Ch 11.1 Trees

- Particular type of graph – Tree
- Trees resemble graphs
- Applications

  Data structures

  Searching

  Compilers

  Databases

  Routing

Trees

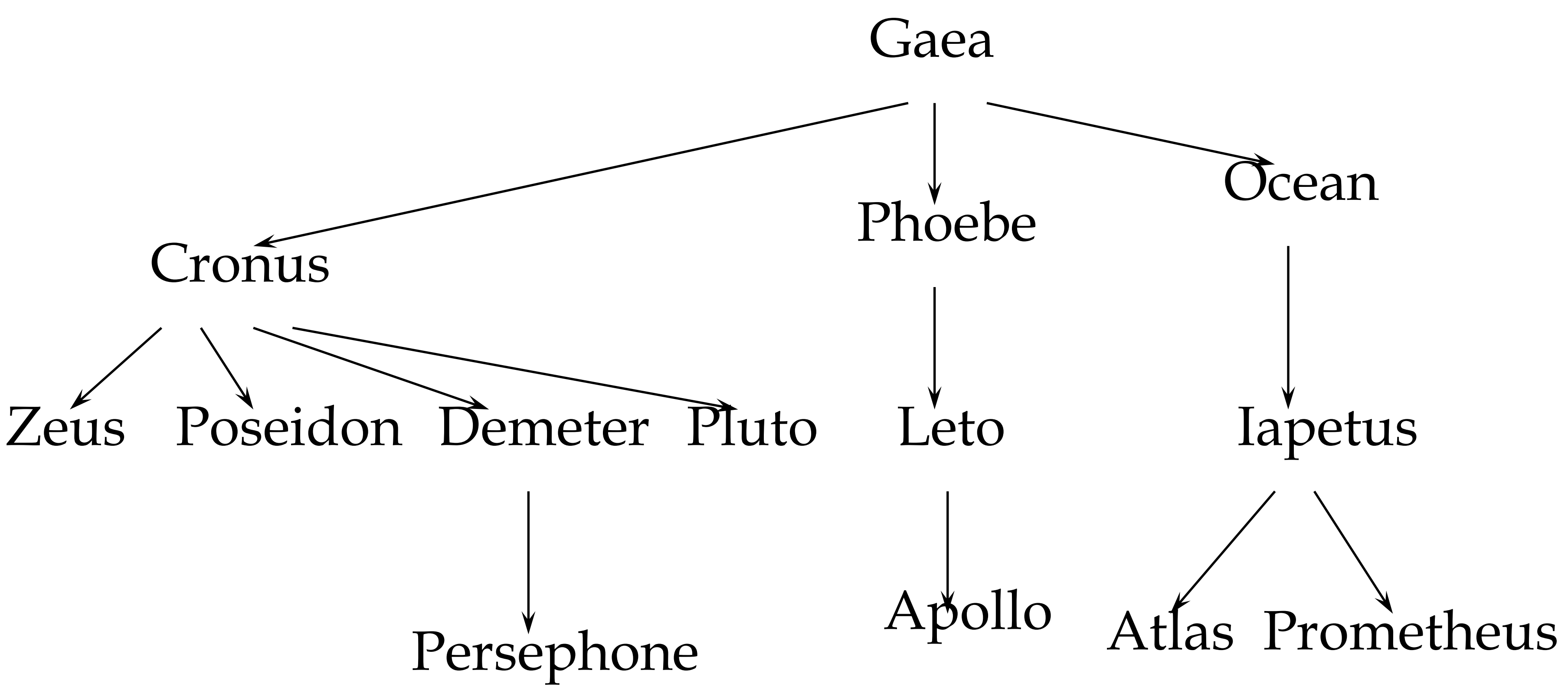
- Family Trees: Graphs that represent genealogical charts

Vertices - represent the members of a family

Edges - represent parent-child relationships

Much of the tree terminology derives from family trees.

Trees


Gaea
Cronus
Phoebe
Ocean
Zeus
Poseidon
Demeter
Pluto
Leto
Iapetus
Persephone
Apollo
Atlas
Prometheus

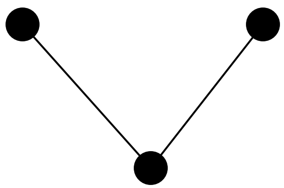
Trees

- Definition:

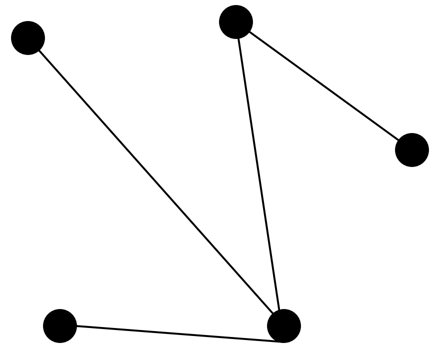
A *tree* is a connected undirected graph with no simple circuits

- A tree cannot contain multiple edges or loops
- A tree must be a simple graph

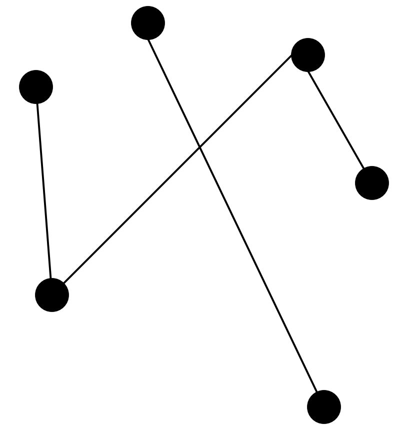
Trees



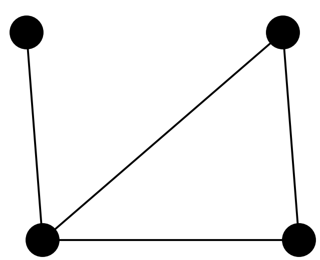
Tree



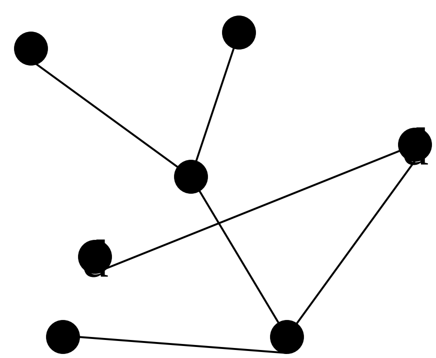
Tree



Not a Tree – not connected

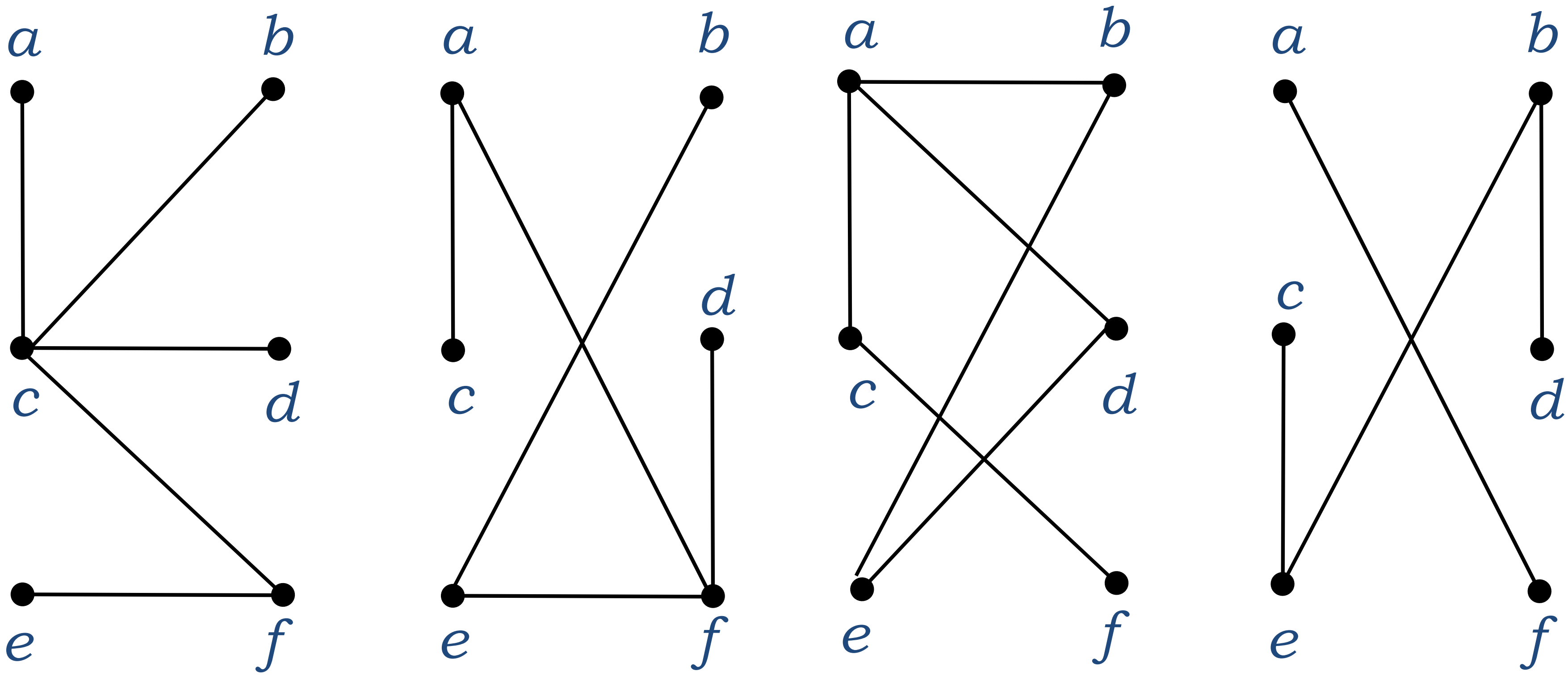


Not a Tree – simple circuit



Tree

Which graphs are trees?

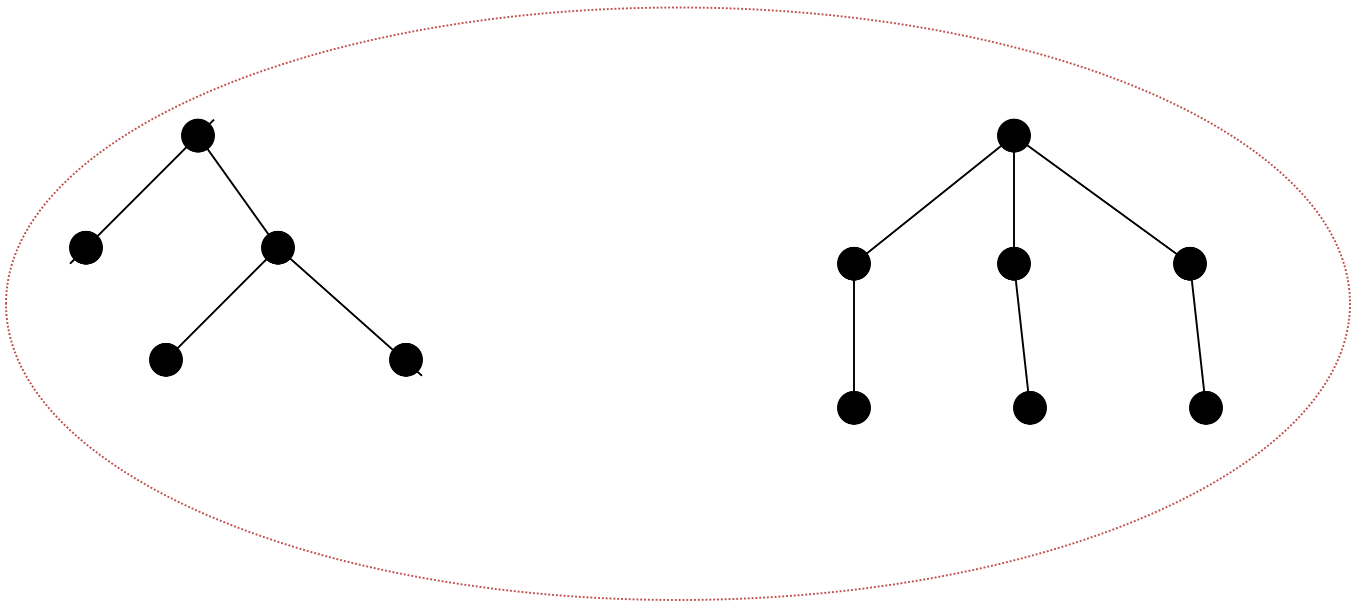


YES YES NO NO

Trees

- Forest

- Graphs containing no simple circuits that are not connected, but each connected component is a tree.

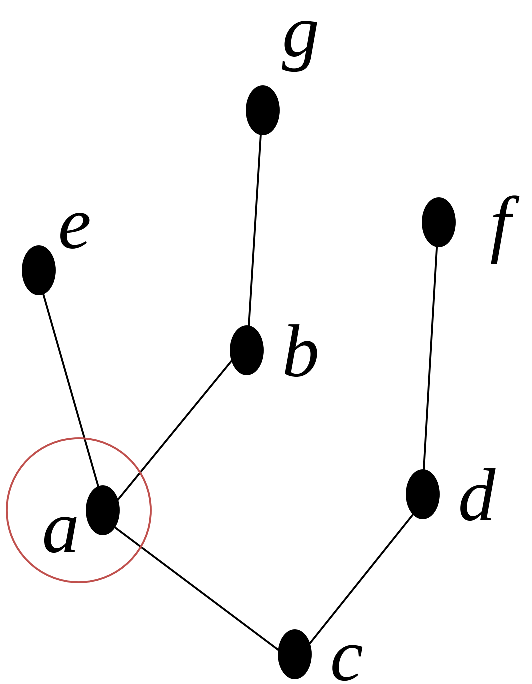


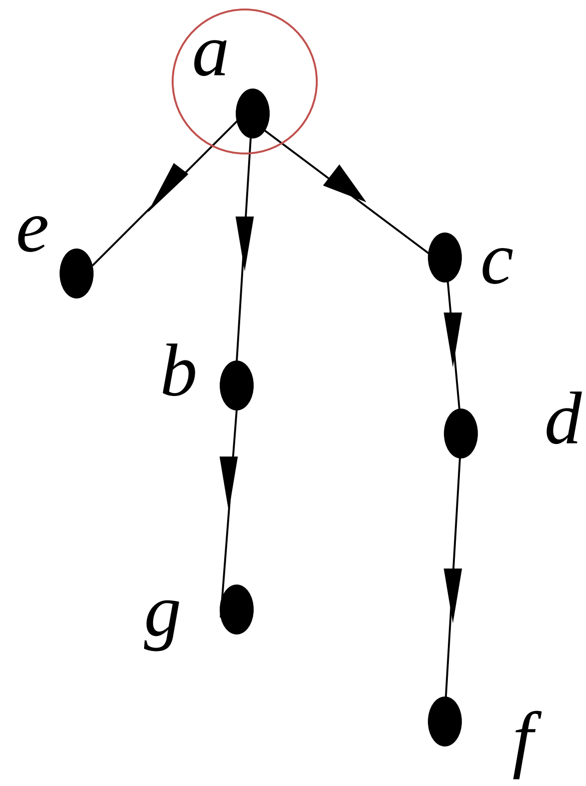
Trees

- Rooted Trees
- A particular vertex of a tree – Root
- Assign a direction to each edge, direct each edge away from the root

Definition: One vertex has been designated as the root and every edge is directed away from the root

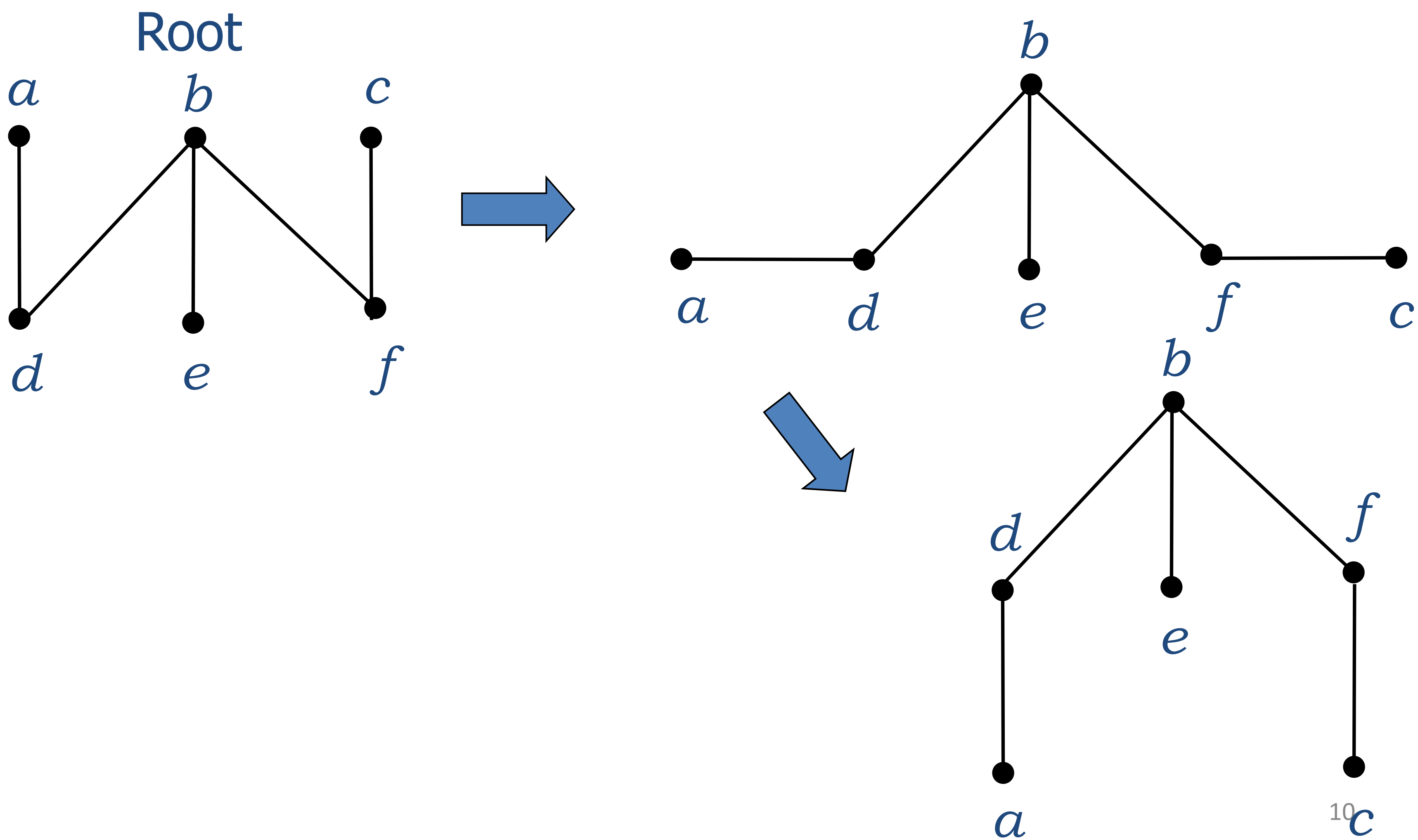
Trees



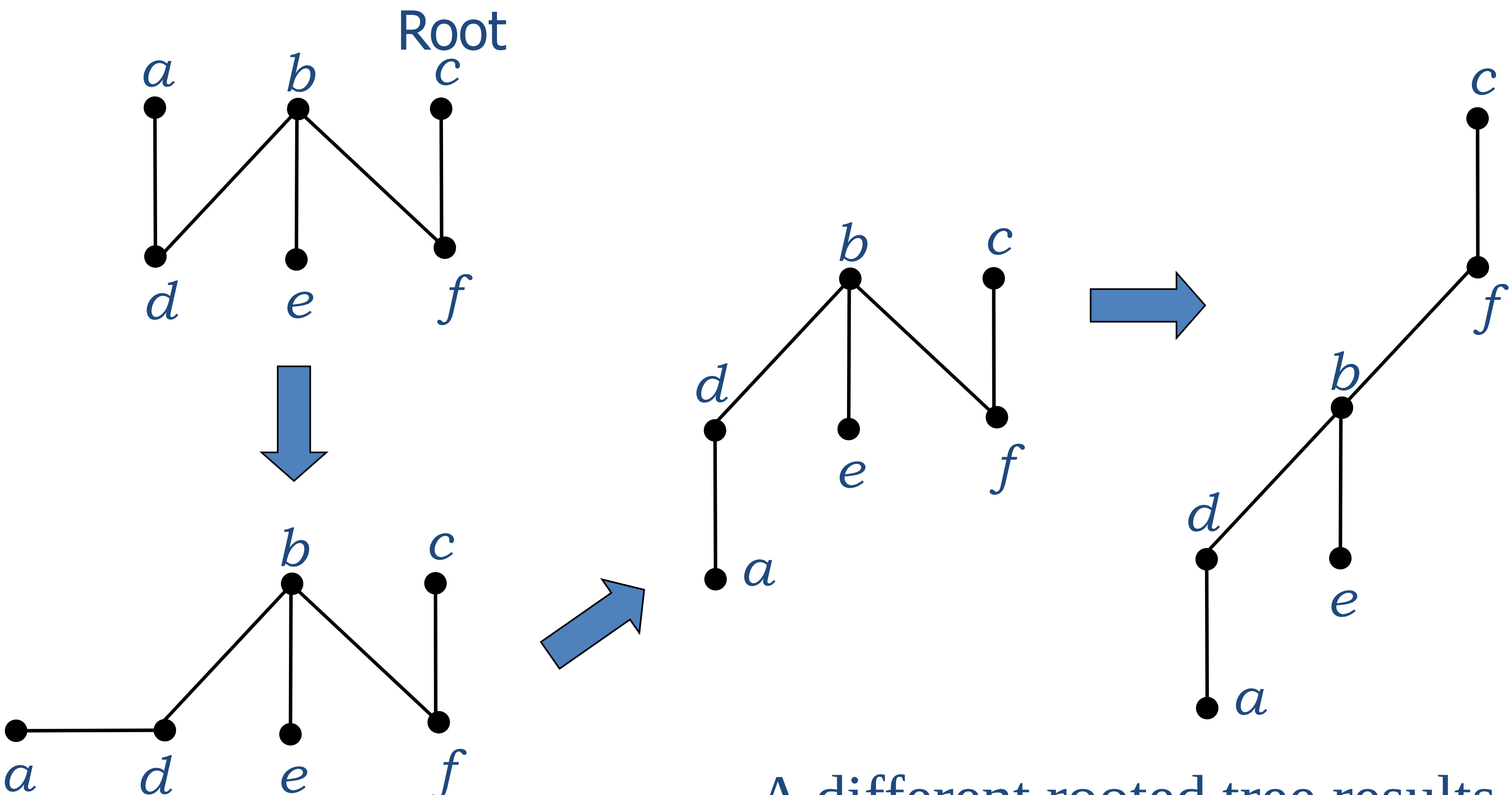


Rooted Trees

Example



What if a different root is chosen?



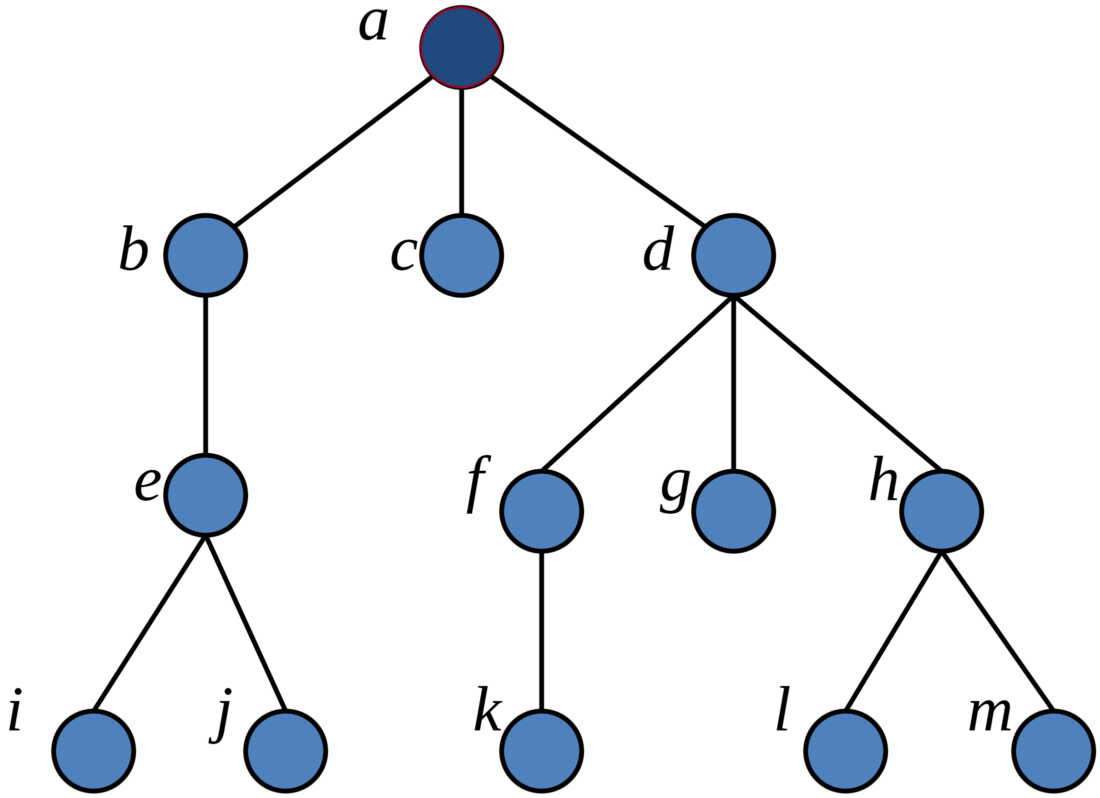
A different rooted tree results.

Tree Terminology

- If v is a vertex of tree T other than the root, the *parent* of v is the unique vertex u such that there is a directed edge from u to v.
- When u is the parent of v, v is called the *child* of u.
- If two vertices share the same parent, then they are called *siblings*.

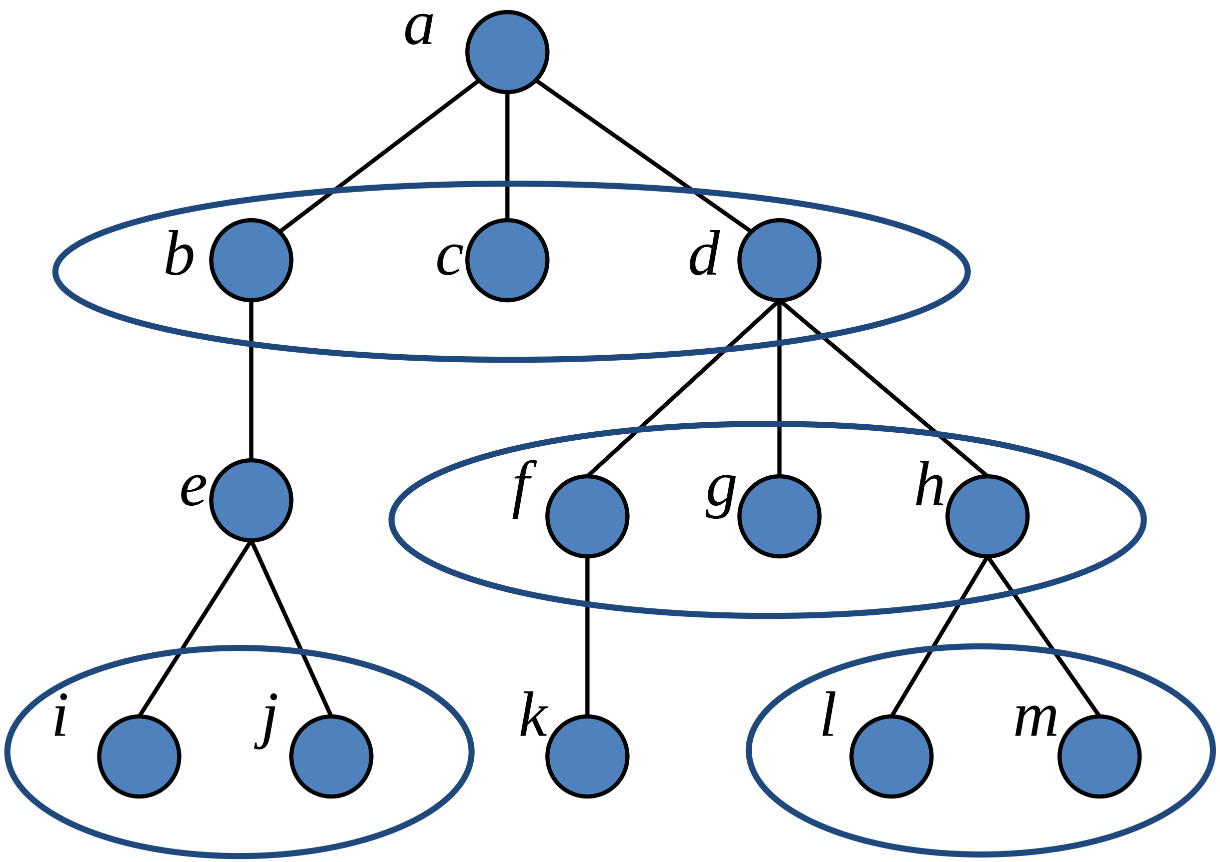
Example

Root



Example

Siblings

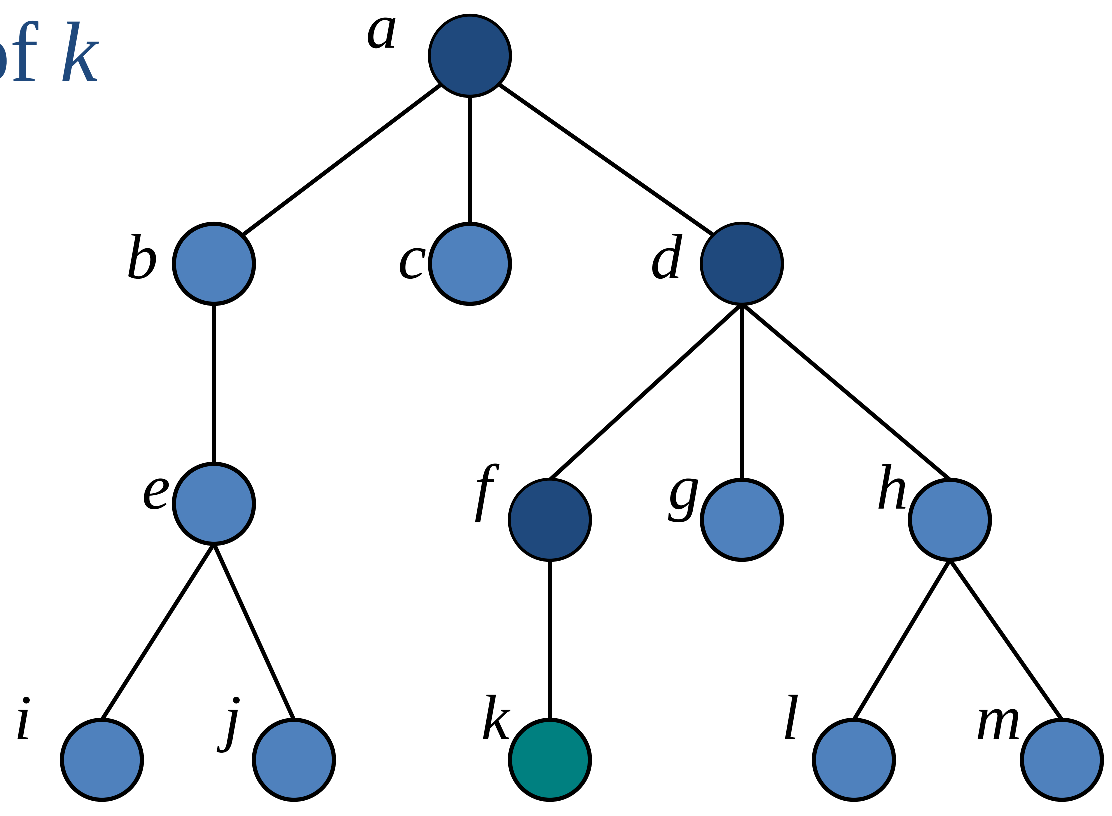


Tree Terminology (Cont.)

- The *ancestors* of a vertex other than the root are the vertices in the path from the root to this vertex, excluding the vertex itself and including the root.
- The *descendants* of a vertex v are those vertices that have v as an ancestor.

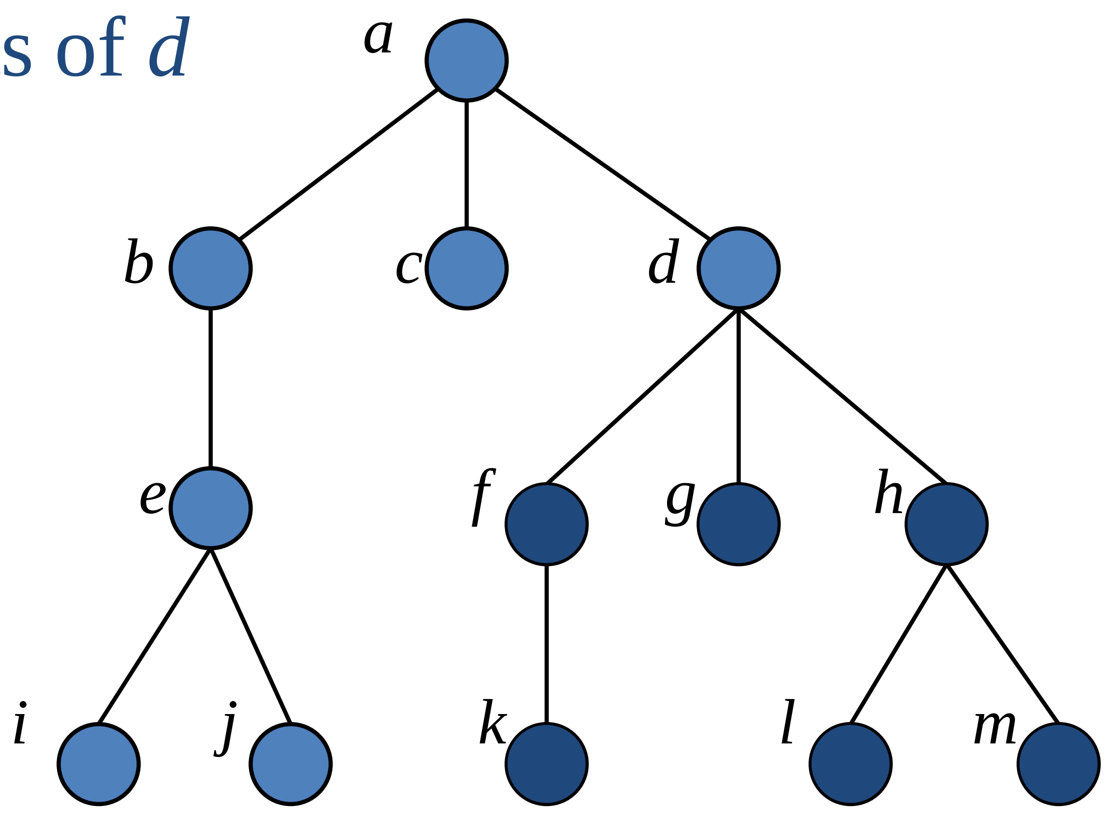
Example

Ancestors of *k*



Example

Descendants of *d*

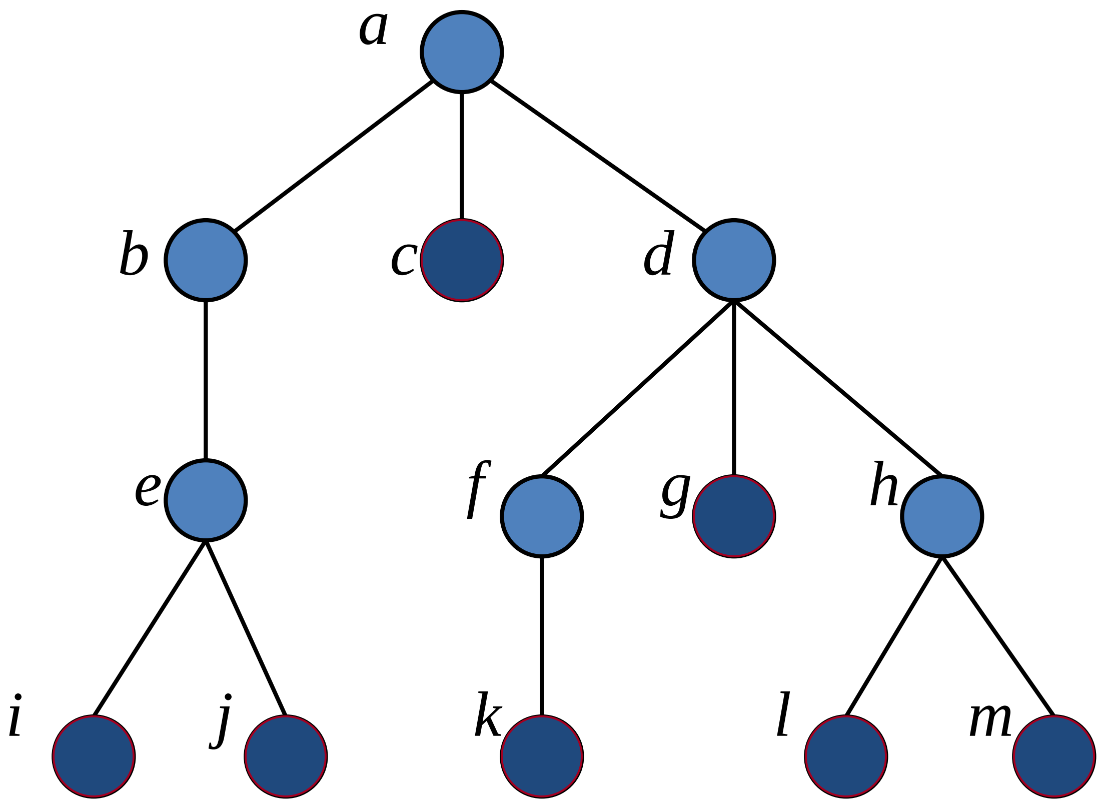


Tree Terminology (Cont.)

- A vertex with no children is called a *leaf*.
- Vertices with children are called *internal vertices*.

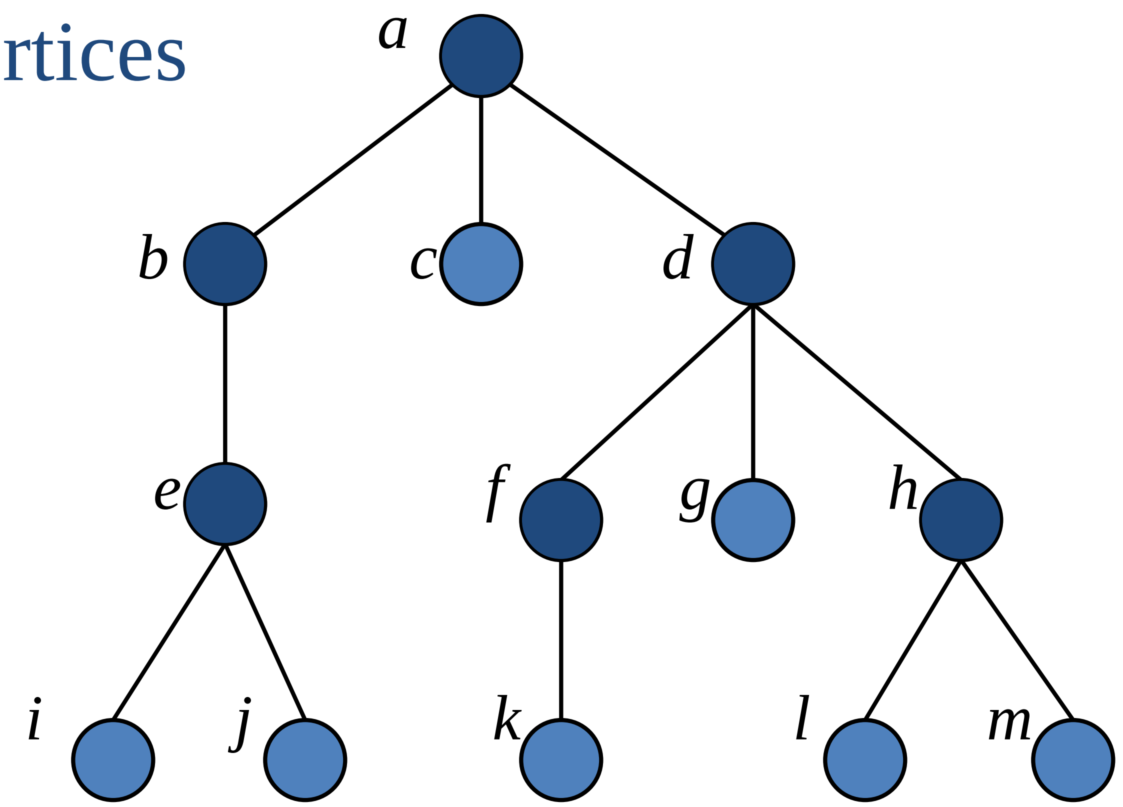
Example

Leaves



Example

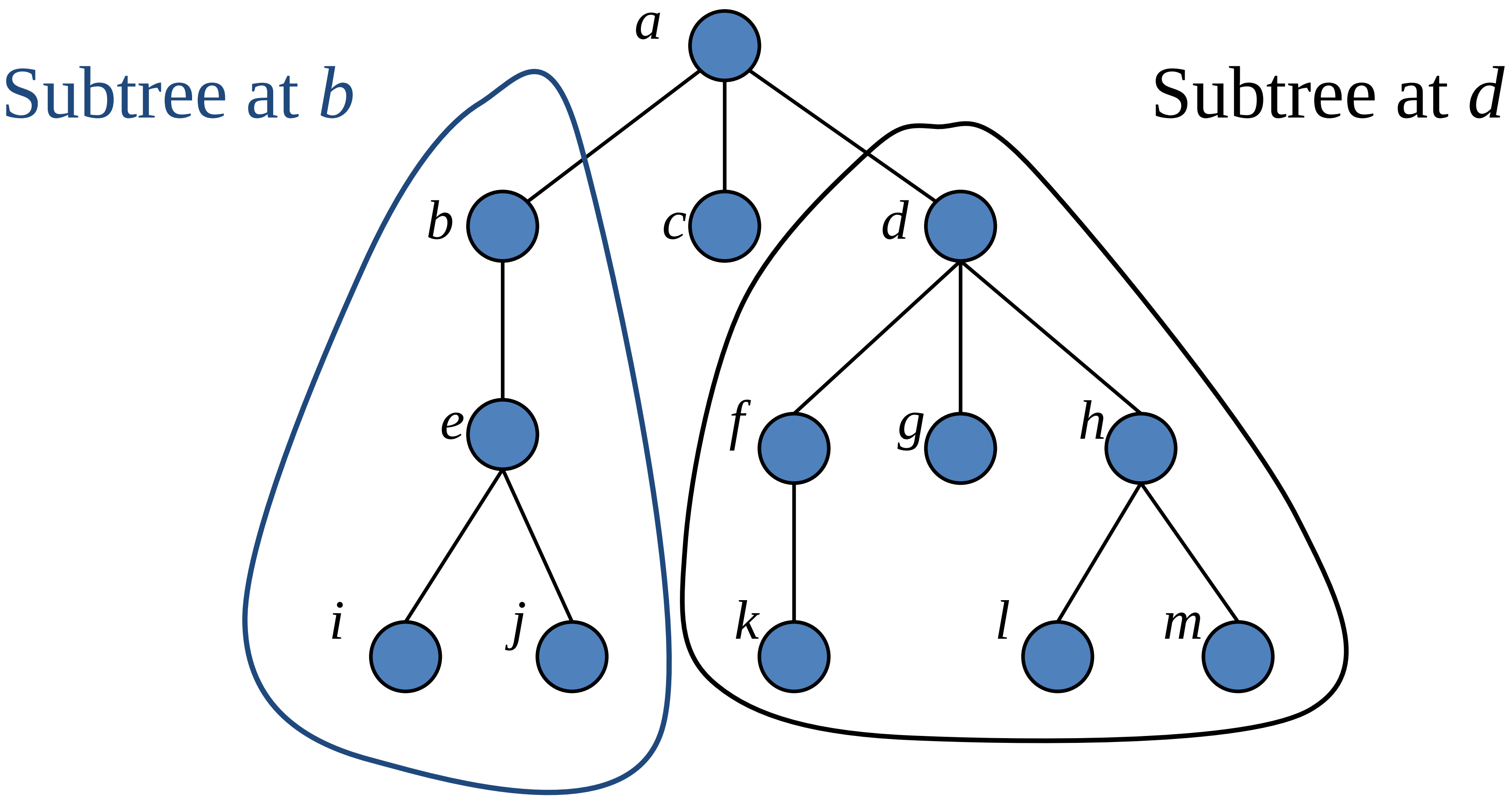
Internal vertices



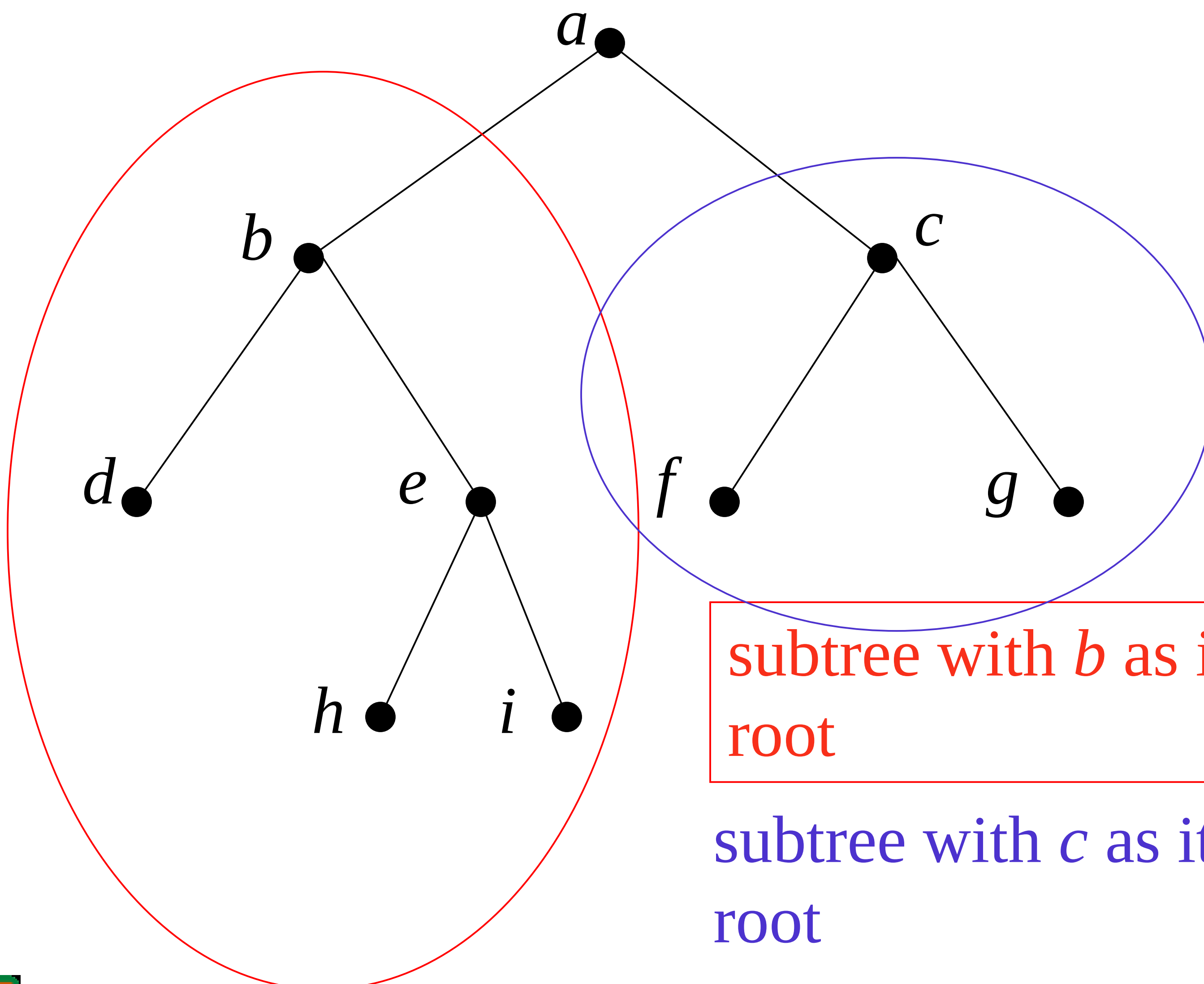
Tree Terminology (Cont.)

- If *a* is a vertex in a tree, the *subtree* with *a* as its root is:
  - the subgraph of the tree consisting of *a* and its descendants, and
  - all edges incident to these descendants.

Example



Trees



subtree with *b* as its root

subtree with *c* as its root

Trees

m-ary trees

A rooted tree is called an *m-ary tree*

- if every internal vertex has no more than *m* children

-The tree is called a *full m-ary tree*

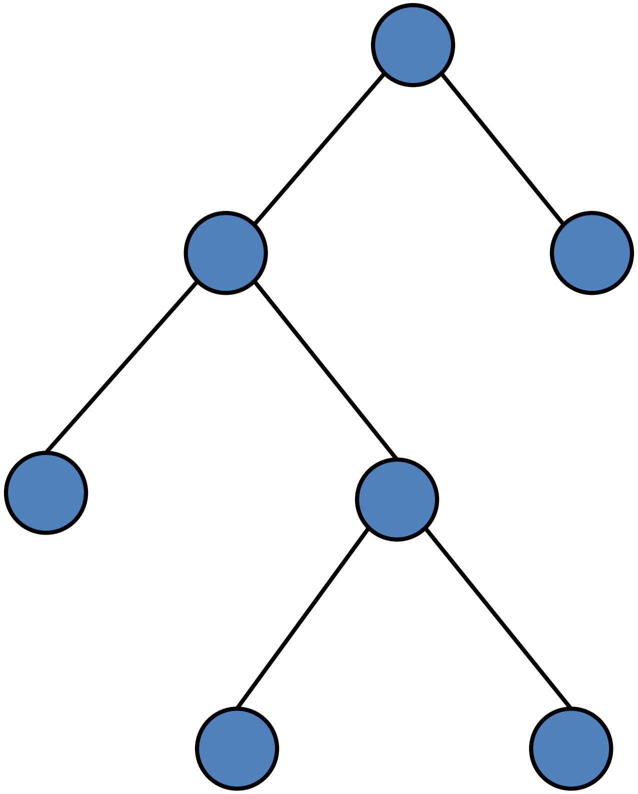
- if every internal vertex has exactly *m* children

- An *m*-ary tree with *m* = 2 is called a *binary tree*

Example

- What is the *arity* of this tree?
- Is this a full *m*-ary tree?

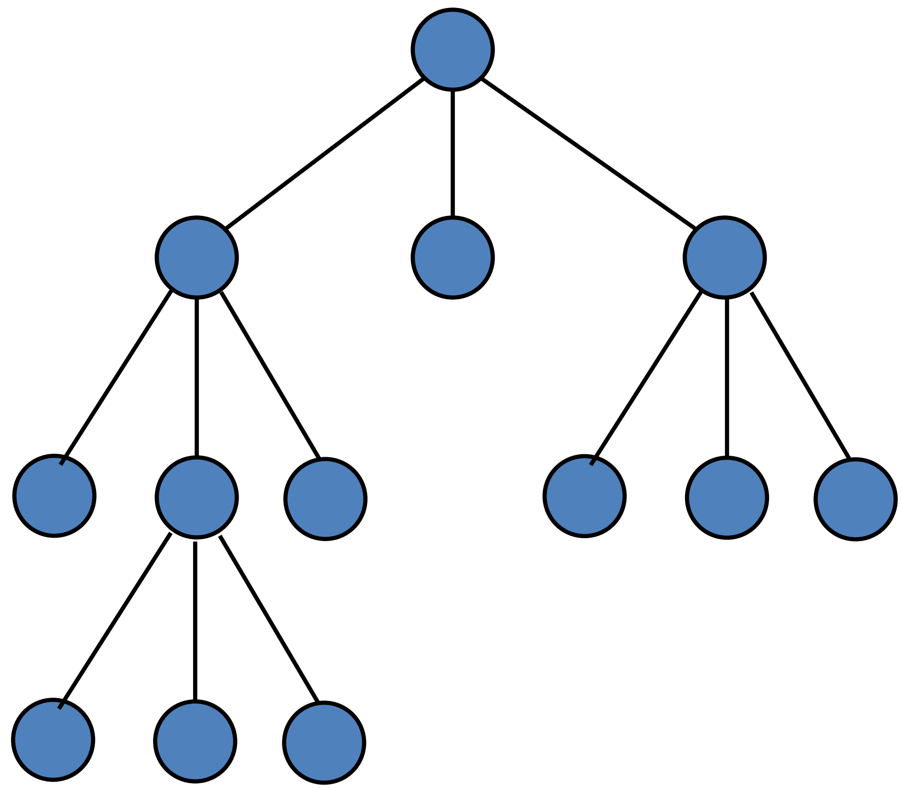
- This is a 2-ary, or *binary*, tree.
- Yes, this is a full binary tree, since every internal vertex has exactly 2 children.



Example

- What is the *arity* of this tree?
- Is this a full *m*-ary tree?

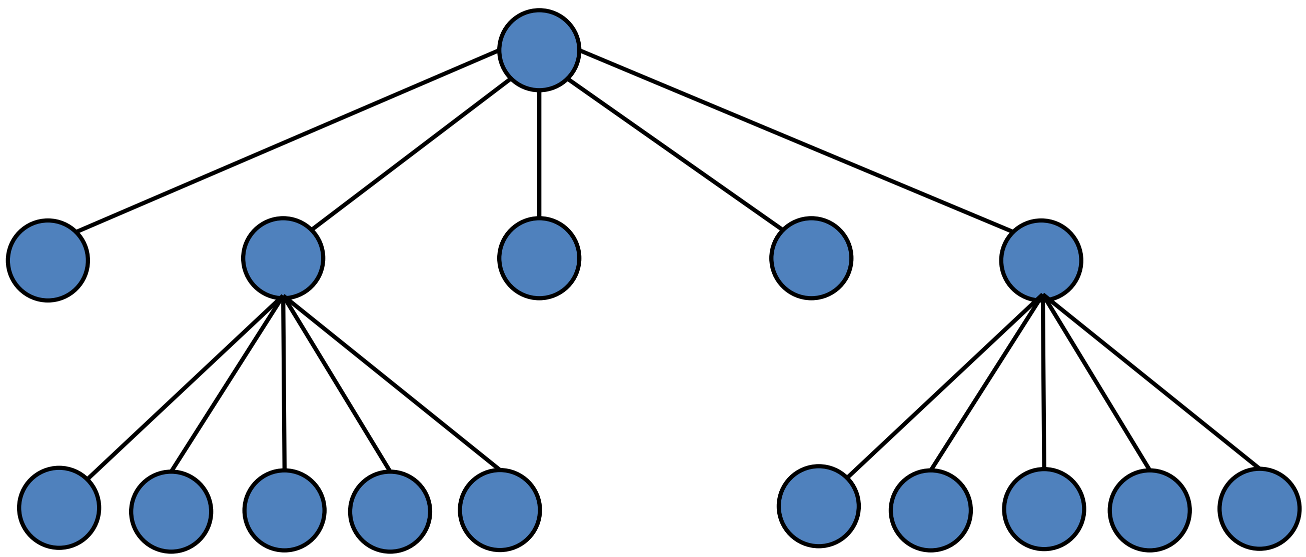
- This is a 3-ary tree.
- Yes, this is a full 3-ary tree, since every internal vertex has exactly 3 children.



Example

- What is the *arity* of this tree?
- Is this a full *m*-ary tree?

- This is a full 5-ary tree.

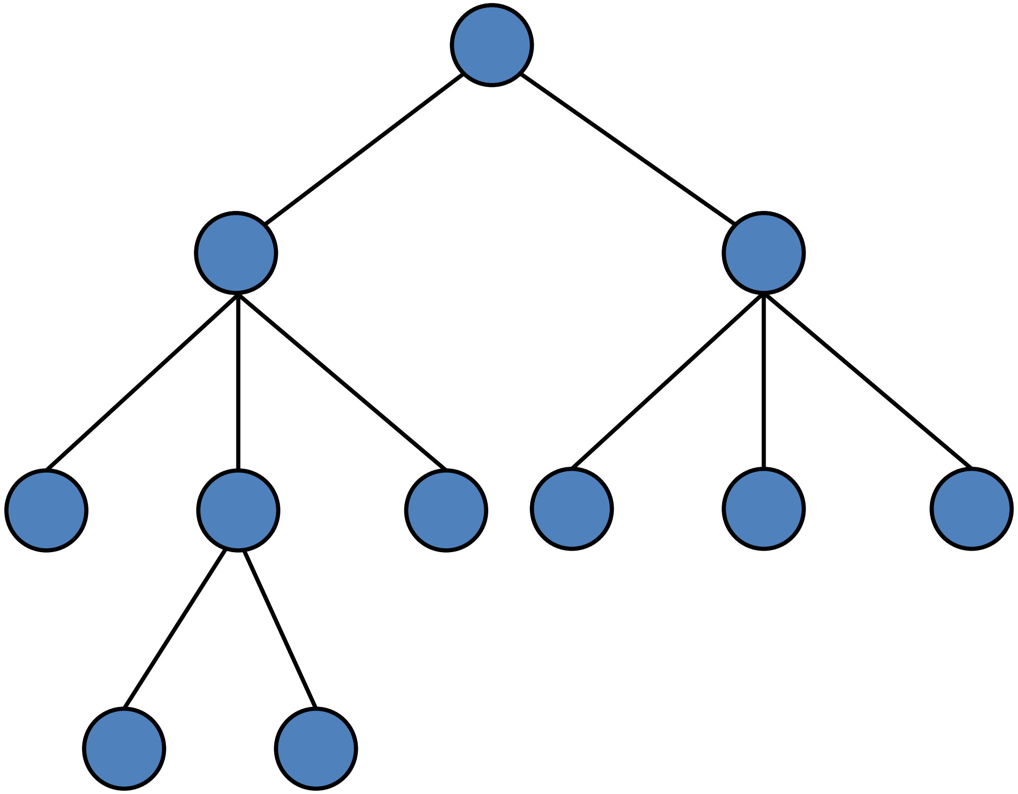


Example

- What is the *arity* of this tree?
- Is this a full *m*-ary tree?

Some internal nodes have 2 children, but some have 3, so this is a 3-ary tree.

It is not a full-3-ary tree, since one internal node has only 2 children.



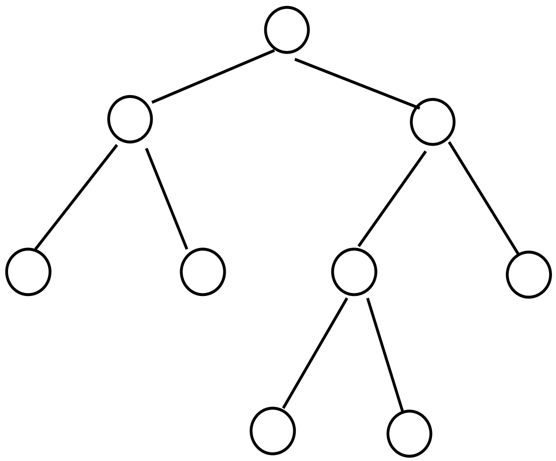
Full and Complete Binary Trees

A **full binary tree** is a binary tree in which each node is either a leaf node or has degree 2 (i.e., has exactly 2 children).

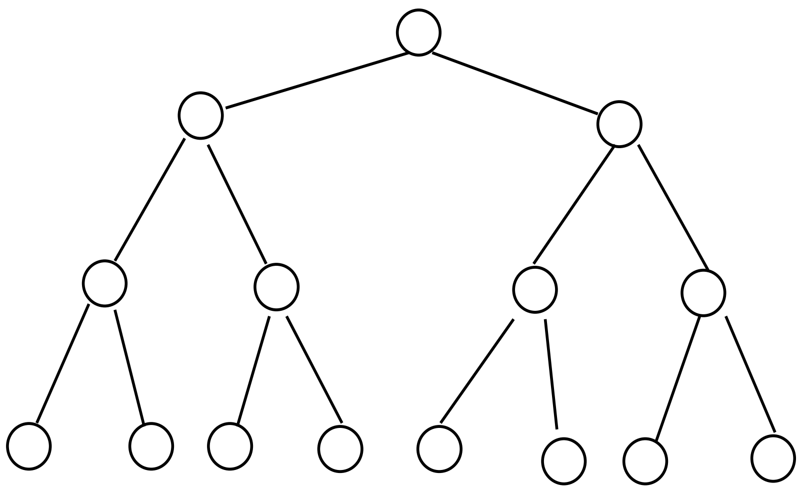
A **complete binary tree** is a full binary tree in which all leaves have the same depth.

Examples

Full binary tree:



Complete binary tree:



Trees

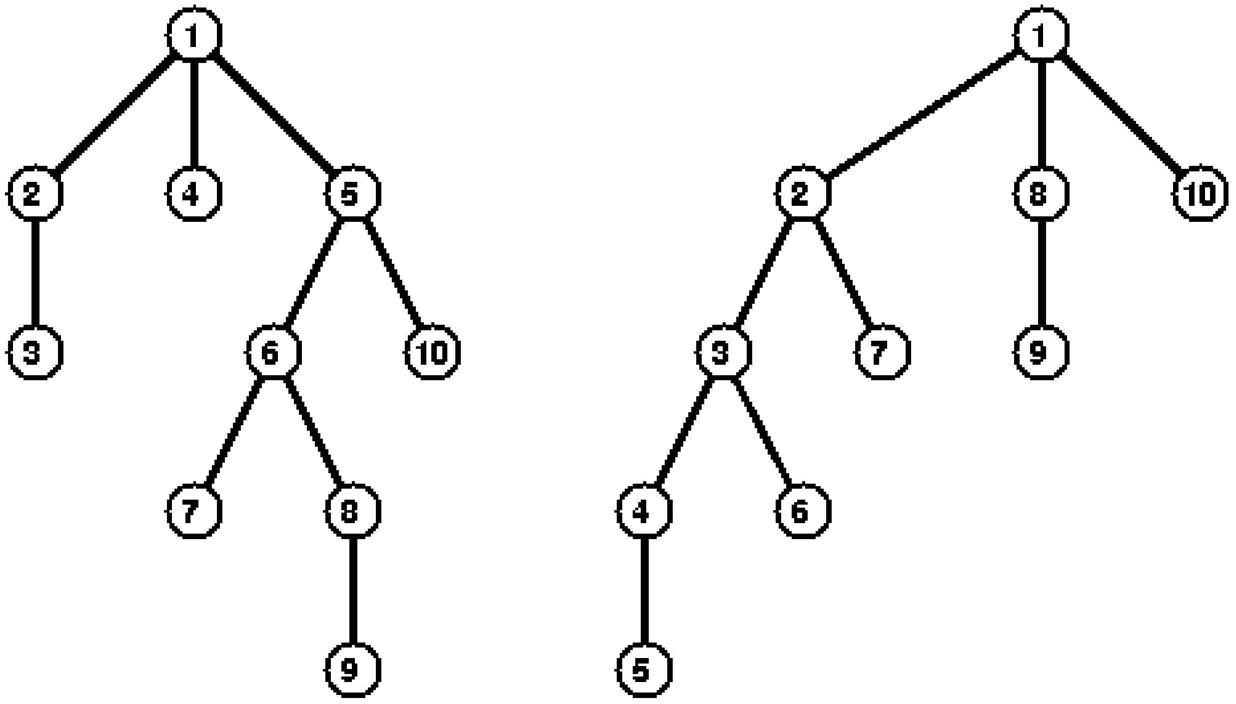
Ordered Rooted Tree

- An ordered rooted tree is a rooted tree

- where the children of each internal vertex are ordered

- Ordered trees are drawn so that the children of each internal vertex are shown in order from left to right

Trees

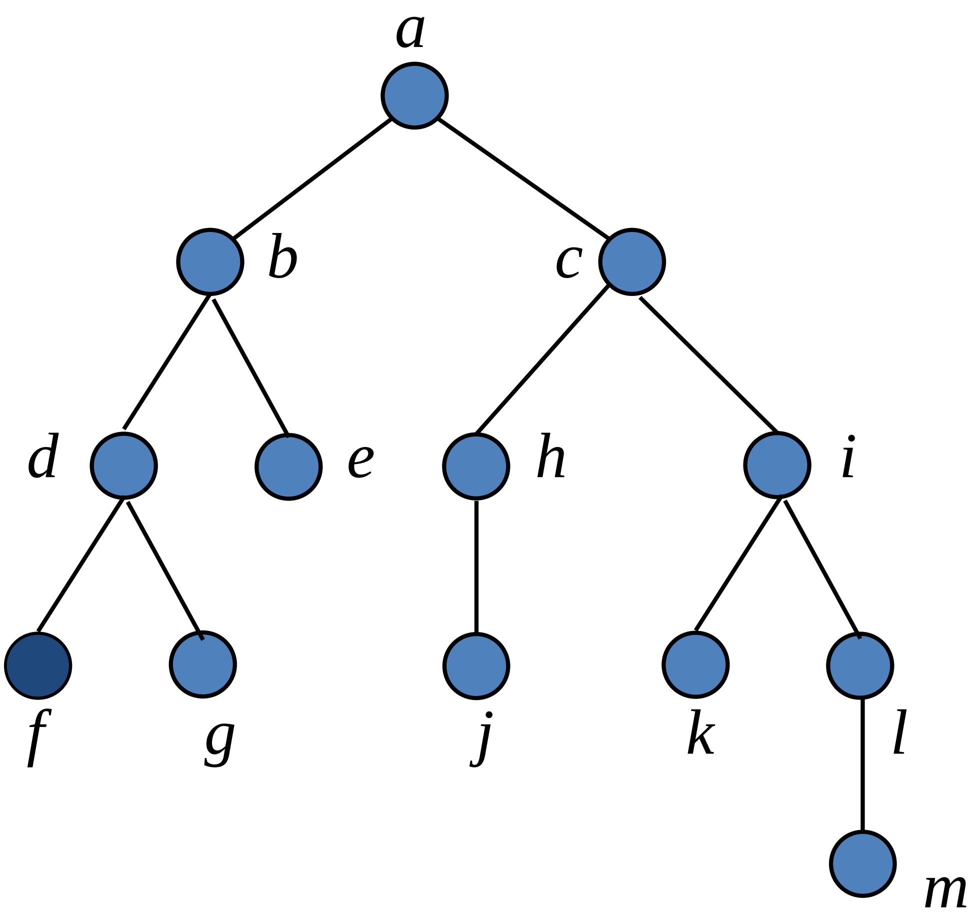


Ordered Rooted Tree

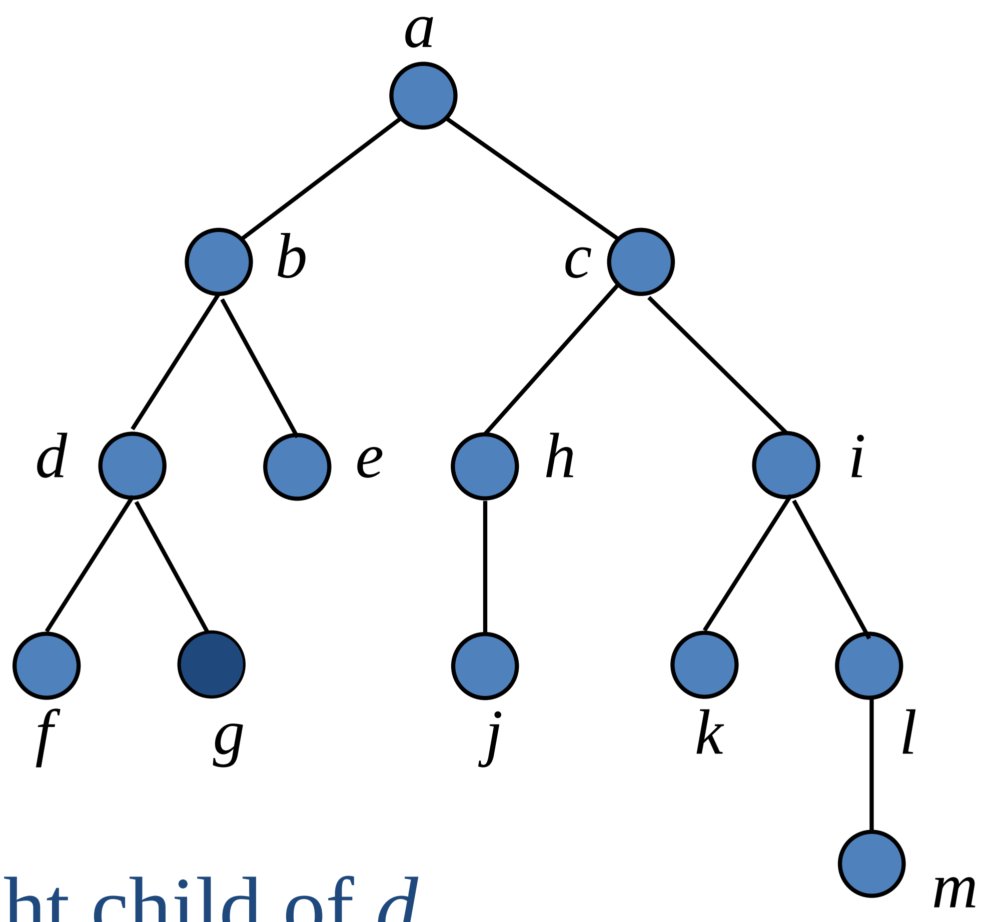
- In an *ordered binary tree*, if an internal vertex has two children, then they are called *left child* and *right child*.
- The subtree rooted at the left child of a vertex is called the *left subtree* and subtree rooted at the right child of a vertex is called the *right subtree*.

Example

Left child of *d*

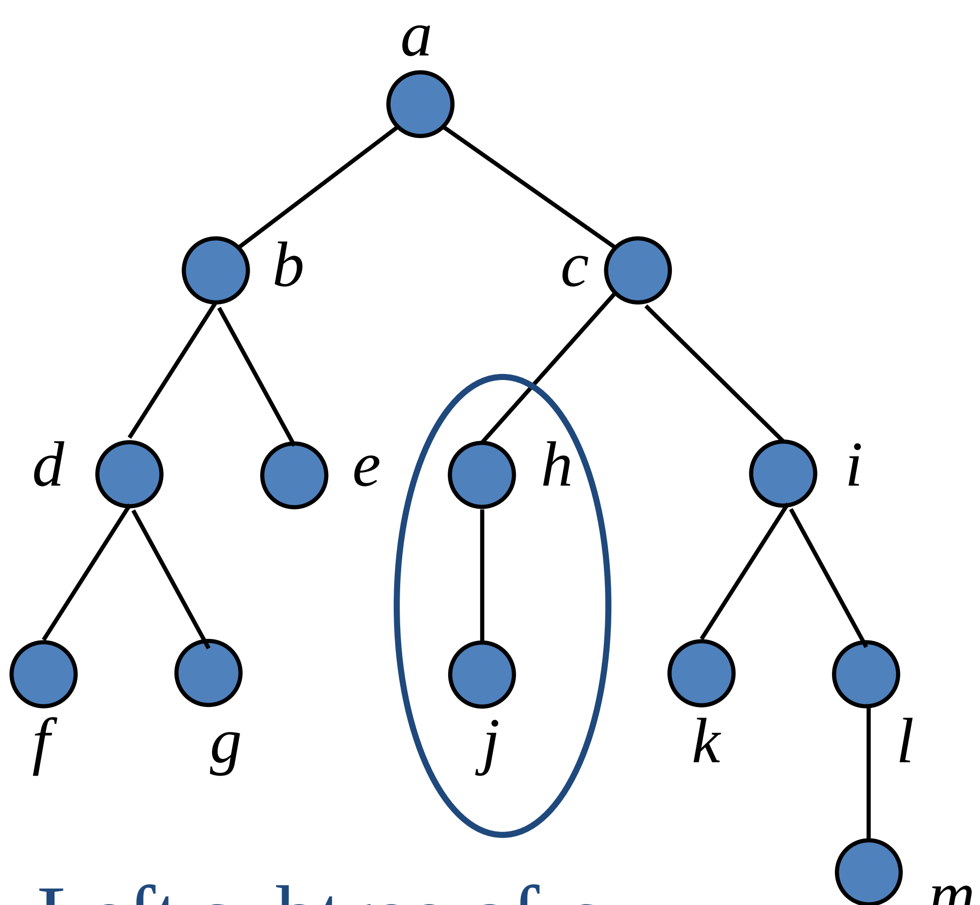


Example



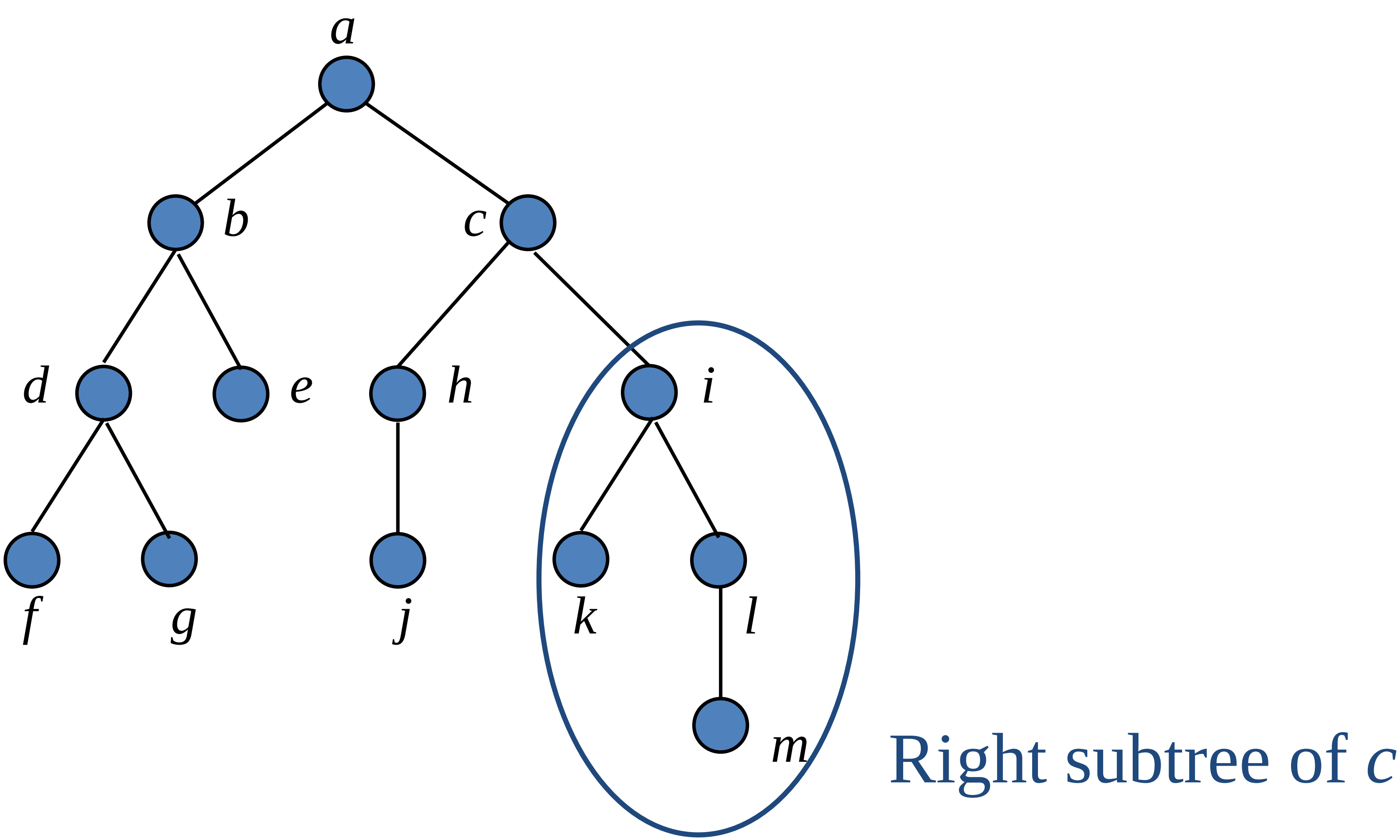
Right child of *d*

Example

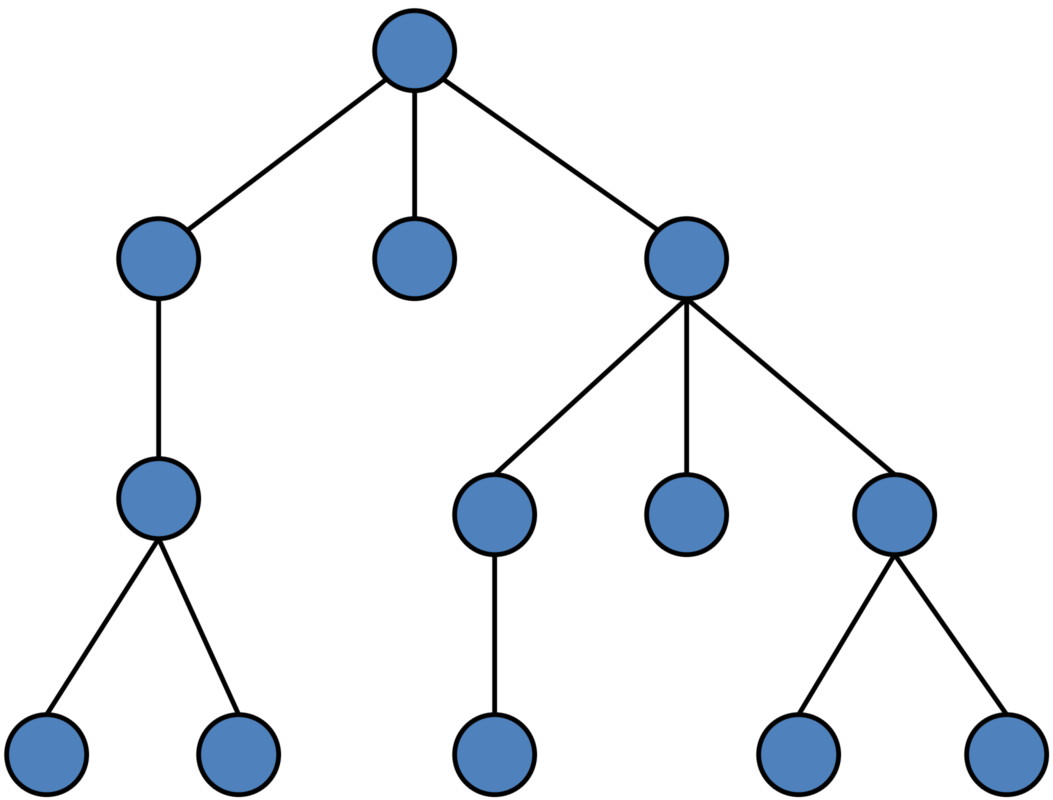


Left subtree of *c*

Example



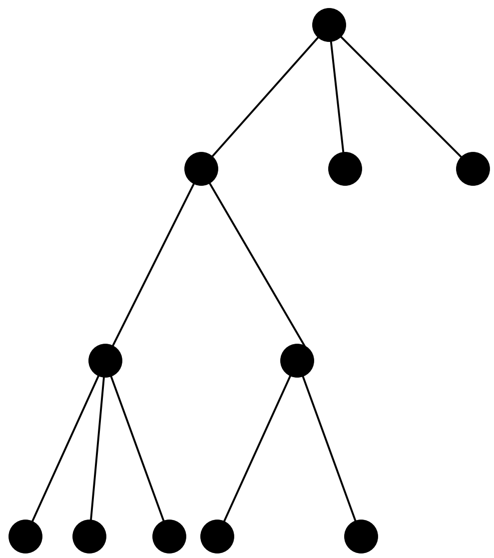
Analogy



Trees

Properties of Trees

A tree with n vertices has n-1 edges.

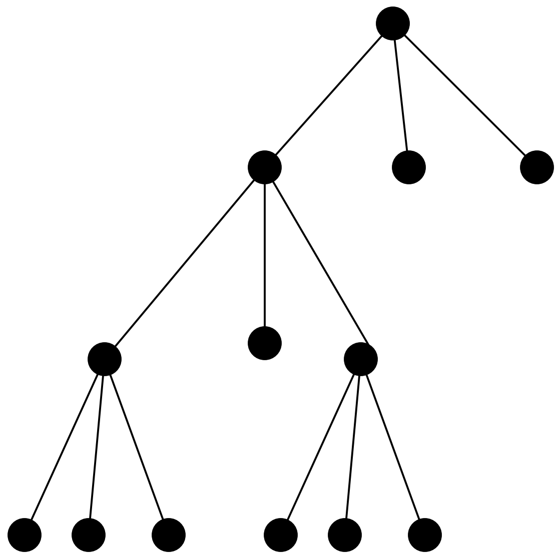


11 vertices, 10 edges

Trees

Properties of Trees

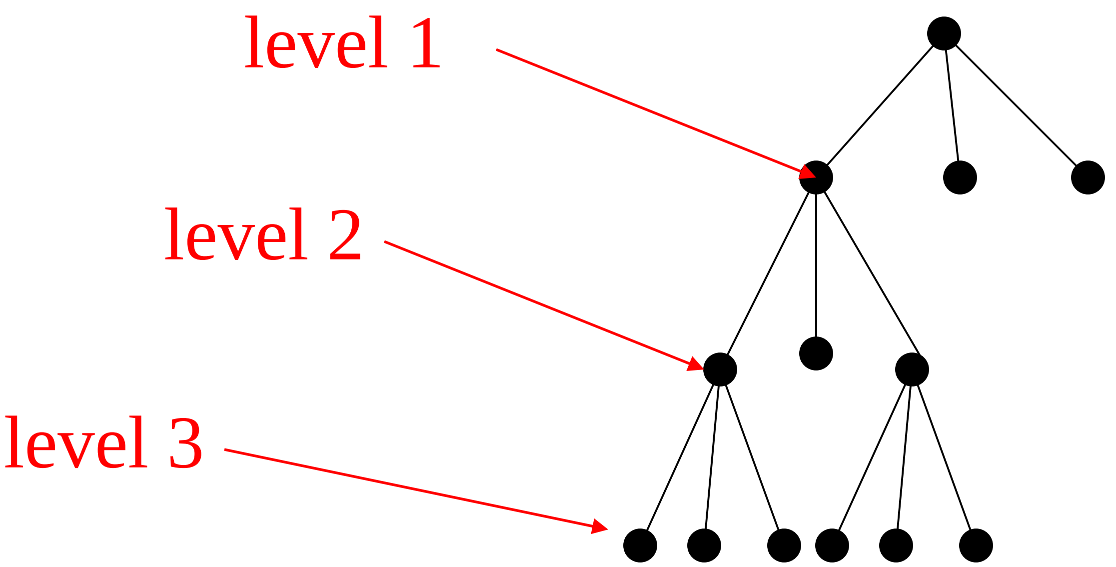
A full m-ary tree with i internal vertices contains $n = mi + 1$ vertices.



Trees

Properties of Trees

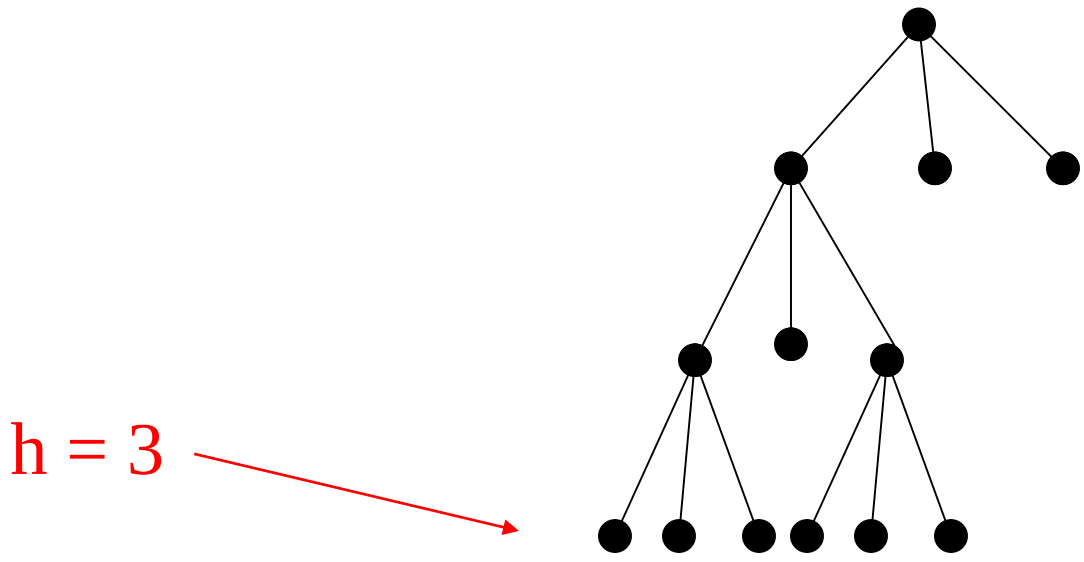
The level of a vertex *v* in a rooted tree is the length of the unique path from the root to this vertex.



Trees

Properties of Trees

The *height* of a rooted tree is the maximum of the levels of vertices.



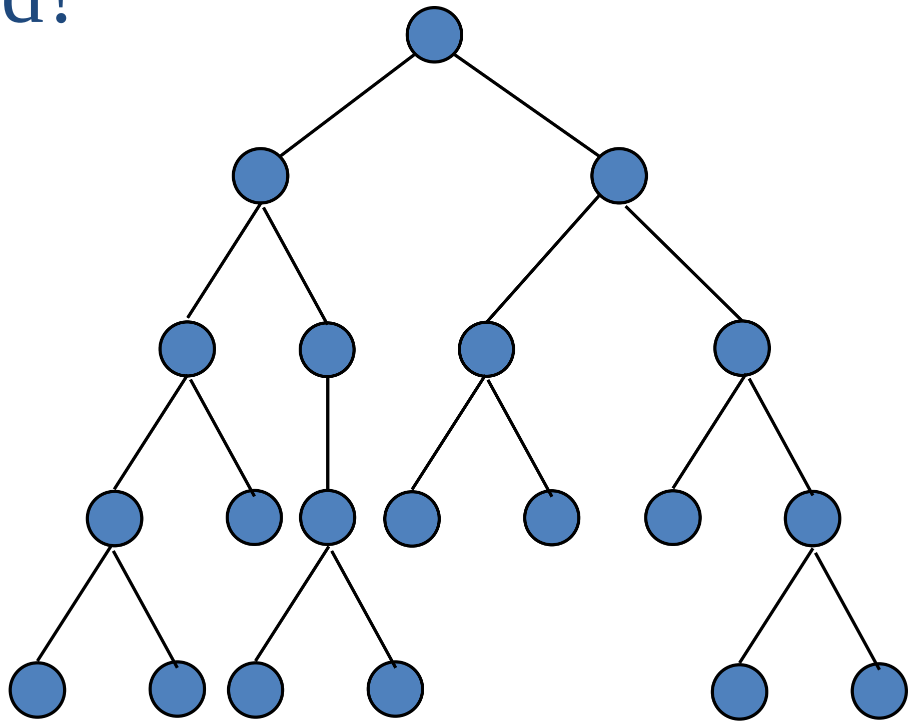
Trees

Properties of Trees

- A rooted m-ary tree of height h is called *balanced*
- if all leaves are at levels h or h-1.

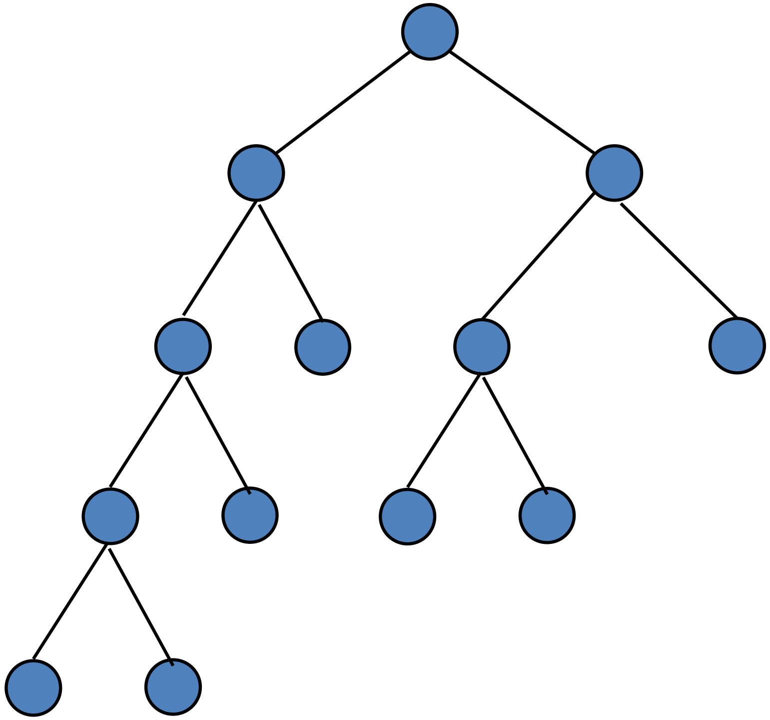
Example

Is this tree balanced?



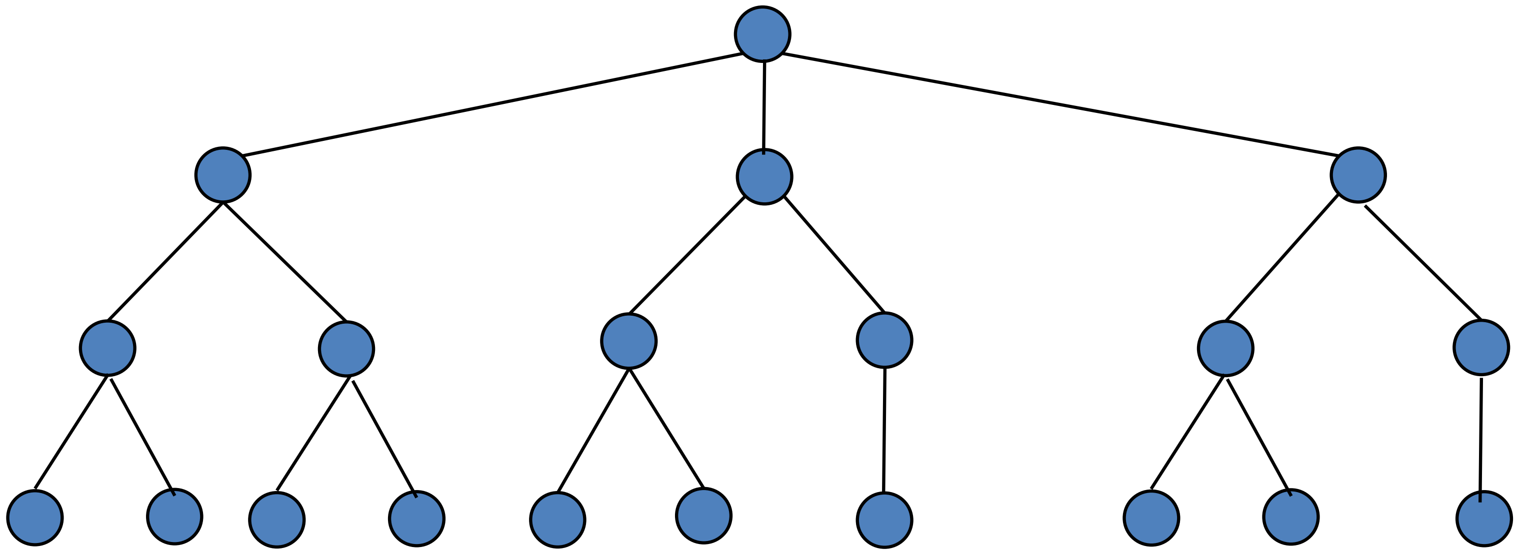
Example

Is this tree balanced?



Example

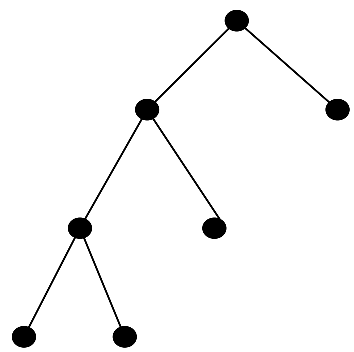
Is this tree balanced?

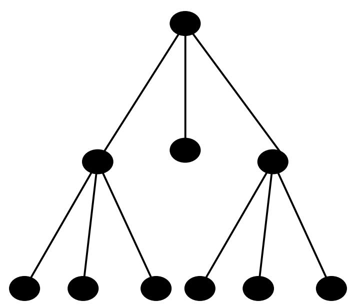


Trees

Properties of Trees

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



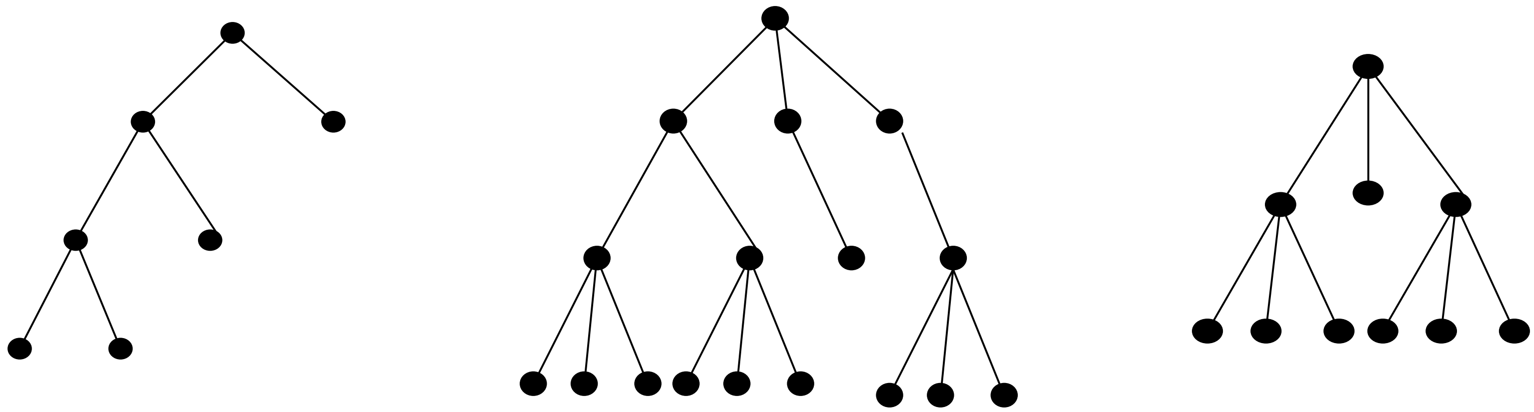


Trees

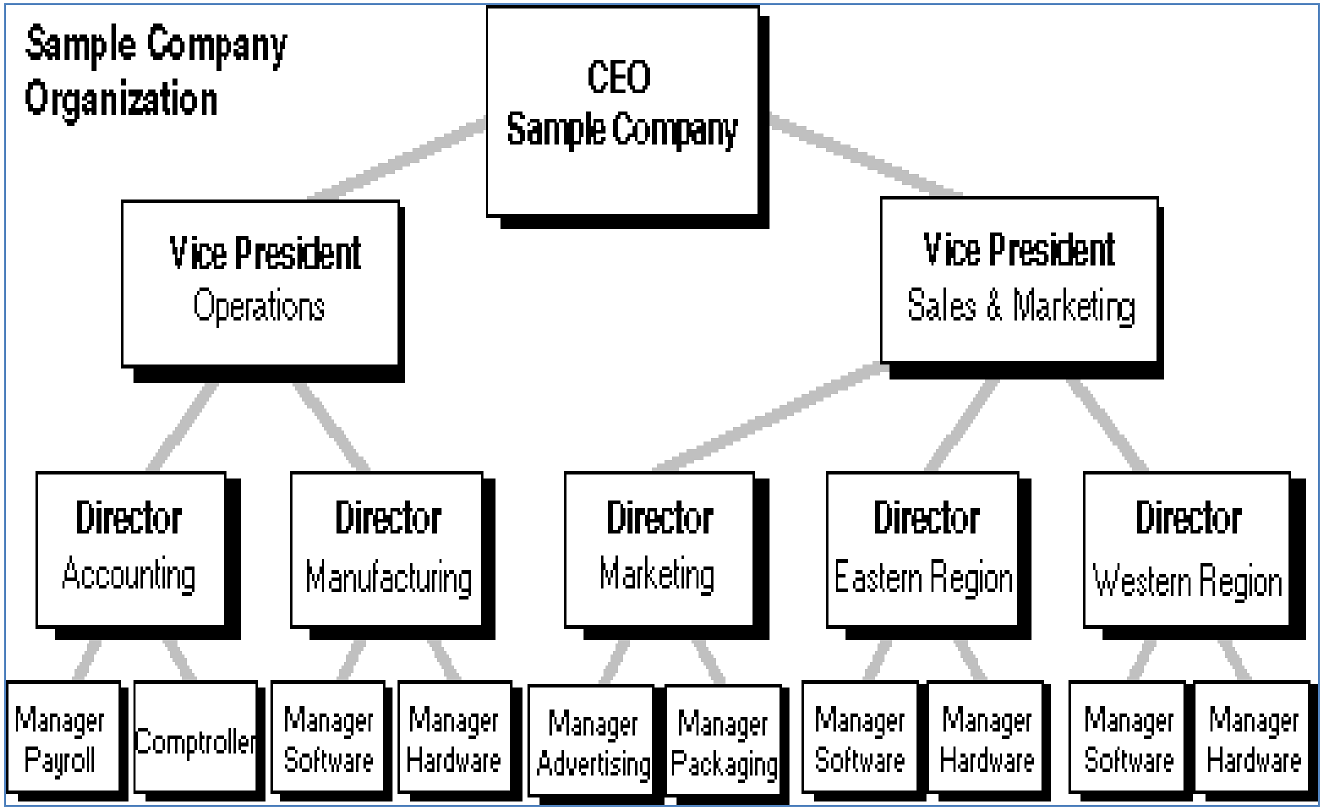
Properties of Trees

If an m-ary tree of height h has l leaves, then

$$h \geq \lceil \log_m l \rceil$$



Practical Examples



Practical Examples