5. Spanning Trees

5.1. Spanning Trees.

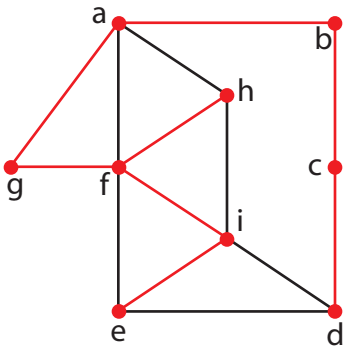
DEFINITION 5.1.1. *Given a connected graph G, a connected subgraph that is both a tree and contains all the vertices of G is called a* **spanning tree** *for G.*

Discussion

Given a connected graph G, one is often interested in constructing a connected subgraph, which contains all of the vertices of G, but has as few the edges of G as possible. For example, one might wish to find a minimal graph in a connected circuit in which each pair of nodes is connected. Such a subgraph will be a spanning tree for G. As we will soon see, if G is not already a tree, then it will have more than one spanning tree.

5.2. Example 5.2.1.

EXAMPLE 5.2.1. *In the figure below we have a graph drawn in red and black and a spanning tree of the graph in red.*



Discussion

The spanning tree of a graph is *not* unique.

EXERCISE 5.2.1. *Find at least two other spanning trees of the graph given in Example 5.2.1.*

5.3. Example 5.3.1.

EXAMPLE 5.3.1. *K_5 has 3 nonisomorphic spanning trees.*

Discussion

The three nonisomorphic spanning trees would have the following characteristics. One would have 3 vertices of degree 2 and 2 of degree 1, another spanning tree would have one vertex of degree three, and the third spanning tree would have one vertex of degree four. There *are* more than 3 spanning trees, but any other will be isomorphic to one of these three.

EXERCISE 5.3.1. *How many nonisomorphic (unrooted) spanning trees are there of the graph in Example 5.2.1?*

5.4. Existence.

THEOREM 5.4.1. *A graph is connected if and only if it has a spanning tree.*

Discussion

One of the key points of Theorem 5.4.1 is that any connected graph has a spanning tree. A spanning tree of a connected graph can be found by removing any edge that forms a simple circuit and continuing until no such edge exists. This algorithm turns out to be very inefficient, however, since it would be time-consuming to look for circuits each time we wish to consider removing an edge. There are two recursive algorithms, the depth-first search and the breadth-first search algorithms, that are fairly efficient, but we will not discuss them here, since they will be covered in depth in your computer science courses.

PROOF OF THEOREM 5.4.1. First we let G be a graph that contains a spanning tree, T. Let u and v be vertices in G. Since T is a spanning tree of G, u and v are also vertices in T. Now, T is a tree, so it is connected. Thus there is a path from u to v in T. T is, by definition of a spanning tree, a subgraph of G, so the path in T from u to v is also a path in G. Since u and v were arbitrary vertices in G we see that G is connected.

Conversely, we assume G is a connected graph. Notice that if G is not simple we may remove all loops and all but one edge between any pair of vertices that has more than one edge between them and the resulting graph will be a simple connected graph. Thus we may assume without loss of generality that G is simple.

Let $n = |V(G)|$. If G has fewer than $n-1$ edges then it would not be connected as a result of an exercise in *Connectivity*, thus $|E(G)| \geq n-1$. Also from *Introduction*

to Trees we recall that G is a tree if and only if $|E(G)| = n - 1$, so we assume G has at least $n - 1$ edges, say $|E(G)| = (n - 1) + k$.

We will prove that a connected graph with n vertices and $(n-1)+k$ edges contains a spanning tree by induction on k, $k \geq 0$.

Basis Step: $k = 0$. Then G is already a tree by our observation above.
Induction Step: Assume that every connected graph with n vertices and $(n - 1) + k$ edges, $k \geq 0$, contains a spanning tree. Suppose G is a connected graph with n vertices and $(n - 1) + (k + 1)$ edges. Since $|E(G)| > n - 1$, G is not a tree so there must be a simple circuit in G. Removing any edge from this circuit will result in a connected subgraph, G_1, of G with the same vertex set and $n - 1 + k$ edges. By our induction hypothesis, G_1 contains a spanning tree. But since G_1 is a subgraph of G containing all of the vertices of G, any spanning tree of G_1 is a spanning tree of G. Thus, G contains a spanning tree.

Therefore, by the principle of mathematical induction, every simple connected graph contains a spanning tree.

□

This proof can be used as a basis of a recursive algorithm for constructing spanning trees. It is, however, nothing more than the inefficient algorithm we alluded to above.

COROLLARY 5.4.1.1. *A connected subgraph of G that has the minimum number of edges and still contains all the vertices of G must be a spanning tree for G. Moreover, a spanning tree of a graph with n vertices must have exactly $n - 1$ edges.*

5.5. Spanning Forest.

DEFINITION 5.5.1. *A* **spanning forest** *of a graph G is the union of a collection of one spanning tree from each connected component of G.*

THEOREM 5.5.1. *Every finite simple graph has a spanning forest.*

Discussion

A graph that is not connected does not have a spanning tree. It does, however, have a spanning forest.

EXERCISE 5.5.1. *Prove Theorem 5.5.1.*

5.6. Distance.

DEFINITION 5.6.1. *Let T_1 and T_2 be spanning trees of the graph G. The* **distance**, *$d(T_1, T_2)$, between T_1 and T_2 is the number of edges in $E(T_1) \oplus E(T_2)$. That is,*

$$d(T_1, T_2) = |E(T_1) \oplus E(T_2)|.$$

Discussion

Recall the notation $\oplus$ from sets is the symmetric difference of two sets: $A \oplus B = A \cup B - A \cap B$.

Consider the spanning tree in Example 5.2.1. It is possible to find another spanning tree of the given graph by removing one edge from the given spanning tree and adding an edge not already in use. (See Exercise 5.6.4 below.) The distance between this new spanning tree and the old one is 2, since there is 1 edge in the first that is not in the second and one edge in the second that is not in the first.

EXERCISE 5.6.1. *What is the parity of the distance between any two spanning trees of a graph G? Explain. (Parity is even or odd).*

EXERCISE 5.6.2. *Prove that if A, B, and C are arbitrary finite sets, then*

$$|A \cap C| \geq |A \cap B| + |B \cap C| - |B|.$$

[Hint: If X and Y are finites sets, $|X \cup Y| = |X| + |Y| - |X \cap Y|$.]

EXERCISE 5.6.3. *This one is in your homework. Do not post the solution on the discussion board.*

Prove that if T_1, T_2, and T_3 are spanning trees of a simple connected graph G then

$$d(T_1, T_3) \leq d(T_1, T_2) + d(T_2, T_3).$$

[Hint: First, reduce the inequality to the one in Exercise 5.6.2.]

EXERCISE 5.6.4. *Suppose T and T' are spanning trees of a simple connected graph G and $T \neq T'$. Prove that there is an edge e in T that is not in T' and an edge e' in T' that is not in T such that if we remove e from T and then add e' to $T - e$, the resulting subgraph T'' of G is a spanning tree. [Hint: First show that $T \cup T'$ is a connected subgraph of G that is not a tree.]*

EXERCISE 5.6.5. *In Exercise 5.6.4 what is the relationship between $d(T', T'')$ and $d(T, T')$?*

EXERCISE 5.6.6. *Prove that if T and T' are spanning trees of a simple connected graph G and $T \neq T'$, then there are spanning trees $T_0, T_1, T_2, ..., T_k$ of G, $k \geq 1$, such that*

(a) $T_1 = T$,

(b) $T_k = T'$, *and*
(c) $d(T_{i-1}, T_i) = 2$, *for* $i = 1, ..., k$.

[Hint: Use induction on n, *where* $d(T, T') = 2n$, *and Exercise 5.6.4.]*

EXERCISE 5.6.7. *Prove that if* $T_0, T_1, T_2, ..., T_k$ *are are spanning trees of a simple connected graph* G, $k \geq 0$, *such that* $d(T_{i-1}, T_i) = 2$, *for* $i = 1, ..., k$, *then* $d(T_0, T_k) \leq 2k$.