Mathematical Foundations of Computer Science
Lecture Outline
October 15, 2024

Example. For a n-vertex graph G, the following are equivalent and characterize trees with n vertices.

(1) G is a tree.
(2) G is connected and has exactly $n-1$ edges.
(3) G is minimally connected, i.e., G is connected but $G - \{e\}$ is disconnected for every edge $e \in G$.
(4) G contains no cycle but $G + \{x, y\}$ does, for any two non-adjacent vertices $x, y \in G$.
(5) Any two vertices of G are linked by a unique path in G.

Solution. **$(1 \to 2)$.** We can prove this by induction on n. The property is clearly true for $n = 1$ as G has 0 edges. Assume that any tree with k vertices, for some $k \geq 1$, has $k-1$ edges. We want to prove that a tree G with $k+1$ vertices has k edges. From the example we did in last class we know that G has a leaf, say v, and that $G' = G - \{v\}$ is connected. By induction hypothesis, G' has $k-1$ edges. Since $deg(v) = 1$, G has k edges.

$(2 \to 3)$. Note that $G - \{e\}$ has n vertices and $n-2$ edges. We know that such a graph has at least 2 connected components and hence is disconnected.

$(3 \to 4)$. We are assuming that removing *any* edge in G disconnects G. If G contains a cycle then removing any edge, say $\{u, v\}$, that is part of the cycle does not disconnect G as any path that uses $\{u, v\}$ can now use the alternate route from u to v on the cycle. Since G is connected there is a path from x to y in G. Let $G' = G + \{x, y\}$. G' consists of a cycle formed by the edge $\{x, y\}$ and the path from x to y in G.

$(4 \to 5)$. Note that since $G + \{x, y\}$ creates a cycle for for any two non-adjacent vertices in G, it must be that there must be a path between x and y in G. We will now show that there is exactly one path between any two vertices in G. We will prove this by showing that if there are two vertices that have two different paths between them then G contains a cycle. Assume that there are two paths from u to v. Beginning at u, let a be the first vertex at which the two paths separate and let b be the first vertex after a where the two paths meet. Then, there are two simple paths from a to b with no common edges. Combining these two paths gives us a cycle.
$(5 \to 1)$. Since there is a path between any two vertices in G, G must be connected. Now we want to show that G is acyclic. Assume otherwise. Then, any two vertices on the cycle can reach each other by two disjoint, simple paths that consist of edges of the cycle. This proves that not every pair of vertices in G has a unique path between them. We have thus proved the claim by proving the contrapositive.

Spanning Trees

A *spanning subgraph* of a graph G is a subgraph with vertex set $V(G)$. A *spanning tree* is a spanning subgraph that is a tree.

Example. Every connected graph $G = (V, E)$ contains a spanning tree.

Solution. Let $T' = (V, E')$ be a minimally connected spanning subgraph of G. For a moment assume that such a T' always exists. Then, by the equivalence of statements (1) and (3), T' is a tree. Since T' is also a spanning subgraph of G, it is a spanning tree of G.

We now show that T', a minimally connected subgraph of G always exists. We will show this by actually constructing a minimally connected subgraph of G as follows. For each edge $e \in E$, remove e from E if its removal does not disconnect the graph. Let T' be the resulting subgraph obtained after each edge has been processed once. By construction and because G is connected, T' is connected. Also, by construction, no edge in T' can be removed without disconnecting T'. Hence, T' is minimally connected.

Rooted Trees

A *rooted tree* is a tree in which one vertex is distinguished from the others and is called the *root*. The *level* of a vertex, say u, is the number of edges along the unique path between u and the root. The *height* of a rooted tree is the maximum level of any vertex in the tree. Given any vertex of a rooted tree, the *children* of v are neighbors of v that are one level away from the root than v. If a vertex v is a child of u, then u is called the *parent* of v. Two vertices that are both children of the same parent are called *siblings*. Given vertices v and w, if v lies on the unique path between w and the root, then v is an *ancestor* of w and w is a *descendant* of v. A vertex in a rooted tree is called a *leaf* if it has no children. Vertices that have children are called *internal* vertices. The root is an internal vertex unless it is the only vertex in the graph, in which case it is a leaf. These definitions are illustrated in Figure 1. A *binary tree* is a rooted tree in which every internal vertex has at most two children. Each child in the binary tree is designated either a left child or a right child (but not both). A *full binary tree* is a binary tree in which each internal vertex has exactly two children.

Given an internal vertex v of a binary tree T, the left subtree of v is the binary tree whose root is the left child of v, whose vertices consists of the left child of v and all its descendants, and whose edges consist of all those edges of T that connect the vertices of the left subtree together. The right subtree of v is defined analogously.

Example. Prove the following. If k is a positive integer and T is a full binary tree with k internal vertices then T has a total of $2k + 1$ vertices and has $k + 1$ leaves.

Solution. Suppose T is a full binary tree with k internal vertices. Observe that the set of all vertices of T can be partitioned into two disjoint subsets: the set of all vertices in T that have a parent and the set of vertcies in T that do not have a parent. The root of T is the only vertex in T that does not have a parent. Also, every internal vertex of a full binary

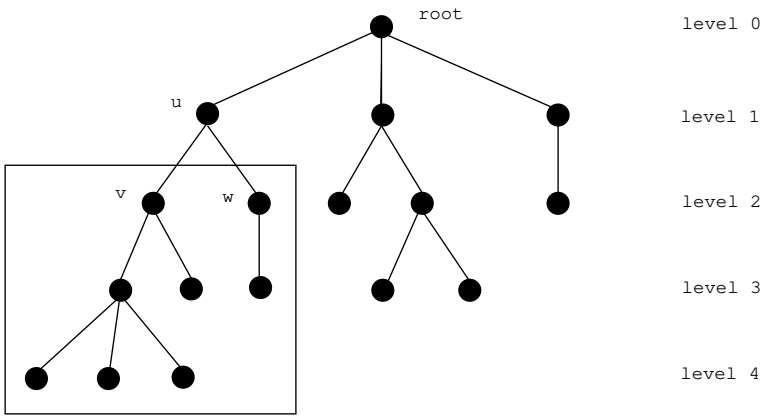

Figure 1: A rooted tree of height 4. In this tree v is a child of u, u is a parent of v, and v and w are siblings. All vertices in the marked portion of the tree descendants of u, which is an ancestor of each vertex.

tree has exactly two children. Thus, the total number of children of all internal vertices equals $2k$. This is also the number of vertices that have a parent. Adding one for the root to this number gives us the total number of vertices in T to be $2k+1$.

Also, the total number of vertices in T is the sum of the number of internal vertices in T and the number of leaves in T. Hence, the number of leaves in T equals $2k+1-k = k+1$.

Example. Prove that any binary tree of height at most h has at most 2^h leaves.

Solution. We will prove the claim by doing induction on h. Let $P(h)$ be the property that a binary tree of height at most h has at most 2^h leaves.

Induction Hypothesis: Assume that $P(k)$ is true for some $k \geq 0$.
Base Case: $P(0)$ is clearly true as there is only one tree of height at most zero. This tree has only one vertex which is a leaf. This equals $2^0 = 1$.
Induction Step: We want to prove that $P(k+1)$ is true. Consider any binary tree T having height at most $k+1$, and root r. Since we have already proven the claim for trees with height zero in the base case, we will assume that the height of T is at least one. The root r has at least one and at most two children. Each subtree rooted at a child of r is a rooted binary tree of height at most k. By induction hypothesis, the number of leaves in these subtrees is at most 2^k. Since r has at most two subtrees rooted at its children, the total number of leaves in T is at most $2 \times 2^k = 2^{k+1}$. This proves that $P(k+1)$ is true and hence completes the proof.