Math 3322: Graph Theory[1] Mikhail Lavrov

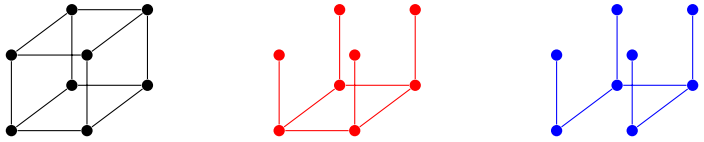
Lecture 10: Trees and spanning trees

September 12, 2024 *Kennesaw State University*

1 Spanning trees

What does it take to connect a graph?

We have seen many examples of connected graphs. For example, the cube graph (shown in the first diagram below) is connected:



But not all the edges of the cube graph are necessary to have a connected graph. For example, we can remove all edges between the four vertices in the "top half" of the cube, and the result is still connected, because those vertices can still get to each other through the bottom half. (Second diagram.)

Even that is not the best we can do. Remove any one of the edges in the bottom half, and the result is still connected: the bottom half of the cube forms a path subgraph. Now we have a connected subgraph of the cube graph that cannot lose any more edges and still remain connected. (Third diagram.)

1.1 Trees

In general, we say that a graph T is a **tree** if it is a *minimally connected* graph: T is connected, but for every edge $e \in E(T)$, the subgraph $T - e$ is no longer connected. Every edge of a tree is absolutely necessary to keep the tree connected.

The graph we drew in the third diagram above is a tree; moreover, it is what we call a spanning tree of the cube graph. A **spanning tree** of a graph G is a subgraph T which is a tree and satisfies $V(T) = V(G)$: T includes all the vertices of G.

The term **spanning subgraph** is sometimes used to mean a subgraph that has all the vertices of the original graph. Using this term, we can define a spanning tree of G as a spanning subgraph of G which is a tree.

Theorem 1.1. *A graph G is connected if and only if it has a spanning tree.*

Proof. Let G be any connected graph. To find a spanning tree T of G, we will delete edges of G one at a time until we get a tree.

[1] This document comes from an archive of the Math 3322 course webpage: http://misha.fish/archive/3322-fall-2024

This can be done essentially any way you like. Suppose we have ended up at an intermediate graph H (some spanning subgraph of G) and H has an edge e such that $H - e$ is still connected. Then just delete edge e and keep going. (If there are many options for e, pick any of them.)

Eventually, we stop (because there are only finitely many edges to delete), and the only way this process can stop is when deleting any edge would disconnect the remaining graph. That is exactly what it means to be a tree: we have arrived at a spanning tree of G.

In the other direction, suppose G has a spanning tree T. Then any two vertices v, w of G are also vertices of T, and T is connected, so there is a $v - w$ walk in T. That walk is also a $v - w$ walk in G, because T is a subgraph of G. Therefore G is connected. □

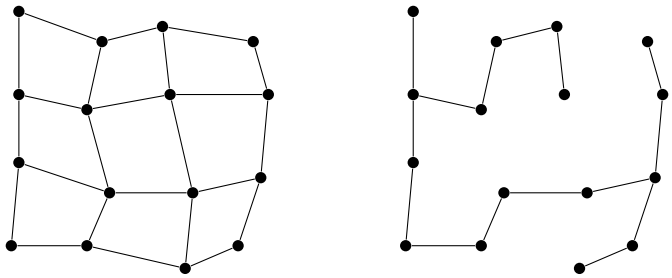
Theorem 1.1 is a useful distinguishing property of connected graphs; it's much easier to use than the best tools we had before. Checking the definition of a connected graph boils down to checking that for any two vertices v, w, there is a $v - w$ walk: we need to find many different objects in the graph.

1.2 Minimum-cost spanning trees

A spanning tree of G, on the other hand, is a *single* object which is enough to demonstrate that G is connected. All connected graphs have them—no other graphs do.

Spanning trees arise in many applications where we need to connect a graph as *cheaply* as possible. In this case, we consider a graph where each edge has a cost (a nonnegative real number) attached, and we want the spanning tree we find to minimize the sum of the costs of its edges. This is the problem of finding a **minimum-cost spanning tree** (MCST).

Below is an example of a graph and its minimum-cost spanning tree. In this example, the costs of the edges are taken to be the lengths of the edges as line segments in our diagram.[2]



Looking at the proof of Theorem 1.1, if we want to modify it to solve the MCST problem, we might try to use the edge costs to try to tell us *which* edges to delete until we get a spanning tree. Specifically, it seems reasonable to look at the edges in descending order of cost, and delete each edge we look at unless it must be kept to avoid disconnecting the graph.

This is a "greedy" algorithm: it does the best thing it sees in the moment, with no regard to how this affects future choices. Does it work? That should not be obvious yet! What if keeping a very expensive edge can allow us to delete multiple edges that are only a bit less expensive, later?

[2]This is a departure from the usual rule that the diagram we choose to represent the graph is irrelevant. It's still true that a different diagram could represent the same graph—but here, our model doesn't just include the graph, but also the edge costs.

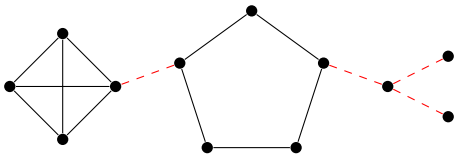
We will return to this problem, and prove that this algorithm works, after we learn a bit more about trees.

2 Bridges

Theorem 1.1 will be very useful to us—eventually. To get as much as possible out of this theorem, we need to learn more about trees. Fortunately, there are lots of things to learn.

To begin with, let's take another look at the way we found a spanning tree in the proof of Theorem 1.1. We kept deleting edges if deleting them would not disconnect the graph, until there were no more edges left that we could delete. What sort of edges do we get in this way?

In a connected graph G, an edge e is called a **bridge** if $G - e$ is not connected. For example, in the graph below, the dashed edges are the bridges:



Not all graphs have any bridges at all. However, as we delete edges of our graph, this might cause some of the remaining edges to become bridges. Eventually, we can't delete any more edges because all of the remaining edges are bridges. That's what it means to be a tree! (An equivalent phrasing of our definition is "T is a tree if all edges of T are bridges.")

So how do we tell if an edge is a bridge?

Theorem 2.1. *In a connected graph G, edge vw is a bridge if and only if it is not on any cycles.*

Proof. Suppose edge vw is part of a cycle: $(v_1, v_2, \dots, v_k, v_1)$ where $v_i = v$ and $v_{i+1} = w$. Our basic idea: edge vw is not necessary because we can always go "the long way" around this cycle.

Since G is connected, there are walks between every pair of vertices. In all of those walks, whenever edge vw is used (whenever the walk goes $\dots, v, w, \dots$) we can replace it by the longer sequence $v_i, v_{i-1}, \dots, v_1, v_k, v_{k-1}, \dots, v_{i+1}$. (If a walk goes $\dots, w, v, \dots$, use this sequence in reverse.)This gives walks between every pair of vertices, none of which use edge vw, so it gives walks between every pair of vertices in $G - vw$. Therefore $G - vw$ is still connected, and vw is not a bridge.

In the other direction, suppose vw is not a bridge. Then $G - vw$ is connected, and in particular, $G - vw$ contains a $v - w$ path: $(v_0, v_1, \dots, v_\ell)$ with $v_0 = v$ and $v_\ell = w$. Therefore G contains the cycle $(v_0, v_1, \dots, v_\ell, v_0)$ whose last edge is the edge vw. □

3 Properties of trees

What does Theorem 2.1 tell us about trees? In a tree T, every edge is a bridge, so no edge is part of any cycles. Therefore a tree T has no cycles at all.

In fact, this is another defining property of trees. If a graph G is connected and has no cycles, then by Theorem 2.1, every edge of G must be a bridge: deleting any edge of G disconnects it. Therefore G is a tree. We have shown the following:

Proposition 3.1. *A graph T is a tree if and only if it is connected and **acyclic** (has no cycles).*

There are many equivalent properties of trees. Here is one more:

Proposition 3.2. *A graph T is a tree if and only if it is acyclic, but adding any edge would create a cycle.*

Proof. Suppose T is a tree; by Proposition 3.1, we already know it is acyclic. Suppose v, w are *not* adjacent in T; we want to show that $T + vw$ (the graph we get if we add edge vw to T) has a cycle. Well, vw cannot be a bridge of $T + vw$: deleting it gives us the connected graph T. So by Theorem 2.1, $T + vw$ has a cycle (containing vw).

Suppose T is acyclic, and adding any edge would create a cycle. We want to prove that T is connected: then we can use Proposition 3.1 and conclude that T is a tree.

Suppose v, w are vertices of T; we want to know that there is a $v - w$ walk in T. If vw is an edge, then (v, w) is such a walk. If vw is not an edge, then $T + vw$ has a cycle. That cycle must include vw (because otherwise, the cycle would have existed in T). As in the proof of Theorem 2.1, deleting vw from that cycle leaves a $v - w$ path, finishing our proof that T is connected. □

Proposition 3.2 is description of trees that, in a way, is the opposite of the definition of a tree. The definition says that T is a tree if and only if it is minimally connected: connected with as few edges as possible. Proposition 3.2 says that T is a tree if and only if it is maximally acyclic: acyclic with as many edges as possible.

4 Minimum-cost spanning trees

Now, let's return to the problem of finding a minimum-cost spanning tree (MCST) of a graph in which every edge has an associated cost. We have a proposed algorithm:

1. Let $e_1, e_2, \ldots, e_m$ be a list of the edges of G in decreasing order of cost: from most expensive to cheapest. Also, let $G_0 = G$; we will construct a sequence $G_1, G_2, \ldots, G_m$ of graphs as we go.
2. Starting at $i = 1$, look at edge e_i and ask: is e_i a bridge of G_{i-1}?

   If e_i is a bridge, set $G_i = G_{i-1}$; if e_i is not a bridge, set $G_i = G_{i-1} - e_i$.
3. Repeat step 2 for $i = 2, 3, \ldots, m$, until all edges have been considered. Return G_m as the minimum-cost spanning tree.

We will assume that there are no ties between the costs of the edges. (One of the practice problems at the end of these notes will ask you to generalize our result to allow for ties.)

Theorem 4.1. *Starting from a connected graph G where each edge has a distinct cost, this algorithm will always produce a minimum-cost spanning tree: a spanning tree with the minimum sum of edge costs.*

Proof. The algorithm always produces a connected graph, because it never deletes a bridge. Also, the only edges that are kept are bridges. Specifically, if edge e_i survives to the final graph G_m, it must first survive to G_i, which means it must have been a bridge of G_{i-1}. Therefore there is no cycle in G_{i-1} containing e_i. To obtain G_m from G_{i-1}, we only delete edges; therefore G_m also cannot have a cycle containing e_i, and therefore G_m is a tree. Let's give G_m another name: call it T.

To prove that the algorithm is optimal, we need to compare T to an alternate spanning tree T', and somehow argue that T is better than T'. Specifically, what we'll do is prove that if $T' \neq T$, then T' cannot be the MCST, because we can make a small improvement to it. This will prove the theorem, because it leaves T as the only possible candidate for the MCST.

How can we do this? Well, first of all, if $T \neq T'$, we can find an edge e which is present in T, but not T'. (If every edge of T were present in T', then either T' would be equal to T, or T' would consist of T plus some additional edges. In the latter case, T' would not be a tree, because it would not be *minimally* connected.)

By Proposition 3.2, the graph $T' + e$ has a cycle. Let C be that cycle, and let e' be the most expensive edge of C.

By looking at the algorithm, we can prove that e' cannot be part of T, and in particular $e' \neq e$. Here's why: suppose that $e' = e_i$ in the list of edges by cost. In the graph G_{i-1} produced by the algorithm, e_i is still present, and so are all the other edges of C, because we have not gotten to any of them yet. Therefore C is a cycle of G_{i-1} containing e_i, meaning (by Theorem 2.1) that e_i is not a bridge of G_{i-1}. We conclude that e_i (that is, e') is deleted and does not survive to $T = G_m$.

Therefore we can modify T' as follows: add e, but then delete e'. This produces a cheaper connected graph!

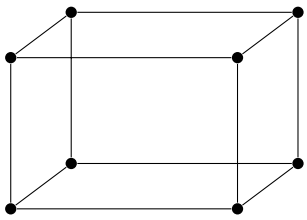
- Why is it cheaper? Because e' is the most expensive edge of C, and e is some other edge of C: we deleted a more expensive edge than we added.
- Why is it still connected? Because we deleted an edge from a cycle of $T' + e$, which (by Theorem 2.1) was not a bridge.

Actually, we'll be able to prove after the next lecture that $(T' + e) - e'$ is a tree itself. But either way, it certainly has a spanning tree by Theorem 1.1, and that spanning tree is cheaper than T'. Therefore T' cannot be the MCST.

Since this is true of any spanning tree other than T, we conclude that T is the unique spanning tree of G. □

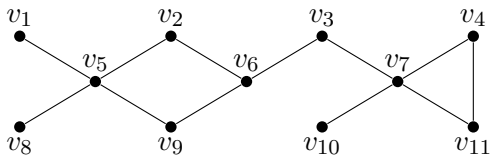
5 Practice problems

1. In the diagram of the cube graph shown below, each diagonal edge has length 1, each vertical edge has length 2, and each horizontal edge has length 3.



   Find a minimum-cost spanning tree of the cube graph, where the cost of each edge is taken to be its length in this diagram.

2. Let G be the graph shown below:



   (a) Identify all the bridges in G. *(Check: there should be* 5 *bridges.)*

   (b) Find all the possible spanning trees of G. *(How does the answer to (a) help here?)*

3. Prove that *every* n-vertex graph with the degree sequence $n-1, 1, 1, \ldots, 1, 1$ is a tree. What does such a graph look like?

4. Find a connected 3-regular graph G which has a bridge!

   (Hint: you'll need at least 10 *vertices.)*

5. It's even harder to find a 4-regular graph which has a bridge.

   (a) Let G be a 4-regular graph in which edge vw is a bridge. Then $G - vw$ should have two components: one containing v, and one containing w. Describe the degree sequences of each component.

   (b) Conclude that this can't happen: a 4-regular graph cannot have a bridge.

6. Prove that T is a tree if and only if between any two vertices of G, there is **exactly one** path. This is another of the many characterizations of trees!

7. Theorem 4.1 assumes that all edges of the graph have distinct costs: there are no ties.

   (a) Prove that if there *are* ties between the costs of some edges, then we can still conclude one of two things in the proof: either we find a spanning tree cheaper than T' (so T' cannot be the MCST) or $(T' + e) - e'$ is a spanning tree with the same cost as T', but "closer" to T in some sense.

   (b) Explain why this is enough to still deduce that T is an MCST (even if it might not be unique).