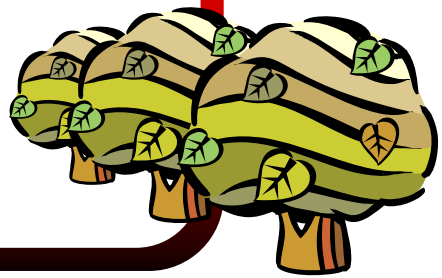
# Trees

## Today's Topics

- Trees
  - Rooted Trees
  - *m*-ary Trees
- Properties of Trees
- Spanning Trees
- Binary Search Trees
- Tree Traversal

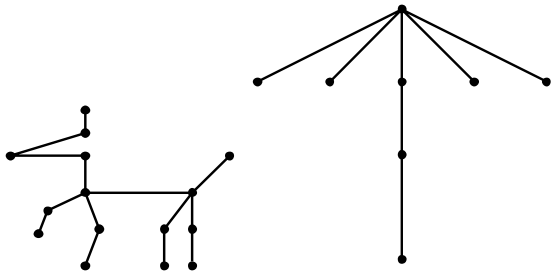




# Trees

*Definition:*

A **tree** is a connected graph with no simple circuits






# Trees

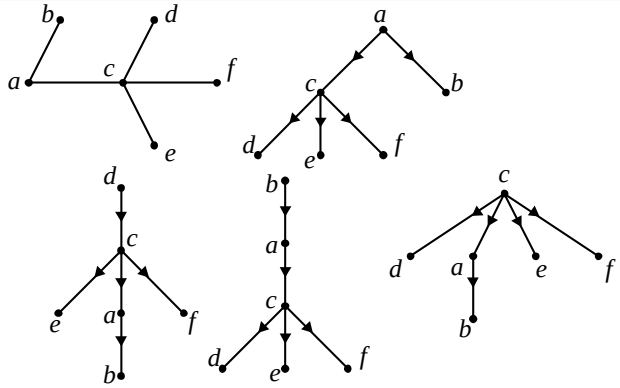
An undirected graph is a tree ↔ there is a unique path between any two of its vertices.



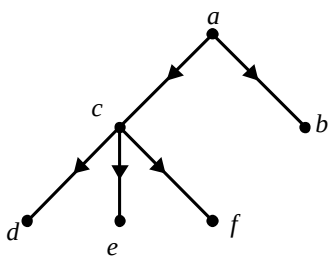
# Rooted Trees

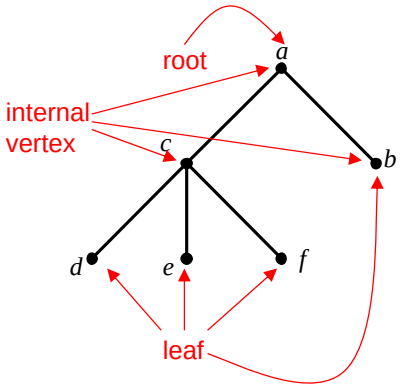
*Definition:*

A **rooted tree** is tree in which one vertex has been designated as the root and every edge is directed away from the root.



# Rooted Trees



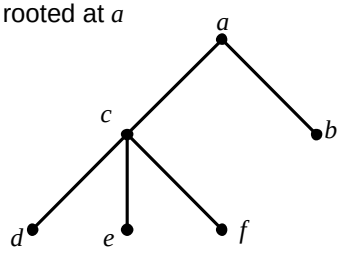

parent, child, siblings

ancestor, descendant



# Subtrees

rooted at *a*



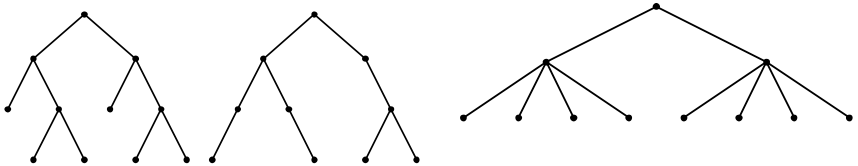

The **subtree** with *v* as its root is the subgraph of the tree consisting of *v* and its descendants and all edges incident to these descendants.



# *m*–ary Trees

*Definition:*

An *m*-ary tree is a **rooted tree** whose every internal vertex has no more than *m* children



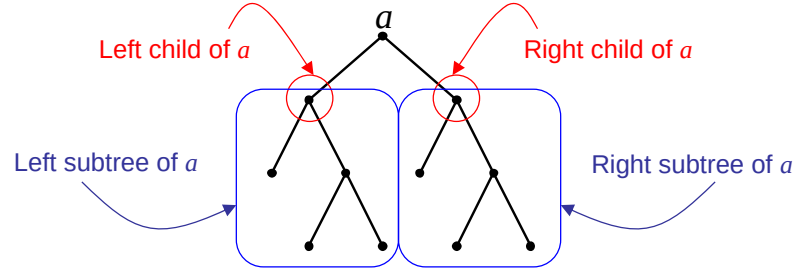
When every internal vertex has exactly *m* children → Full *m*-ary Tree



# Ordered Rooted Trees

Siblings are ordered from left to right

Ordered binary tree (Binary tree)



# Properties of Trees

A tree with $n$ vertices has $n$-1 edges.

A full $m$-ary tree with $i$ internal vertices contains $n = mi+1$ vertices.

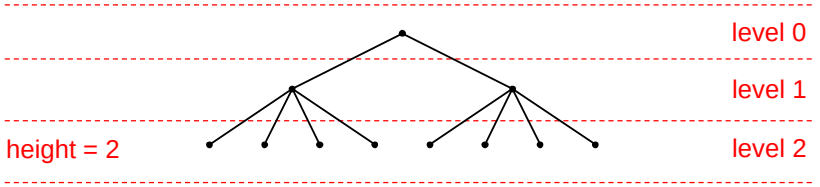
A full 4-ary tree has 13 vertices. How many leaves are there?



# Level and Height

The **level** of a vertex in a rooted tree is the length of the unique path form the root to this vertex.

**Height** = maximum of the levels



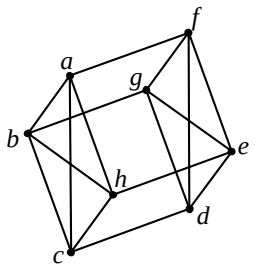

There are at most $m^h$ leaves in an $m$-ary tree of height $h$.



# Spanning Trees

*Definition:*

Let $G$ be a simple graph. A **spanning tree** of $G$ is a subgraph of $G$ that is a tree containing every vertex of $G$.






# Spanning Trees

A simple graph is connected ↔ It has a spanning tree

# Binary Search Trees

- Each vertex is labeled with a **key**.
- The key of a vertex is:
  - larger than the keys of all vertices in its left subtree.
  - smaller than the keys of all vertices in its right subtree.



# Forming a Binary Search Tree



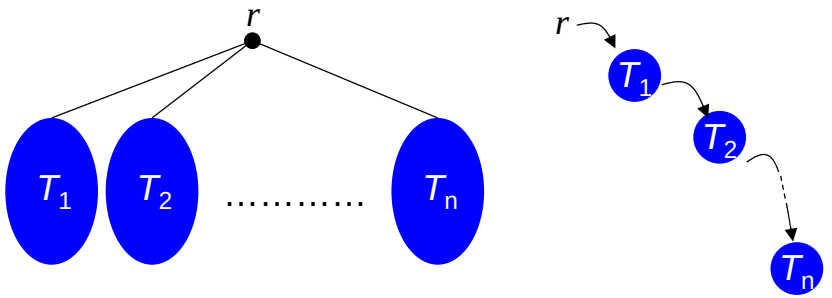
# Locating / Adding Items



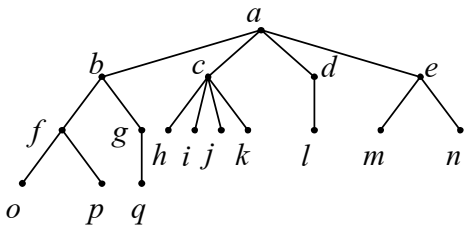
# Tree Traversal

**Pre-order Traversal**

Visit $r$, then pre-orderly traverse each subtree whose root is a child of $r$, from left to right.



# Pre-order Traversal

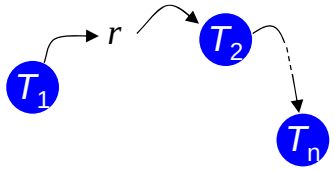





# In-order Traversal

## In-order Traversal

In-orderly traverse the subtree whose root is the leftmost child of $r$, visit $r$, then in-orderly traverse each subtree (excluding the leftmost one) whose root is a child of $r$, from left to right.






# Post-order Traversal

## Post-order Traversal

Post-orderly traverse each subtree whose root is a child of $r$, from left to right, then visit $r$.