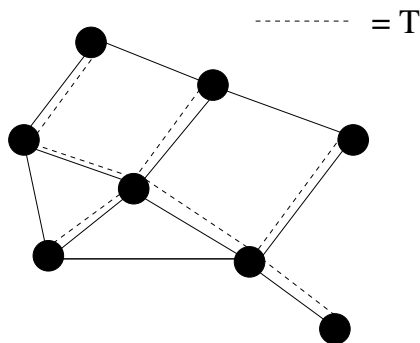


Besides this theorem, there are many other ways to characterize a tree, though we won't cover them here.

Graphs typically contain lots of trees as subgraphs. Of special interest are trees that contain every node of the graph. Such trees are called *spanning trees*.

**Definition.** *A tree  $T = (V, E)$  is a spanning tree for a graph  $G = (V', E')$  if  $V = V'$  and  $E \subseteq E'$ .*

The following figure shows a spanning tree  $T$  inside of a graph  $G$ .



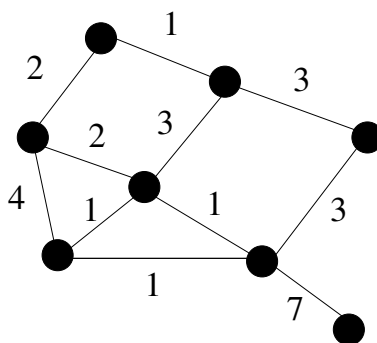
Spanning trees are interesting because they connect all the nodes of a graph using the smallest possible number of edges. For example, in the graph above there are 7 edges in

the spanning tree, while there are 8 vertices in the graph. It is not hard to show that any connected graph contains a spanning tree, and often lots of them.

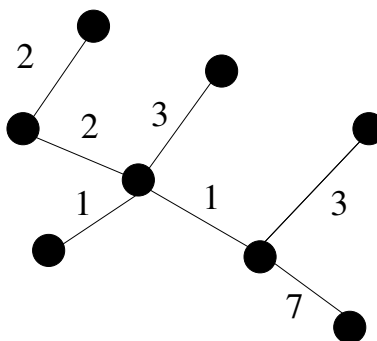
Spanning trees are very useful in practice, but in the real world, not all spanning trees are equally desirable. That's because, in practice, there are often costs associated with the edges of the graph.

For example, suppose the nodes of a graph represent buildings or towns and edges represent connections between buildings or towns. The cost to actually make a connection may vary a lot from one pair of buildings or towns to another. The cost might depend on distance or topography. For example, the cost to connect LA to NY might be much higher than that to connect NY to Boston. Or the cost of a pipe through Manhattan might be more than the cost of a pipe through a cornfield.

In any case, we typically represent the cost to connect pairs of nodes with a *weighted* edge, where the weight of the edge is its cost.



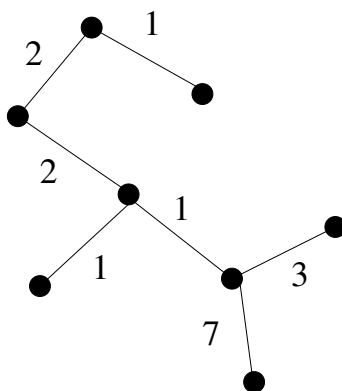
The weight of a spanning tree is then just the sum of the weights of the edges in the tree. For example, the weight of the following spanning tree is 19.



The goal, of course, is to find the spanning tree with minimum weight, called the *min-weight spanning tree* (MST) for short.

**Definition.** *The min-weight spanning tree (MST) of an edge-weighted graph  $G$  is the spanning tree of  $G$  with the smallest possible sum of edge weights.*

Is the spanning tree above an MST of the weighted graph we presented? Actually, it is not, since the following tree is also a spanning tree of our graph, of cost only 17.



Now is this spanning tree an MST? It seems to be, but how do we prove it? In general, how do we find an MST? We could, of course, enumerate all trees, but this could take forever for very large graphs.

Here are two possible algorithms:

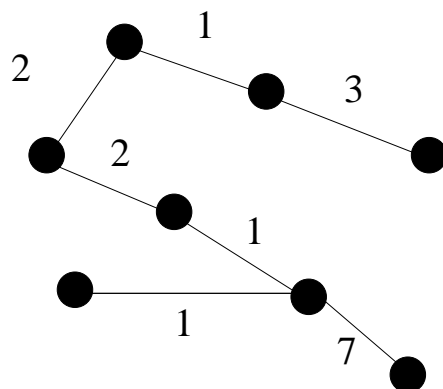
ALG1:

1. Grow a tree one edge at a time by adding the minimum weight edge of the graph to the tree, making sure that you have a tree at each step.

ALG2:

1. Select edges one at a time, always choosing the minimum weight edge that does not create a cycle with previously selected edges. We do not impose the constraint that the graph is a tree - our selected edges may indeed form a disconnected graph.
2. Continue until we get  $N - 1$  edges, i.e., a spanning tree.

For example, in the weighted graph we have been considering, we might run ALG1 as follows. We would start by choosing one of the weight 1 edges, since this is the smallest weight in the graph. Suppose we chose the weight 1 edge on the bottom of the triangle of weight 1 edges in our graph. This edge is incident to two weight 1 edges, a weight 4 edge, a weight 7 edge, and a weight 3 edge. We would then choose the incident edge of minimum weight. In this case, one of the two weight one edges. At this point, we cannot choose the third weight 1 edge since this would form a cycle, but we can continue by choosing a weight 2 edge. We might end up with the following spanning tree, which has weight 17, the smallest we've seen so far.



Now suppose we instead ran ALG2 on our graph. We might again choose the weight 1 edge on the bottom of the triangle of weight 1 edges in our graph. Now, instead of choosing one of the weight 1 edges it touches, we might choose the weight 1 edge on the top of the graph. Note that this edge still has minimum weight, and does not cause us to form a cycle, so ALG2 can choose it. We would then choose one of the remaining weight 1 edges. Note that neither causes us to form a cycle. Continuing the algorithm, we may end up with the same spanning tree as above, though this need not always be the case.

It turns out that both algorithms work, but they might end up with different MSTs. The MST is not necessarily unique - indeed, if all edges of an  $N$ -node graph have the same weight ( $= 1$ ), then all spanning trees have weight  $N - 1$ .

These are examples of greedy approaches to optimization. Sometimes it works and sometimes it doesn't. The good news is that it works to find the MST. In fact, both variations of ALG work. It's a little easier to prove it for ALG2 so we'll do that one here.

**Theorem.** *For any connected, weighted graph  $G$ , ALG2 produces an MST for  $G$ .*

*Proof.* The proof is a bit tricky. We need to show the algorithm terminates, i.e., if we have selected  $< N - 1$  edges, then we can always find an edge to add that does not create a cycle. We also need to show the algorithm creates a tree of minimum weight.

The key to doing all of this is to show that the algorithm never gets stuck or goes in a bad direction by adding an edge that will keep us from ultimately producing an MST. The natural way to prove this is to show that the set of edges selected at any point is contained in some MST - i.e., we can always get to where we need to be. We'll state this as a Lemma.

**Lemma 4.** *For any  $m \geq 0$ , let  $S$  consist of the first  $m$  edges selected by ALG2. Then there exists some MST  $T = (V, E)$  for  $G$  s.t.  $S \subseteq E$ , i.e., the set of edges we are growing is always contained in some MST.*

We'll prove this momentarily, but first let's see why it helps prove the theorem. Assume the lemma is true. Then how do we know ALG2 can always find an edge to add without creating a cycle? Well, as long as there are  $< N - 1$  edges picked, there exists some edge in  $T \setminus S$  and so the tree has no cycle. Next, how do we know that we get an MST at the end? Well, once  $M = N - 1$ , we know that  $S$  is a tree

Okay, so the theorem is an easy corollary of the lemma. Let's prove the lemma. Any ideas how to proceed? We'll use induction on the number of edges chosen by the algorithm so far. This is very typical in proving an algorithm preserves some kind of invariant condition - induct on the number of steps taken, i.e., the number of edges added.

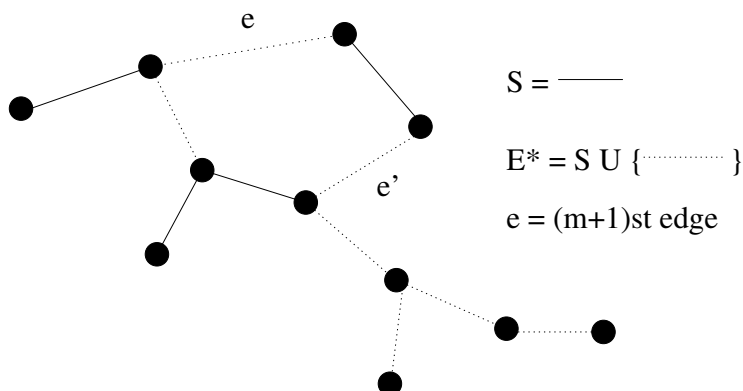
Our inductive hypothesis is the following:  $P(m)$ : For any  $G$  and any set  $S$  of  $m$  edges initially selected by ALG2, there exists an MST  $T = (V, E)$  of  $G$  such that  $S \subseteq E$ .

For the base case, we need to show  $P(0)$ . In this case  $S = \emptyset$ , so  $S \subseteq E$  trivially holds for any MST  $T = (V, E)$ .

For the inductive step, we assume  $P(m)$  holds and we show that it implies  $P(m+1)$ . Let  $e$  denote the  $(m+1)$ st edge selected by ALG2, and let  $S$  denote the first  $m$  edges selected by ALG2. Let  $T^* = (V^*, E^*)$  be the MST such that  $S \subseteq E^*$ , which exists by the inductive hypothesis. There are now two cases.

In the first case,  $e \in E^*$ , in which case  $S \cup \{e\} \subseteq E^*$ , and thus  $P(m+1)$  holds.

In the second case,  $e \notin E^*$ , as illustrated by the following diagram. Now we need to find a different MST that contains  $S$  and  $e$ .



What happens when we add  $e$  to  $T^*$ ? Well since  $T^*$  is a tree, we get a cycle. Moreover, the cycle cannot only contain edges in  $S$ , since  $e$  was chosen so that together with the edges in  $S$ , it does not form a cycle. This implies that  $e \cup T^*$  contains a cycle that contains an edge  $e'$  of  $E^* \setminus S$ . See the figure above.

Now, we claim that the weight of  $e$  is at most that of  $e'$ . Indeed, ALG2 picks the minimum weight edge that does not make a cycle with  $S$ . Note that  $e' \in T^*$  so it cannot make a cycle with  $S$ .

Okay, we're almost done. Now we'll make an MST that contains  $S \cup \{e\}$ . Let  $T^{**} = (V, E^{**})$  where  $E^{**} = (E^* - \{e'\}) \cup \{e\}$ , that is, we swap  $e$  and  $e'$  in  $T^*$ .

**Claim.**  $T^{**}$  is an MST.

*Proof.* We first show that  $T^{**}$  is a spanning tree. We have  $|E^{**}| = |E^*| = N - 1$  so  $T^{**}$  has  $N - 1$  edges. Moreover,  $T^{**}$  is still connected since we deleted an edge of  $e \cup T^*$  to form  $T^{**}$ , and that edge was on a cycle. It follows by Theorem 2 that  $T^{**}$  is a spanning tree.

Next, let's look at the weight of  $T^{**}$ . Well, since the weight of  $e$  was at most that of  $e'$ , the weight of  $T^{**}$  is at most that of  $T^*$ , and thus  $T^{**}$  is an MST.  $\square$

Since  $S \cup \{e\} \subseteq E^{**}$ ,  $P(m+1)$  holds. Thus, when ALG2 has  $N-1$  edges, it produces an MST.  $\square$

So now we know for sure that the MST for our example graph has weight 17 since it was produced by ALG2.