Trees

COMS10017 - (Object-Oriented Programming and) Algorithms

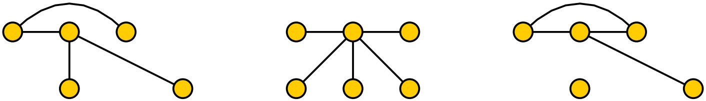
Dr Christian Konrad

Trees

Definition: A *tree* $T = (V, E)$ of size n is a tuple consisting of

$$V = \{v_1, v_2, \dots, v_n\} \text{ and } E = \{e_1, e_2, \dots, e_{n-1}\}$$

with $|V| = n$ and $|E| = n - 1$ with $e_i = \{v_j, v_k\}$ for some $j \neq k$ s.t. for every pair of vertices v_i, v_j $(i \neq j)$, there is a path from v_i to v_j. V are the nodes/vertices and E are the edges of T.

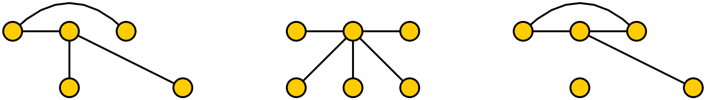


Trees

Definition: A *tree* $T = (V, E)$ of size n is a tuple consisting of

$$V = \{v_1, v_2, \dots, v_n\} \text{ and } E = \{e_1, e_2, \dots, e_{n-1}\}$$

with $|V| = n$ and $|E| = n - 1$ with $e_i = \{v_j, v_k\}$ for some $j \neq k$ s.t. for every pair of vertices v_i, v_j $(i \neq j)$, there is a path from v_i to v_j. V are the nodes/vertices and E are the edges of T.



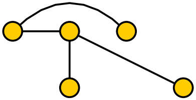
$\checkmark$

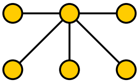
Trees

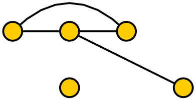
Definition: A *tree* $T = (V, E)$ of size n is a tuple consisting of

$$V = \{v_1, v_2, \ldots, v_n\} \text{ and } E = \{e_1, e_2, \ldots, e_{n-1}\}$$

with $|V| = n$ and $|E| = n - 1$ with $e_i = \{v_j, v_k\}$ for some $j \neq k$ s.t. for every pair of vertices v_i, v_j $(i \neq j)$, there is a path from v_i to v_j. V are the nodes/vertices and E are the edges of T.





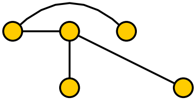


✓ ✓

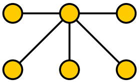
Definition: A *tree* $T = (V, E)$ of size n is a tuple consisting of

$$V = \{v_1, v_2, \ldots, v_n\} \text{ and } E = \{e_1, e_2, \ldots, e_{n-1}\}$$

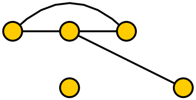
with $|V| = n$ and $|E| = n - 1$ with $e_i = \{v_j, v_k\}$ for some $j \neq k$ s.t. for every pair of vertices v_i, v_j $(i \neq j)$, there is a path from v_i to v_j. V are the nodes/vertices and E are the edges of T.



✓



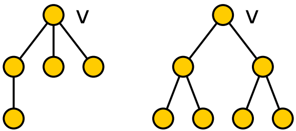
✓



✗

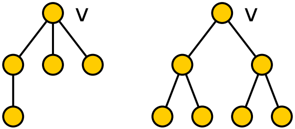
Rooted Trees

Definition: (rooted tree) A *rooted* tree is a triple $T = (v, V, E)$ such that $T = (V, E)$ is a tree and $v \in V$ is a designated node that we call the *root* of T.



Rooted Trees

Definition: (rooted tree) A *rooted* tree is a triple $T = (v, V, E)$ such that $T = (V, E)$ is a tree and $v \in V$ is a designated node that we call the *root* of T.



Definition: (leaf, internal node) A *leaf* in a tree is a node with exactly one incident edge. A node that is not a leaf is called an *internal node*.

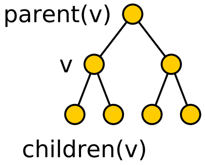
Children, Parent, and Degree

Further Definitions:

Children, Parent, and Degree

Further Definitions:

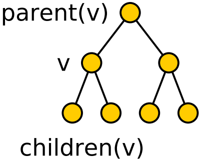
- The *parent* of a node v is the closest node on a path from v to the root. The root does not have a parent.



Children, Parent, and Degree

Further Definitions:

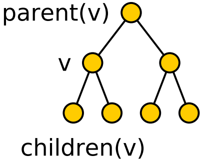
- The *parent* of a node v is the closest node on a path from v to the root. The root does not have a parent.
- The *children* of a node v are v's neighbours except its parent.



Children, Parent, and Degree

Further Definitions:

- The *parent* of a node v is the closest node on a path from v to the root. The root does not have a parent.
- The *children* of a node v are v's neighbours except its parent.

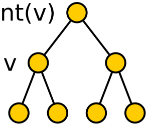


- The *height* of a tree is the length of a longest root-to-leaf path.

Children, Parent, and Degree

Further Definitions:

- The *parent* of a node v is the closest node on a path from v to the root. The root does not have a parent.
- The *children* of a node v are v's neighbours except its parent.

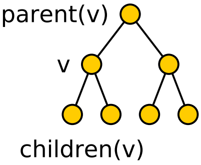


- The *height* of a tree is the length of a longest root-to-leaf path.
- The *degree* $\deg(v)$ of a node v is the number of incident edges to v. Since every edge is incident to two vertices we have

$$\sum_{v \in V} \deg(v) = 2 \cdot |E| = 2(n-1) \ .$$

Children, Parent, and Degree

Further Definitions:



- The *parent* of a node v is the closest node on a path from v to the root. The root does not have a parent.
- The *children* of a node v are v's neighbours except its parent.
- The *height* of a tree is the length of a longest root-to-leaf path.
- The *degree* $\deg(v)$ of a node v is the number of incident edges to v. Since every edge is incident to two vertices we have

$$\sum_{v \in V} \deg(v) = 2 \cdot |E| = 2(n-1) \ .$$

- The *level* of a vertex v is the length of the unique path from the root to v plus 1.

Properties of Trees

Property:

Properties of Trees

Property: Every tree has at least 2 leaves

Property: Every tree has at least 2 leaves

Proof

Properties of Trees

Property: Every tree has at least 2 leaves

Proof Let $L \subseteq V$ be the subset of leaves.

Properties of Trees

Property: Every tree has at least 2 leaves

Proof Let $L \subseteq V$ be the subset of leaves. Suppose that there is at most 1 leaf, i.e., $|L| \leq 1$.

Properties of Trees

Property: Every tree has at least 2 leaves

Proof Let $L \subseteq V$ be the subset of leaves. Suppose that there is at most 1 leaf, i.e., $|L| \leq 1$. Then:

Properties of Trees

Property: Every tree has at least 2 leaves

Proof Let $L \subseteq V$ be the subset of leaves. Suppose that there is at most 1 leaf, i.e., $|L| \leq 1$. Then:

$$\sum_{v \in V} \deg(v)$$

Properties of Trees

Property: Every tree has at least 2 leaves

Proof Let $L \subseteq V$ be the subset of leaves. Suppose that there is at most 1 leaf, i.e., $|L| \leq 1$. Then:

$$\sum_{v \in V} \deg(v) \;=\; \sum_{v \in L} \deg(v) + \sum_{v \in V \setminus L} \deg(v)$$

Properties of Trees

Property: Every tree has at least 2 leaves

Proof Let $L \subseteq V$ be the subset of leaves. Suppose that there is at most 1 leaf, i.e., $|L| \leq 1$. Then:

$$\begin{aligned} \sum_{v \in V} \deg(v) &= \sum_{v \in L} \deg(v) + \sum_{v \in V \setminus L} \deg(v) \\ &\geq |L| \cdot 1 + (|V| - |L|) \cdot 2 \end{aligned}$$

Properties of Trees

Property: Every tree has at least 2 leaves

Proof Let $L \subseteq V$ be the subset of leaves. Suppose that there is at most 1 leaf, i.e., $|L| \leq 1$. Then:

$$\begin{aligned} \sum_{v \in V} \deg(v) &= \sum_{v \in L} \deg(v) + \sum_{v \in V \setminus L} \deg(v) \\ &\geq |L| \cdot 1 + (|V| - |L|) \cdot 2 = 2|V| - |L| \geq \end{aligned}$$

Properties of Trees

Property: Every tree has at least 2 leaves

Proof Let $L \subseteq V$ be the subset of leaves. Suppose that there is at most 1 leaf, i.e., $|L| \leq 1$. Then:

$$\begin{aligned}\sum_{v \in V} \deg(v) &= \sum_{v \in L} \deg(v) + \sum_{v \in V \setminus L} \deg(v) \\ &\geq |L| \cdot 1 + (|V| - |L|) \cdot 2 = 2|V| - |L| \geq 2n - 1 \,,\end{aligned}$$

Properties of Trees

Property: Every tree has at least 2 leaves

Proof Let $L \subseteq V$ be the subset of leaves. Suppose that there is at most 1 leaf, i.e., $|L| \leq 1$. Then:

$$\begin{aligned}\sum_{v \in V} \deg(v) &= \sum_{v \in L} \deg(v) + \sum_{v \in V \setminus L} \deg(v) \\ &\geq |L| \cdot 1 + (|V| - |L|) \cdot 2 = 2|V| - |L| \geq 2n - 1 \ ,\end{aligned}$$

a contradiction to the fact that $\sum_{v \in V} \deg(v) = 2(n-1)$ in every tree. $\square$

Binary Trees

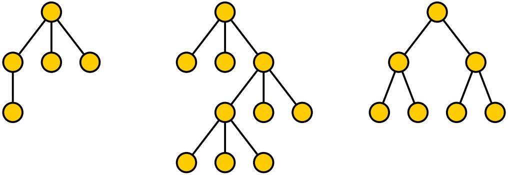
Definition: (k-ary tree) A (rooted) tree is *k-ary* if every node has at most k children. If $k = 2$ then the tree is called binary. A k ary tree is

- *full* if every internal node has exactly k children,
- *complete* if all levels except possibly the last is entirely filled (and last level is filled from left to right),
- *perfect* if all levels are entirely filled.

Binary Trees

Definition: (k-ary tree) A (rooted) tree is *k-ary* if every node has at most k children. If $k = 2$ then the tree is called binary. A k ary tree is

- *full* if every internal node has exactly k children,
- *complete* if all levels except possibly the last is entirely filled (and last level is filled from left to right),
- *perfect* if all levels are entirely filled.



complete 3-ary tree full 3-ary tree perfect binary tree

Height of Perfect and Complete k-ary Trees

Height of k-ary Trees

Height of k-ary Trees

- The number of nodes in a perfect k-ary tree of height $i-1$ is

$$\sum_{j=0}^{i-1} k^j = \frac{k^i - 1}{k - 1} .$$

Height of Perfect and Complete k-ary Trees

Height of k-ary Trees

- The number of nodes in a perfect k-ary tree of height $i-1$ is

$$\sum_{j=0}^{i-1} k^j = \frac{k^i - 1}{k-1} \ .$$

- In other words, a perfect k-ary tree on n nodes has height:

$$\begin{aligned} n &= \frac{k^i - 1}{k-1} \\ k^i &= n(k-1) + 1 \\ i &= \log_k(n(k-1)+1) = O(\log_k n) \ . \end{aligned}$$

Height of Perfect and Complete k-ary Trees

Height of k-ary Trees

- The number of nodes in a perfect k-ary tree of height $i-1$ is

$$\sum_{j=0}^{i-1} k^j = \frac{k^i - 1}{k-1} \ .$$

- In other words, a perfect k-ary tree on n nodes has height:

$$\begin{aligned} n &= \frac{k^i - 1}{k-1} \\ k^i &= n(k-1) + 1 \\ i &= \log_k(n(k-1)+1) = O(\log_k n) \ . \end{aligned}$$

- Similarly, a complete k-ary tree has height $O(\log_k n)$.

Height of Perfect and Complete k-ary Trees

Height of k-ary Trees

- The number of nodes in a perfect k-ary tree of height $i-1$ is

$$\sum_{j=0}^{i-1} k^j = \frac{k^i - 1}{k-1} \ .$$

- In other words, a perfect k-ary tree on n nodes has height:

$$\begin{aligned} n &= \frac{k^i - 1}{k-1} \\ k^i &= n(k-1)+1 \\ i &= \log_k(n(k-1)+1) = O(\log_k n) \ . \end{aligned}$$

- Similarly, a complete k-ary tree has height $O(\log_k n)$.

Remark: The runtime of many algorithms that use tree data structures depends on the height of these trees. We are therefore interested in using complete/perfect trees.