# **Binary Tree Introduction**

## **N-ary Tree**

**Formal n-ary Tree Definition**

An **n-ary tree** is a set ***T*** of nodes that is either empty or partitioned into disjoint subsets:

* A single node ***r***, the root
* n possibly empty sets that are n-ary subtrees of ***r***

Each node can have no more than **n children**. (But can have <= n children!)

The following tree is an example of an n-ary tree with n = 3.

A picture containing line chart

Description automatically generated

An n-ary tree is **not a special kind of general tree**.

An **n-ary tree can be** **empty**

A **general tree** **cannot be empty**.

If an n-ary tree has the restriction that every node has at most two children, it is a **binary tree.**

## **Binary Tree**

**Formal Definition of a Binary Tree**

A **binary tree** is a set ***T*** of nodes that is either empty or partitioned into disjoint subsets:

* A single node r, the root
* Two possibly empty sets that are binary trees, called left and right subtrees of r

A binary tree is **a special kind** ofn-ary tree.

Each node in a binary tree can have no more than **two children**.

**Example**

The following is an example of a binary tree:

A picture containing text, clock

Description automatically generated

As we did with linked lists, we refer to each different data value as a node.

This tree has a total of six nodes.

Some people joke that computer scientists view the world upside down, so imagine turning the diagram around the other way:

A picture containing text, clock

Description automatically generated

**Intuitive Definition of a Binary Tree**

**T is a binary tree if either**

* T has no nodes, or
* T is of the form

*r*

page455image1660818112page455image1660819840

*TL*  *TR*

where ***r***is a node and ***TL***and ***TR***are both binary trees

Notice that the formal definition agrees with this intuitive one:

If ***r***is the root of ***T***, then

the binary tree ***TL***is the **left** **subtree** of node ***r***

and

the binary tree ***TR***is the **right** **subtree** of node ***r***

If *TL* is not empty, its **root** is the **left child** of ***r***

If *TR* is not empty, its **root** is the **right child** of ***r***

Notice that if **both subtrees of a node are empty**, that node is a **leaf**.