Lecture Notes

The last lecture introduced the idea of a graph, a mathematical object used extensively in computer science. In this lecture, we will look at two particularly useful types of graphs: trees and directed acyclic graphs.

1 Trees

Intuitively, a graph with very few edges cannot be connected. In the last lecture, we made this intuition precise with a theorem:

Theorem 1.1 A graph with k edges and n vertices has at least $n - k$ connected components.

This theorem has an important corollary:

Corollary 1.2 Every connected graph with n vertices has at least $n - 1$ edges.

Proof. The proof is by contradiction. Assume for the purpose of contradiction that there exists a connected graph G with n vertices that has fewer than $n-1$ edges. Then by the theorem, there are at least $n - (n-2) = 2$ connected components, contradicting the assumption that G is connected. ■

There are many connected graphs with n nodes and exactly $n-1$ edges, the minimum required by the corollary. One example is the line graph with n nodes. Another example is shown in Figure 1. In general, such graphs are called trees.

Definition A *tree* is a connected n-node graph with exactly $n - 1$ edges.

The vertices in a tree can be classified into two categories. Vertices of degree at most one are called *leaves*, and vertices of degree greater than one are called *internal nodes*.

Trees are usually drawn as in Figure 1 with the leaves on the bottom. Keep this convention in mind; otherwise, phrases like "all the vertices *below*..." will be confusing. (The English mathematician Littlewood once remarked that he found such directional terms particularly bothersome, since he habitually read mathematics reclined on his back!)

Trees arise in many problems. For example, the file structure in a computer system can be naturally represented by a tree. In this case, each internal node corresponds to a directory, and each leaf corresponds to a file. If one directory contains another, there there is an edge between the associated internal nodes. If a directory contains a file, then there there is an edge between the internal node and a leaf.

There are several ways to describe trees that are equivalent to the preceding definition.

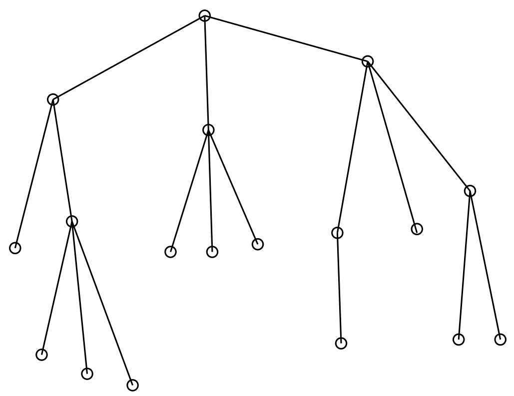

Figure 1: *This is a picture of a connected graph with $n = 18$ vertices and $n - 1 = 17$ edges. In general, a connected graph with n vertices and $n-1$ edges is called a* tree. *This tree has 11 "leaves", which are defined as vertices of degree at most one. The remaining 7 vertices are called "internal nodes".*

Theorem 1.3 Let G be a graph with n vertices. The following are equivalent:

1. G is a connected graph with exactly $n - 1$ edges.
2. G is an acyclic graph with exactly $n - 1$ edges.
3. G is a connected, acyclic graph.

Theorems asserting that several mathematical statements are equivalent are quite common. One way to prove such a theorem would be to consider every pair of statements in turn and prove them equivalent. That is, we would prove $1 \Leftrightarrow 2$, $2 \Leftrightarrow 3$, and $3 \Leftrightarrow 1$.

This is overkill. A simpler approach is to show $1 \Rightarrow 2$, $2 \Rightarrow 3$, and $3 \Rightarrow 1$. These three implications are sufficient to prove every pair of statements equivalent. For example, $2 \Leftrightarrow 3$ follows because we prove $2 \Rightarrow 3$ directly, and prove $3 \Rightarrow 2$ with the two steps $3 \Rightarrow 1 \Rightarrow 2$. The following lemma will be useful in proving two of the three implications.

Lemma 1.4 Every n-vertex graph with at least n edges contains a cycle.

Proof. The proof is by contradiction. Assume for the purpose of contradiction that there exists an acyclic n-vertex graph with at least n edges. Let G be a counterexample with a minimum number of vertices. (Such a graph exists by the well-ordering principle.) G can not contain a vertex of degree 0 or degree 1, since deleting that vertex would create a smaller counterexample and contradict the assertion that G is minimal.

Therefore, every vertex in G has degree at least 2. But now we can find a cycle in G. Begin at an arbitrary vertex and traverse an arbitrary edge. Exit each subsequent vertex via an edge not previously traversed. Since there are a finite number of edges, eventually we must enter a vertex v such that every edge has been previously traversed. Since v has degree at least 2, it must have been visited earlier on the path. This implies that graph G contains a cycle, contradicting the assertion that G is acyclic. ∎

Now we proceed with the proof of the theorem.

Proof. We prove the statements equivalent by showing $1 \Rightarrow 2$, $2 \Rightarrow 3$, and $3 \Rightarrow 1$.

$1 \Rightarrow 2$: Let G be a connected graph with exactly $n-1$ edges. We must prove that G is an acyclic graph with exactly $n-1$ edges. The number of edges is immediate; all that remains is to show that G is acyclic.

The proof is by contradiction. Assume for the purpose of contradiction that G contains a cycle $(x_1, x_2), (x_2, x_3), \ldots, (x_k, x_1)$. Since G is connected, there is a path between every pair of vertices. Now delete the edge (x_1, x_2), leaving only $n-2$ edges in the graph. If the path between a pair of vertices u and v contained this edge, then we can construct a new path between u and v by replacing the edge with the sequence of edges $(x_2, x_3), \ldots, (x_k, x_1)$ or the reverse sequence. This implies that the graph remains connected. But this contradicts Corollary 1.2, which states that every connected graph must have at least $n-1$ edges.

$2 \Rightarrow 3$: Let G be an acyclic graph with exactly $n-1$ edges. We must prove that G is a connected, acyclic graph. The acyclic condition is immediate; what remains is to prove that G is connected.

The proof is by contradiction. Assume for the purpose of contradiction that G is a disconnected. Then it must have at least two connected components. As a consequence, if every connected component has fewer edges then vertices, then the number of edges in G is at most $n-2$. This contradicts the assertion that G has exactly $n-1$ edges. On the other hand, if there is a connected component with as many edges as vertices, then the lemma implies a cycle, contradicting the assertion that G is acyclic.

$3 \Rightarrow 1$: Let G be a connected, acyclic graph. We must prove that G is a connected graph with exactly $n-1$ edges. Certainly, G is connected. The only thing left is to show that G has exactly $n-1$ edges.

Since G is connected, Corollary 1.2 implies that G has at least $n-1$ edges. Since G is acyclic, the lemma implies that G can not have at most $n-1$ edges. ∎

2 Special Kinds of Trees

Two species of tree arise particularly often in computer science: rooted trees and binary trees. A *rooted tree* is just a tree with one vertex designated as the "root". Given an ordinary tree, one can produce different rooted trees by choosing different vertices as the root. In a picture of a rooted tree, the topmost vertex usually represents the root.

Here is a nice property of rooted trees: from a vertex other than the root, there is a unique simple path to the root. (The graph would contain a cycle if there were more than one simple path, and the graph would be disconnected if there were no simple path. But Theorem 1.3 implies that every tree is both acyclic and connected.) If u is the first vertex on the unique simple path from vertex v to the root, then u is called the *parent* of v, and v is called the *child* of u.

Recall that a leaf of a tree is a vertex with degree at most one. In a rooted tree, therefore, a vertex with no children is a leaf. Furthermore, if the root has a single child or no children, then it is also a leaf.

There is a special type of rooted tree called a *binary tree*. Binary trees can be defined recursively as follows:

Definition A *binary tree* is a rooted tree consisting of either:

1. A single vertex.
2. A vertex connected to a binary tree called the *left subtree*.
3. A vertex connected to a binary tree called the *right subtree*.
4. A vertex connected to both left and right subtrees.

A *full binary tree* is a binary tree in which every vertex has either no subtrees or else both a left and a right subtree. In effect, cases 2 and 3 are struck from the recursive definition of a binary tree given above.

Theorem 2.1 In every full binary tree, the number of leaves is one more than the number of internal nodes.

Proof. The proof is by strong induction. Let $P(n)$ be the predicate that in every full binary tree with n vertices, the number of leaves is one more than the number of internal nodes. In the base case, $P(1)$ is true because the full binary tree consisting of a single node has one leaf and zero internal nodes.

In the inductive step, for $n \geq 1$ assume that in every full binary tree with at most n vertices, the number of leaves is one more than the number of internal nodes. We will use this fact to prove that the number of leaves is one more than the number of internal nodes in every full binary tree with $n + 1$ vertices.

Let T be a full binary tree with $n + 1$ vertices. Since $n + 1 \geq 2$, T can not consist of a single vertex as in case 1 of the definition of a full binary tree. Therefore, T must consist of a vertex connected to both left and right subtrees as in case 4 of the definition. Furthermore, each subtree can contain at most n vertices, since the original graph contained $n+1$ vertices and neither subtree contains the root.

The number of leaves and internal nodes in T can be computed by adding the number of vertices of each type in the left and right subtrees and at the root. By induction, in both subtrees the number of leaves is one more than the number of internal nodes. However, the root of T is an internal node since it has two children. Adding these up, the number of internal nodes in T is exactly one less than the number of leaves. ■

There is something strange about this proof. The induction hypothesis is proven for all binary trees of size 1, 2, 3, etc. However, there are no full binary trees with an even number of vertices! This is a little bizarre, but not a bug in the proof. The induction hypothesis is just vacuously true for all even n. An an analogy, the statement "every flying pig is purple" is vacuously true for the same sort of reason: there are no flying pigs.

3 Directed Acyclic Graphs

Recall that in a directed graph, edges are intuitively regarded not as lines, but as arrows. There is one particularly important type of directed graph:

Definition A *directed acyclic graph* or *DAG* is a graph that does not contain a directed cycle.

(A *directed cycle* is just a cycle in which all edges are oriented the same way. A directed path is defined similarly.)

An example of a directed acyclic graph is shown in Figure 2. In the figure, each vertex is labeled with an MIT course number. An arc from vertex u to vertex v indicates that course u is a prerequisite for course v.

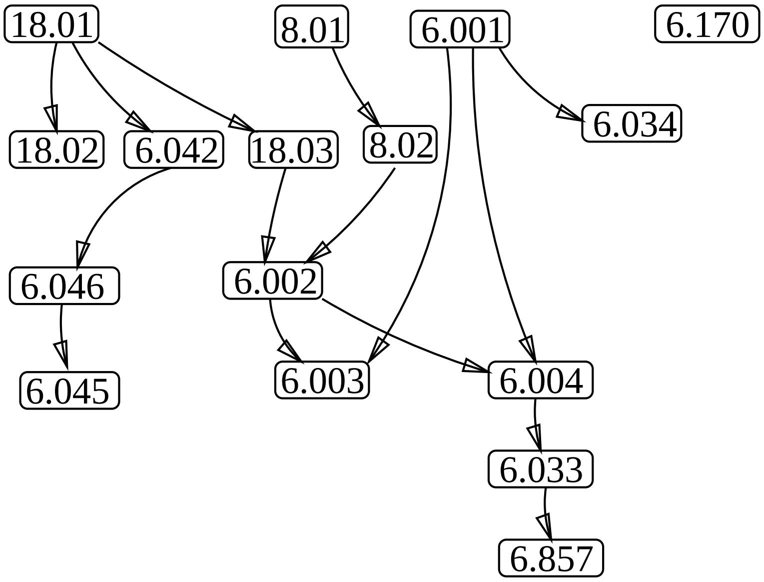

Figure 2: *This is a picture of a directed acyclic graph. Vertices are labeled with course names. An edge from u to v indicates that course u is a prerequisite for course v.*

Sometimes a directed acyclic graph is called a *partially ordered set*. The example in Figure 2 suggests the reason for this name. The arcs in this graph restrict the order in which a student may take classes. For example, 6.001 must be taken before 6.003. Of course, the student still has some choice about the order; for this reason, the set of classes is said to be "partially ordered" by this system of prerequisites. (Note that we could enforce the same partial ordering with a somewhat different directed acyclic graph; for example, adding an arc from 18.01 to 6.002 would not impose any new restrictions on the order in which courses are taken.)

Two vertices u and v in a partially ordered set are said to be *comparable* if there is a directed path from u to v or vice-versa. Otherwise, u and v are incomparable. For example, 18.01 and 6.002 are comparable since there is a directed path from 18.01 to 6.002. However, 18.01 and 8.01 are incomparable because there is no directed path from one to the other.

Problem How many terms are necessary to complete all of the courses in Figure 2, assuming prerequisites are observed, but course load is no issue?

The answer is suggested by the way Figure 2 is drawn. Each term the student could take all the courses in a horizontal row, starting with the first row and working down. All of the courses in each row are incomparable, and all the necessary prerequisites are in previous rows. Since there are six rows, the student can finish in six terms.

Furthermore, there is no way to finish in fewer than six terms, since each term the student can take at most one course in the six-course sequence 18.01, 18.03, 6.002, 6.004, 6.033, 6.857. Notice that this sequence constitutes the longest directed path in the graph.

Our observations about this example can be generalized to a theorem that holds for all directed acyclic graphs. First, some definitions:

Definition A *chain* in a partially ordered set is a sequence of vertices $a_1, a_2, \ldots, a_n$ such that there is a directed edge from a_1 to a_2, a directed edge from a_2 to a_3, ..., and a directed edge from a_{n-1} to a_n.

Definition An *antichain* in a partially ordered set is a set of vertices $\{b_1, b_2, \ldots, b_n\}$ such there is no directed path between any pair of distinct vertices in the set.

The longest chain in a DAG is called the *critical path*. In the prerequisite example, the critical path was that six-course sequence 18.01, 18.03, 6.002, 6.004, 6.033, 6.857. In any DAG representing a prerequisite structure, a student can complete at most one course on the critical path each term. Therefore, if the critical path contains t courses, then *at least* t terms are required. The following theorem says there actually is a way to finish all courses in *exactly* t terms.

Theorem 3.1 If the longest path in a DAG G has t vertices, then all the vertices in G can be partitioned into t antichains $A_1, A_1, \ldots, A_t$ such that for all $a \in A_i$, if there is an arc from b to a, then $b \in A_1 \cup A_2 \cup \ldots \cup A_{i-1}$.

This theorem implies that t terms are sufficient. In term i, the student should take the set of courses A_i. Since the sets $A_1, A_2, \ldots, A_t$ partition the set of all courses, the student will complete every course. Furthermore, the theorem ensures that for every course a, every prerequisite b was taken in a previous term.

Proof. Let A_i consist of all vertices a such that the longest path ending at a contains i vertices. This partitions the vertices of the graph into sets $A_1, A_2, \ldots, A_t$.

First, we have to prove that each set A_i is an antichain. The proof is by contradiction. Assume for the purpose of contradiction that A_i is not an antichain. Then there exist distinct vertices $a, b \in A_i$ such that there is a path from a to b. By definition of the set A_i, the longest path ending at a contains i vertices. If we append to this the path from a to b, then we obtain a path ending at b with more than i vertices. This contradicts the assertion that $b \in A_i$.

Second, we have to show that for all $a \in A_i$, if there is an arc from b to a, then $b \in A_1 \cup A_2 \cup \ldots \cup A_{i-1}$. Again, the proof is by contradiction. Assume for the purpose of contradiction that $b \notin A_1 \cup A_2 \cup \ldots \cup A_{i-1}$. Then the longest path ending at b must contain more than $i-1$ vertices. If we append the arc from b to a to this path, then we have constructed a path ending at a containing more than i vertices. This contradicts the assertion that $a \in A_i$. ■