Basic Concepts and Properties of Trees

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong
This lecture provides a formal definition of trees, which constitute an important approach to organize data in computer science. We will also prove some basic properties of trees that will be useful later in the course.
An undirected simple graph is a pair of \((V, E)\) where:

- \(V\) is a set of elements, each of which called a node.
- \(E\) is a set of unordered pairs \(\{u, v\}\) such that \(u\) and \(v\) are distinct nodes.

A node may also be called a vertex. We will refer to \(V\) as the vertex set or the node set of the graph, and \(E\) the edge set.
This is a graph \((V, E)\) where

- \(V = \{a, b, c, d, e\}\)
- \(E = \{\{a, b\}, \{b, c\}, \{a, d\}, \{b, d\}, \{c, e\}\}\).

The number of edges equals \(|E| = 5\).
Let $G = (V, E)$ be an undirected simple graph. A path in $G$ is a sequence of nodes $(v_1, v_2, ..., v_k)$ such that

- For every $i \in [1, k - 1]$, there is an edge between $v_i$ and $v_{i+1}$.

A cycle in $G$ is a path $(v_1, v_2, ..., v_k)$ such that $k \geq 4$ and $v_1 = v_k$. 

Yufei Tao

Basic Concepts and Properties of Trees
Example

(a, b, d, a) is a cycle, whereas (a, b, c, e) is a path but not a cycle.
An undirected graph $G = (V, E)$ is connected if, for any two distinct vertices $u$ and $v$, $G$ has a path from $u$ to $v$. 
Example

Connected

Not connected
A tree is a connected undirected graph contains no cycles.
A Property

**Lemma:** A tree with $n$ nodes has $n - 1$ edges.

The proof will be left to you as an exercise.
Rooting a Tree

Given any tree $T$ and an arbitrary node $r$, we can allocate a level to each node as follows:

- $r$ is the root of $T$—this is level 0 of the tree.
- All the nodes that are 1 edge away from $r$ constitute level 1 of the tree.
- All the nodes that are 2 edges away from $r$ constitute level 2 of the tree.
- And so on.

The number of levels is called the height of the tree. We say that $T$ has been rooted once a root has been designated.
Example

Rooting the tree at $b$

Rooting the tree at $e$

Height 3

Height 4
Consider a tree $T$ that has been rooted.

Let $u$ and $v$ be two nodes in $T$. We say that $u$ is the parent of $v$ if:

- $v$ is at the level immediately below $u$, and
- There is an edge between $u$ and $v$.

Accordingly, we say that $v$ is a child of $u$. 
Example

Node $b$ is the parent of two child nodes: $a, d$. Node $e$ is the parent of $c$, which is in turn the parent of $b$. 
Consider a rooted tree $T$.

Let $u$ and $v$ be two nodes in $T$. We say that $u$ is an ancestor of $v$ if one of the following holds:

- $u = v$,
- $u$ is the parent of $v$, or
- $u$ is the parent of an ancestor of $v$.

Accordingly, we say that $v$ is a descendant of $u$.

In particular, if $u \neq v$, we say that $u$ is a proper ancestor of $v$, and likewise, $v$ is a proper descendant of $u$. 
Node $b$ is an ancestor of $b, a$ and $d$.
Node $c$ is an ancestor of $c, b, a$, and $d$.
Node $c$ is a proper ancestor of $b, a, d$. 
Concepts on Rooted Trees—Subtrees

Let $u$ be a node in a rooted tree $T$. The *subtree* of $u$ is the part of $T$ that is “at or below” $u$.

![Diagram of a rooted tree and its subtrees](image-url)

- **Tree**: The complete rooted tree with nodes $a$, $b$, $c$, $d$, and $e$.
- **Subtree of $c$**: Subtree containing nodes $a$, $b$, and $c$.
- **Subtree of $b$**: Subtree containing nodes $a$, $b$, $d$, and $e$. 

---

Yufei Tao | Basic Concepts and Properties of Trees
In a rooted tree, a node is a **leaf node** if it has no children; otherwise, it is an **internal node**.

Internal nodes: $e$, $c$, and $b$. Leaf nodes: $a$ and $d$. 
Lemma: Let $T$ be a rooted tree where every internal node has at least 2 child nodes. If $m$ is the number of leaf nodes, then the number of internal nodes is at most $m - 1$.

Proof: Consider the tree as a schedule for a tournament as follows. The competing teams are initially placed at the leaf nodes. Each internal node $v$ represents a match among the teams at the child nodes, such that only the winning team advances to the match at the parent of $v$. The team that wins the match at the root is the champion.

Each match eliminates at least one team. There are at most $m - 1$ teams to eliminate before the champion is determined. Hence, there can be at most $m - 1$ matches (i.e., nodes).
A \textit{k-ary tree} is a rooted tree where every internal node has at most \( k \) child nodes.

A 2-ary tree is called a \textit{binary tree}.

3-ary

Binary
A binary tree is **left-right labeled** if

- Every node $v$—except the root—has been designated either as a left or right node of its parent.
- Every internal node has at most one left child, and at most one right child.

Throughout this course, we will discuss only binary trees that have been left-right labeled. Because of this, by a “binary tree”, we always refer to a left-right labeled one.
A (left-right labeled) binary tree implies an ordering among the nodes at the same level.

Let $u, v$ be nodes at the same level with parents $p_u$ and $p_v$, respectively. We say that $u$ is on the left of $v$ if either of the following holds:

- $p_u = p_v$, and $u$ is the left child (implying that $v$ is the right child);
- $p_u \neq p_v$, and $p_u$ is on the left of $p_v$.

Accordingly, we say that $v$ is on the right of $u$. 
At Level 1, $b$ is on the left of $c$.
At Level 2, the nodes from left to right are $d$, $e$, and $f$.
At Level 3, the nodes from left to right are $g$, $h$, $i$, $j$, and $k$. 
Consider a binary tree with height $h$. Its Level $\ell$ ($0 \leq \ell \leq h - 1$) is full if it contains $2^\ell$ nodes.

Levels 0 and 1 are full, but Levels 2 and 3 are not.
A binary tree of height $h$ is complete if:

- Levels 0, 1, ..., $h - 2$ are all full (i.e., the only possible exception is the bottom level).
- At Level $h - 1$, the leaf nodes are “as far left as possible”.
  - This means that if you were to add a leaf node $v$ at Level $h - 1$, $v$ would need to be on the right of all the existing leaf nodes.
Example

Complete binary trees:

Not complete binary trees:
Lemma: A complete binary tree with \( n \geq 2 \) nodes has height \( O(\log n) \).

Proof: Let \( h \) be the height of the binary tree. As Levels 0, 1, ..., \( h - 2 \) are full, we know that

\[
2^0 + 2^1 + \ldots + 2^{h-2} \leq n \\
\Rightarrow 2^{h-1} - 1 \leq n \\
\Rightarrow \quad h \leq 1 + \log_2(n + 1) = O(\log n).
\]