C++
Trees

CO658 Data Structures & Algorithms

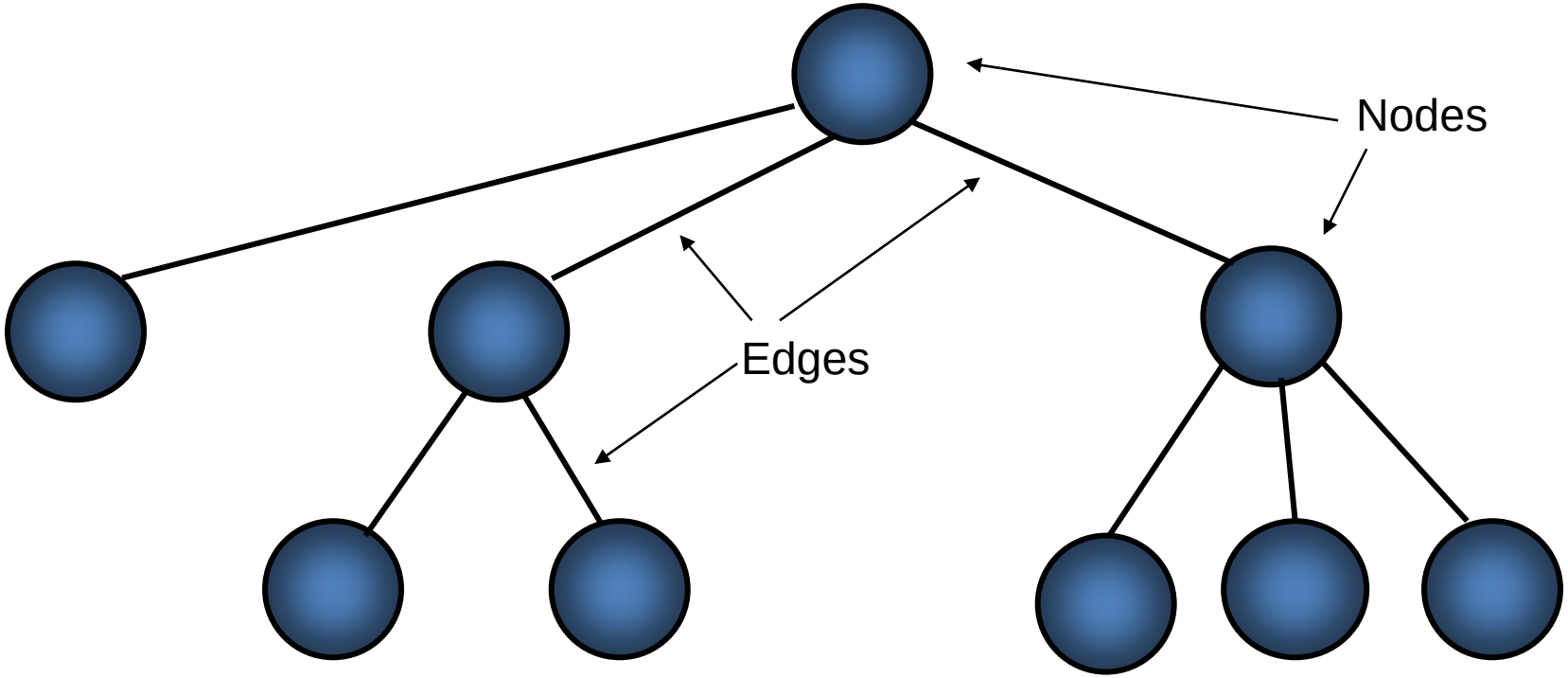
What are they?

- A data structure that combines the best features of an ordered array and linked list.
- Quick searching (ordered array)
- Fast insertion and deletion (linked list)

Structure

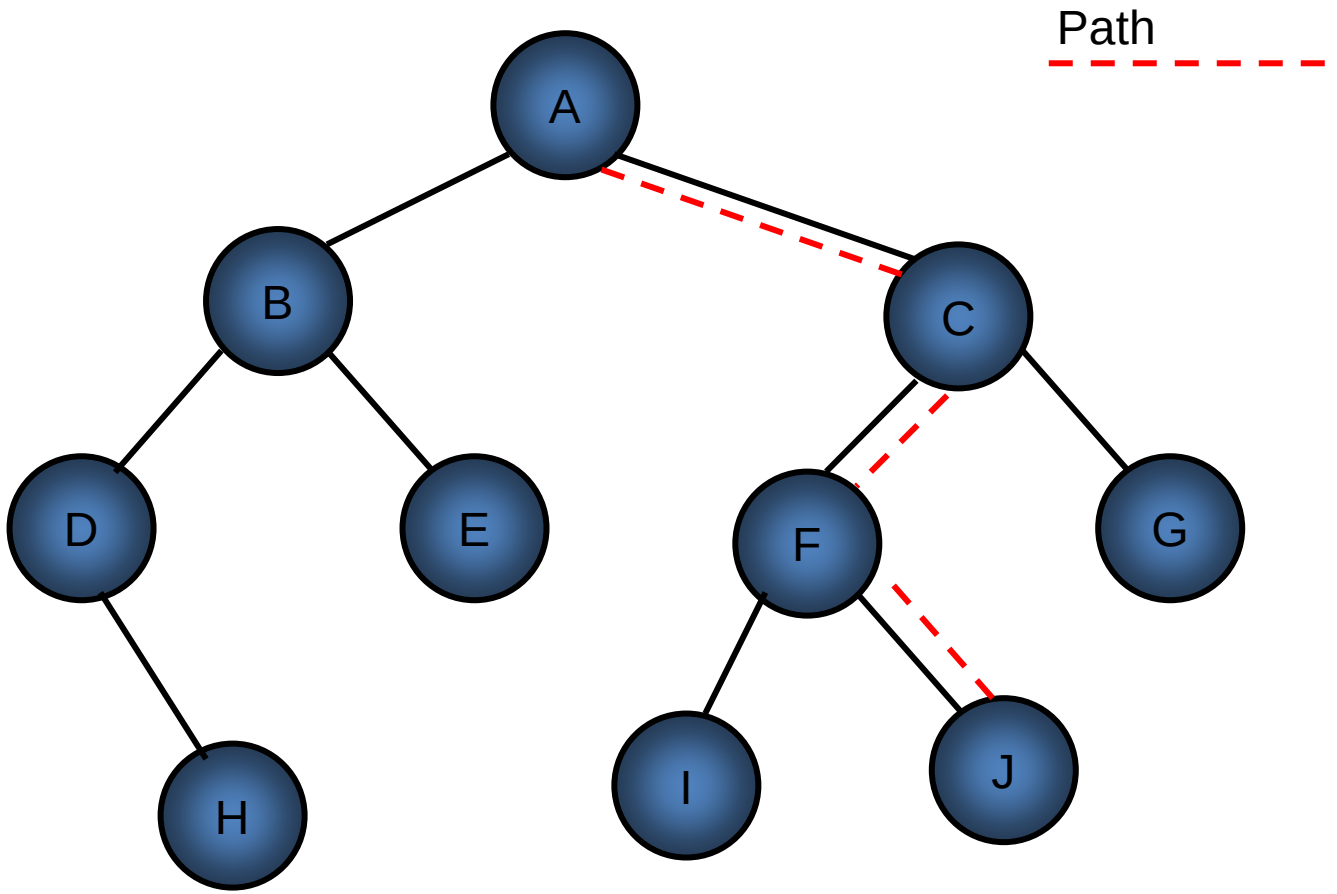
Consists of

- Nodes – representing entities (objects)
- Edges – represent the relationships between nodes



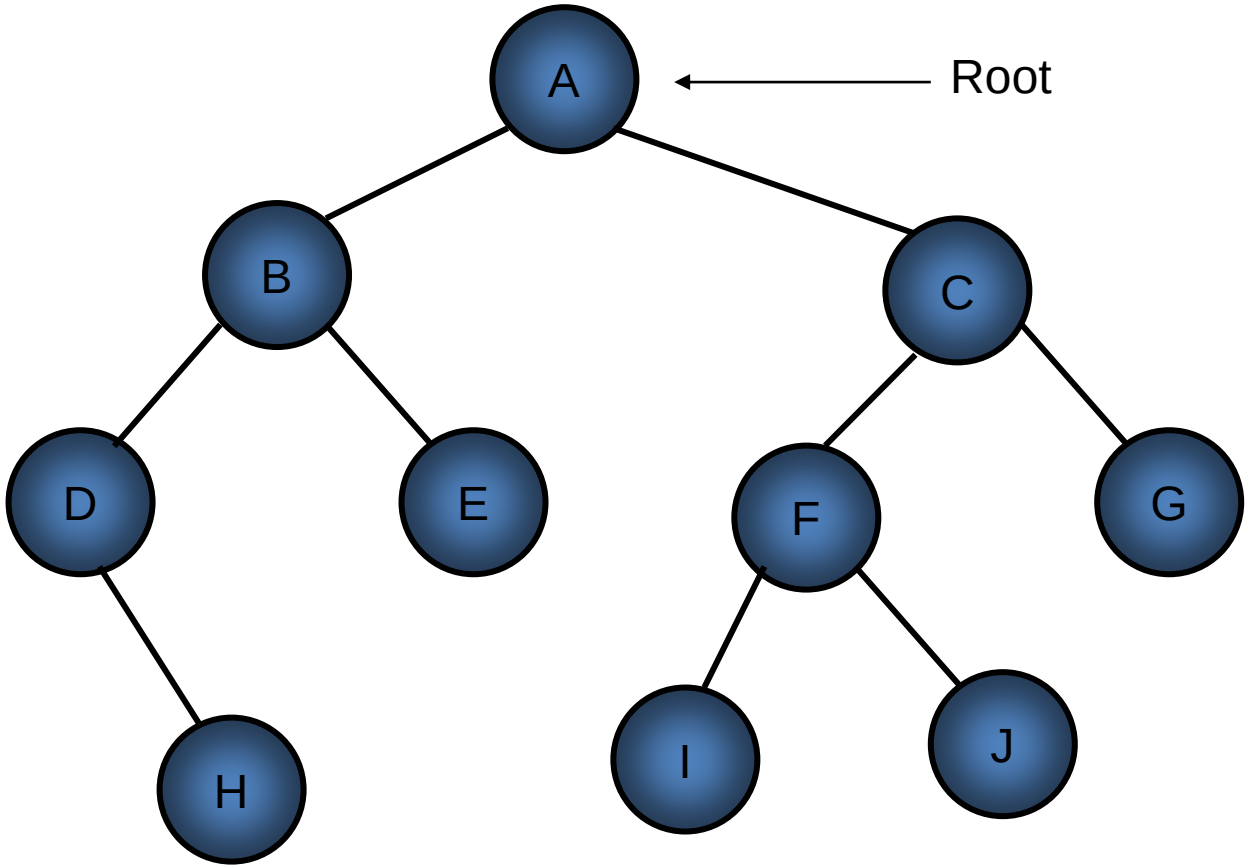
Terminology : Path

A sequence of nodes connected by edges



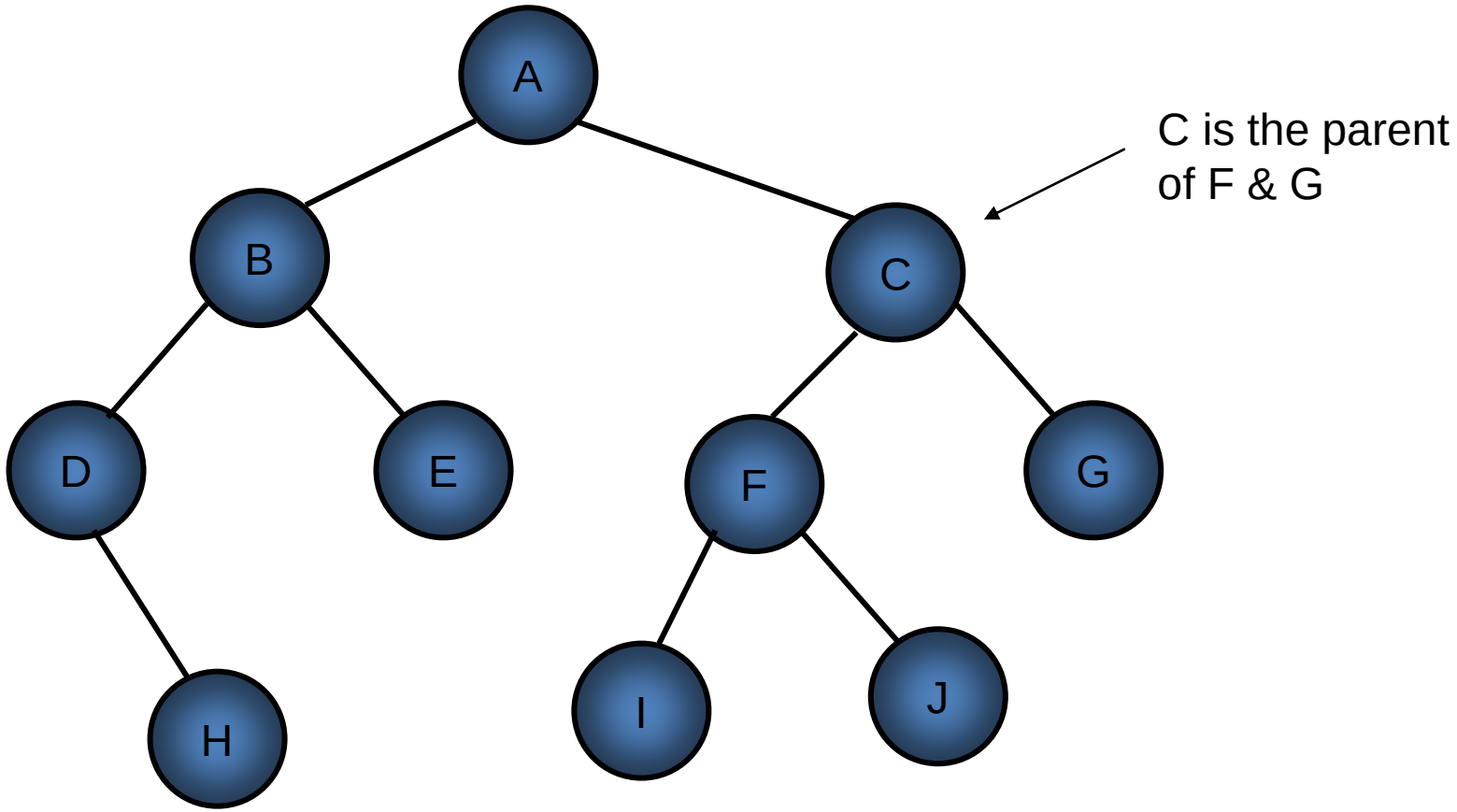
Terminology : Root

The node at the top of the tree. A tree only has one root.



Terminology : Parent

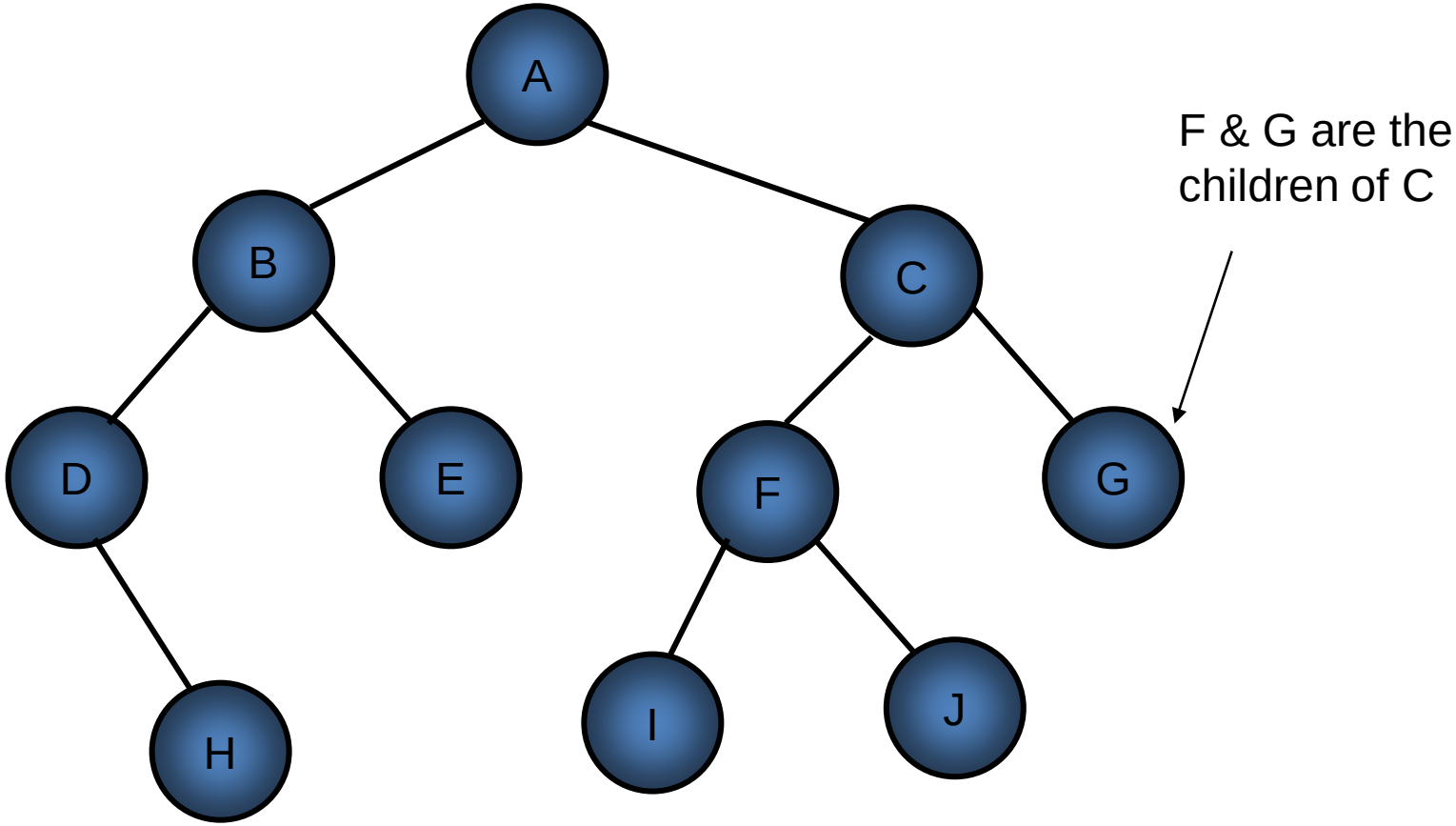
All nodes (except the root) have a single edge running up the tree. The node this edge is connected to is the parent of the current node.



Terminology :Child

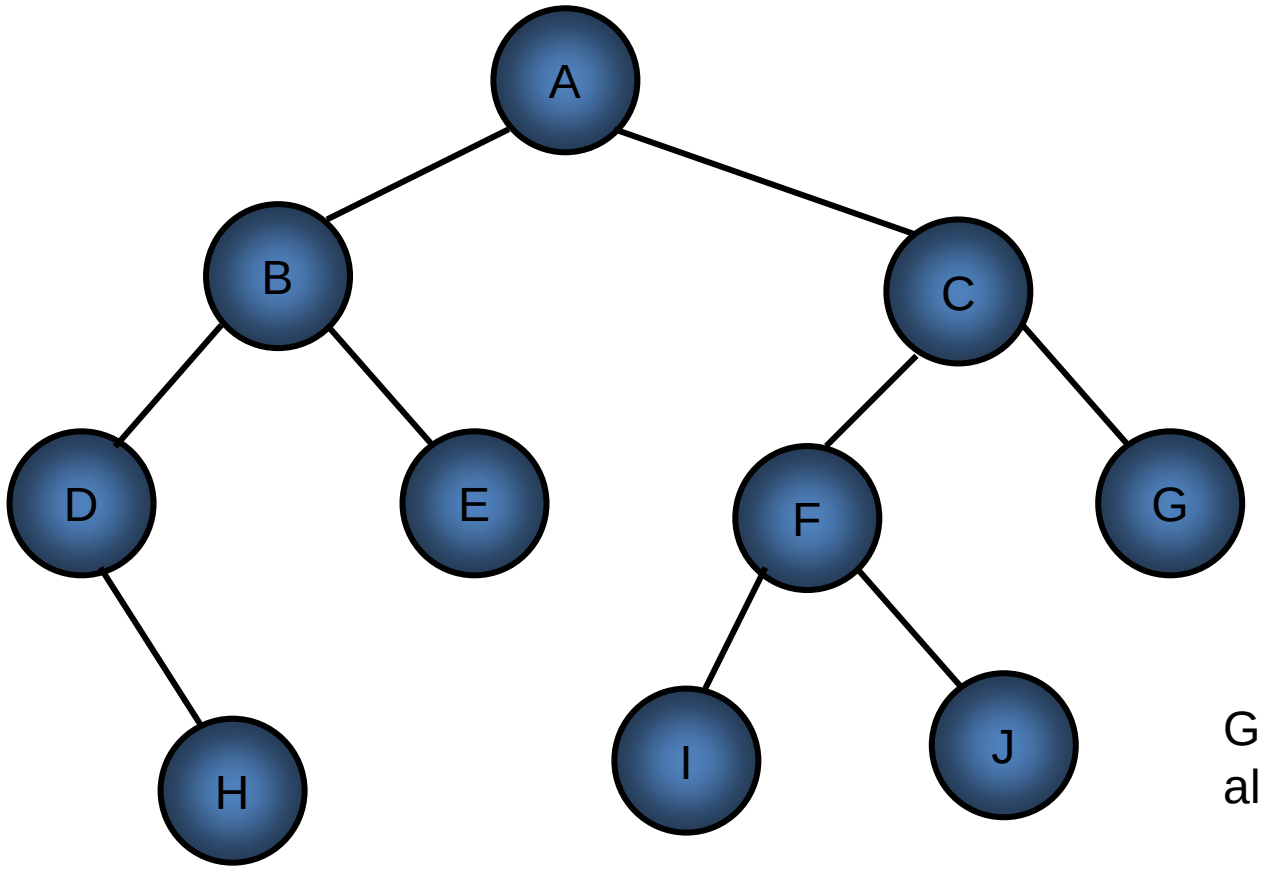
Nodes may have edges running down the tree.

The nodes these edges connect to are the children of the current node.



Terminology : Leaf

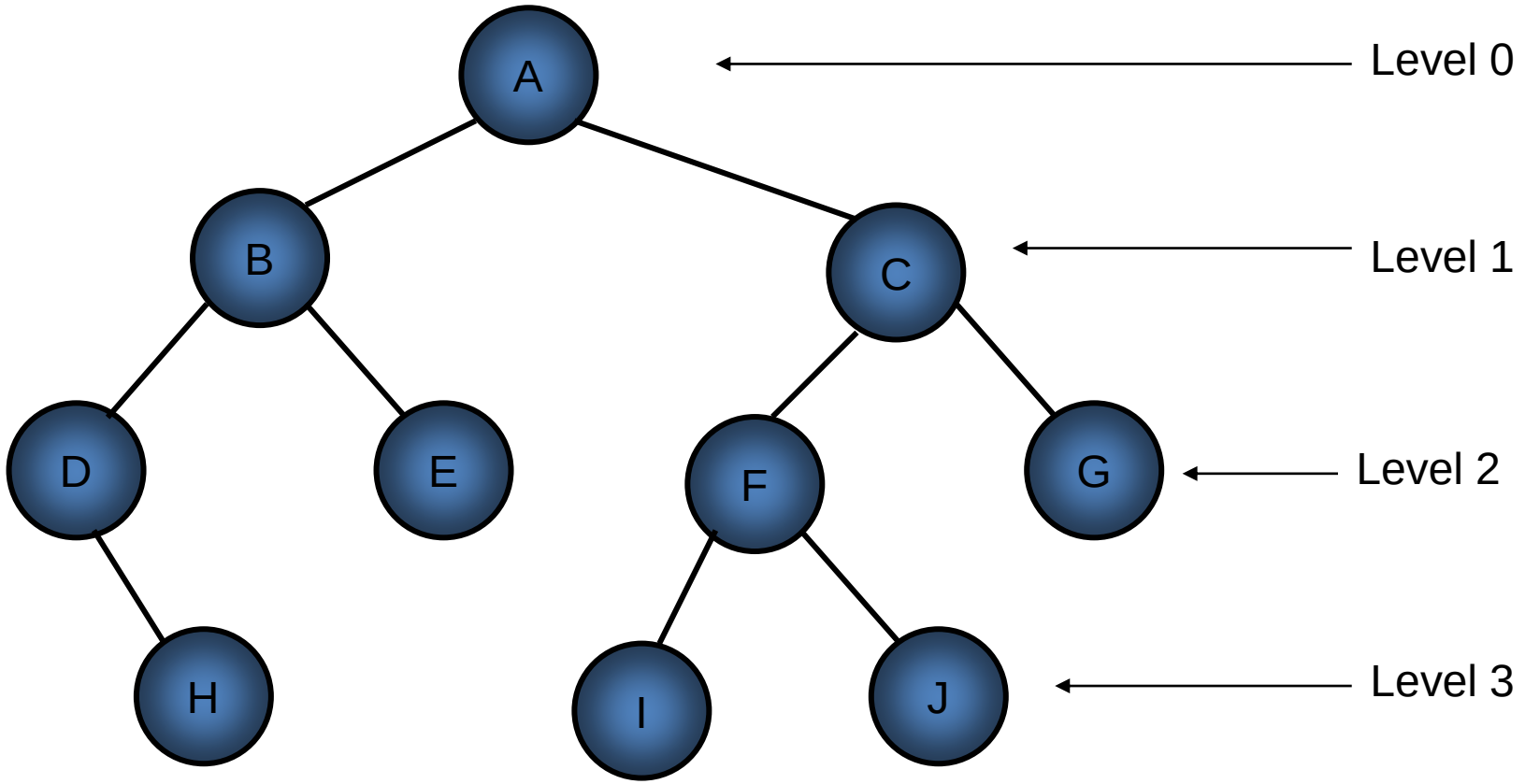
A node with no children. While a tree may only have one root node it may have many leaf nodes.



G,J,I,E & H are all leaf nodes

Terminology : Levels

The number of generations from the root, where root is considered to be level 0.

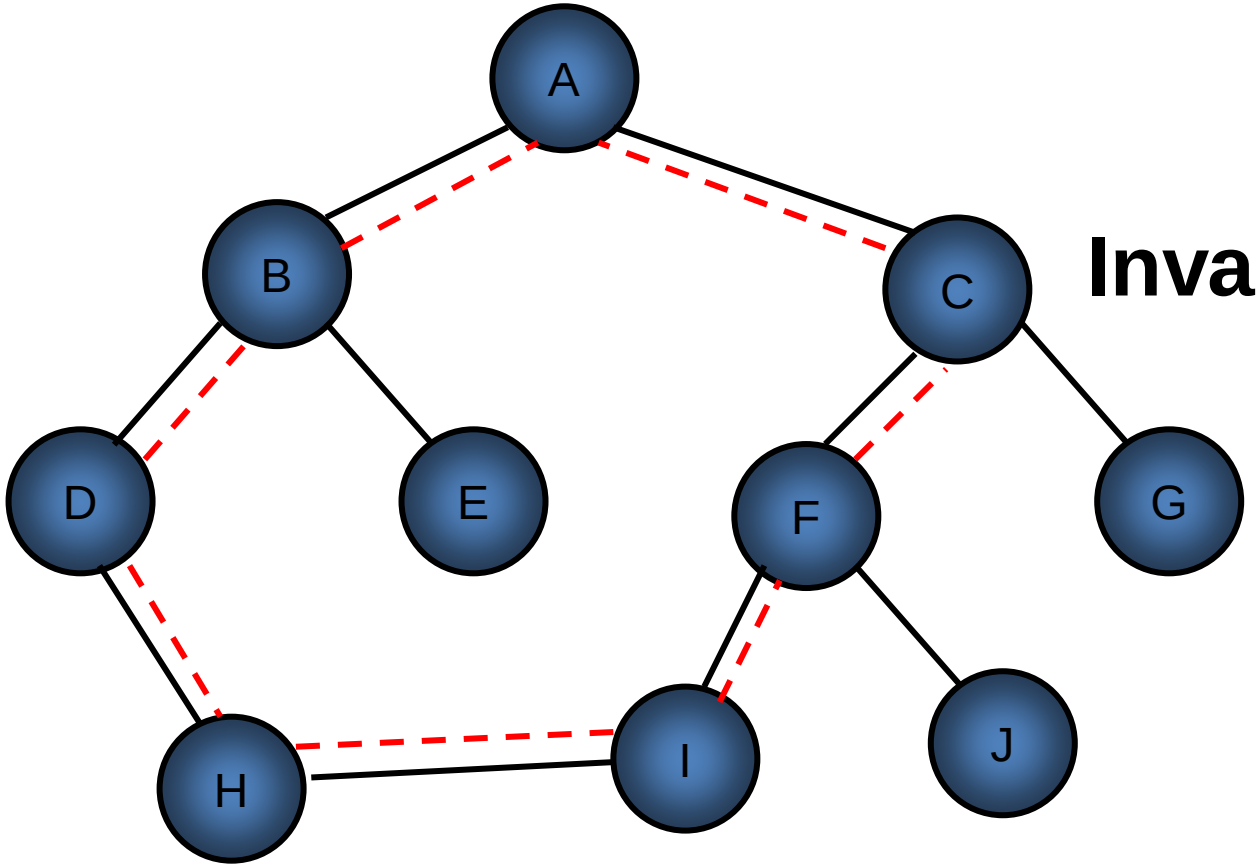


Terminology

- Visiting – Program control arrives at the node to perform some action. Passing over the node on route to another node is not considered to be visiting the node.
- Traversing – Visit all the nodes in a specified order.

Rules

The nodes within a valid tree only have one path from the root to each node.



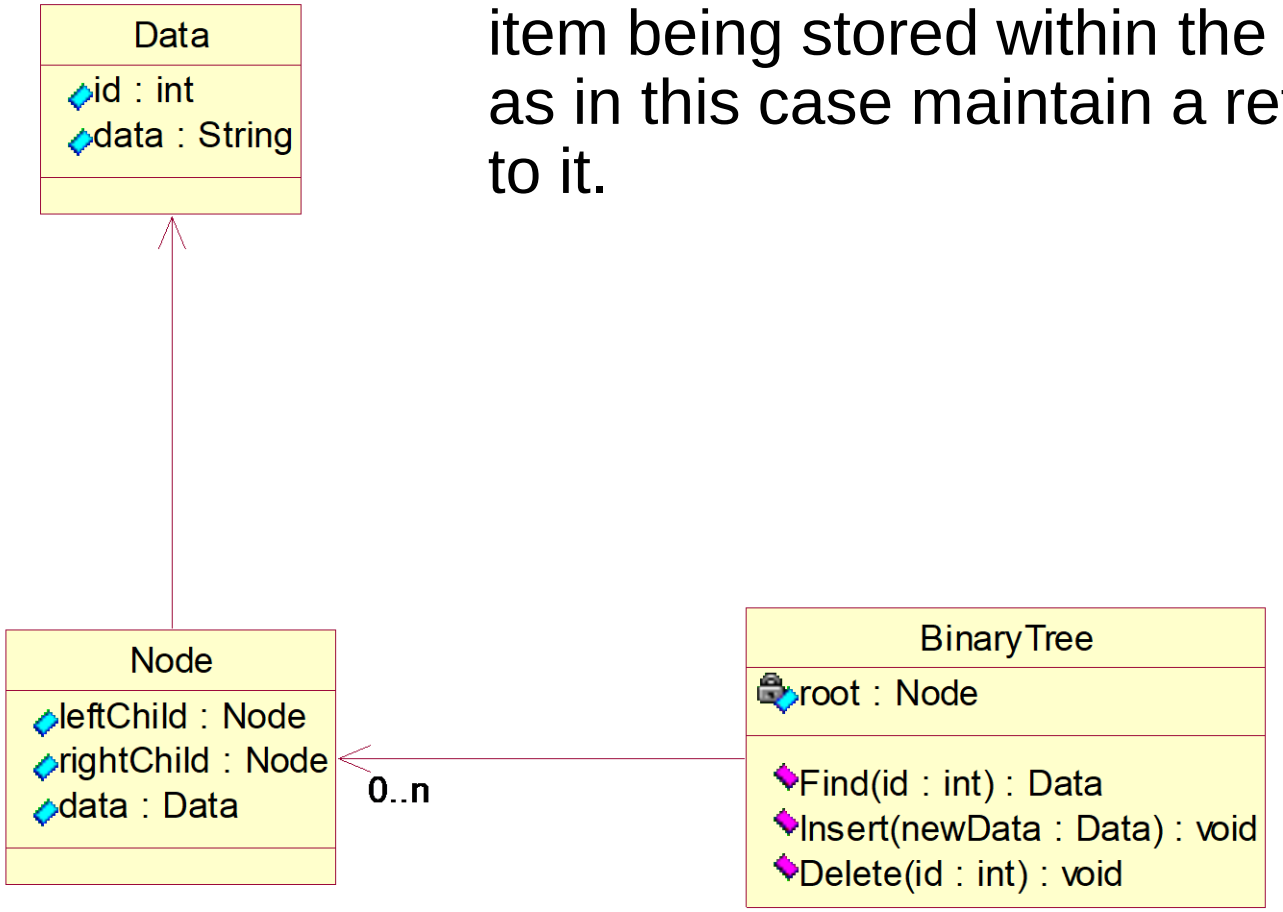
Invalid Tree

Binary Tree

- Nodes have a maximum of two children.
- The children are called the
  - Left Child
  - Right Child
- The left / right refers to the child position when the tree is depicted as a diagram.
- A Node within a binary tree may have only one (left or right) child or none, in which case it is a leaf.

Basic Design

The Node may be derived from the item being stored within the tree or as in this case maintain a reference to it.



Key Value

- Items stored within the binary tree will have a unique key value, that determines its location within the tree. This is synonymous with primary keys within database tables.
- In the previous slide the id field represents the key value.

Activity

- Attempt to set up the Node class (Exercise 1)
- Attempt to set up the Tree class (Exercise 2)

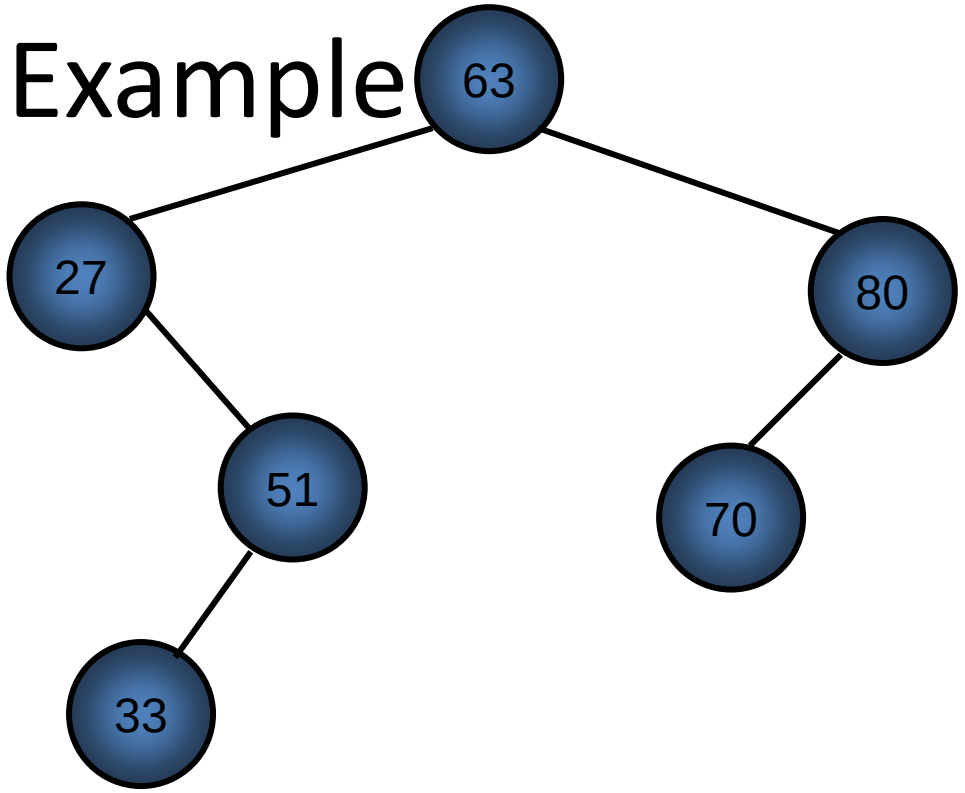
Recursion

- A technique used extensively to navigate over data structures.
- A recursive function is one that calls itself.
- The function must contain logic to ensure it stops the recursive calls.
- If not it will generate a stack overflow exception.

Inserting A Node

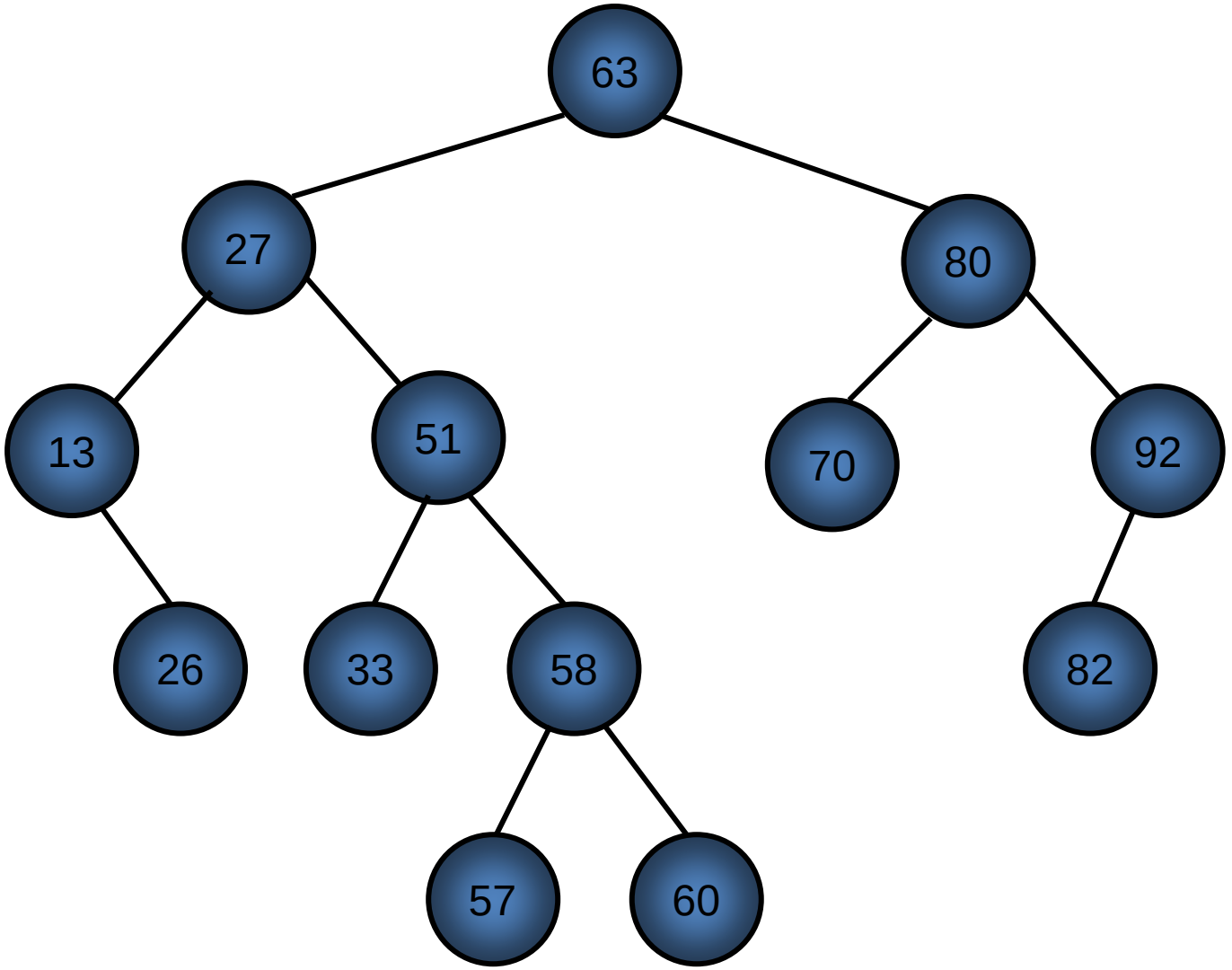
- The first node to be inserted into the tree will become the root node.
- The second insertion will compare its key value with that of root. If greater then it becomes the roots right child if less the left child.
- Subsequent insertions compare their key values with the root and then its left or right child depending upon it being less than or greater than the root’s key value.
- The process continues down the tree until an appropriate left or right child with a null value is found. At this point the new node is assigned to the child.

Insert Example



- Insert (63) : Root
- Insert (80) : > Root
- Insert (70) : > Root, < Root.RightChild
- Insert (27) : < Root
- Insert (51) : < Root, > Root.LeftChild
- Insert (33) : < Root, > Root.LeftChild, < Root.LeftChild.RightChild

Binary Tree Key Value Example



BST Algorithm for Insert

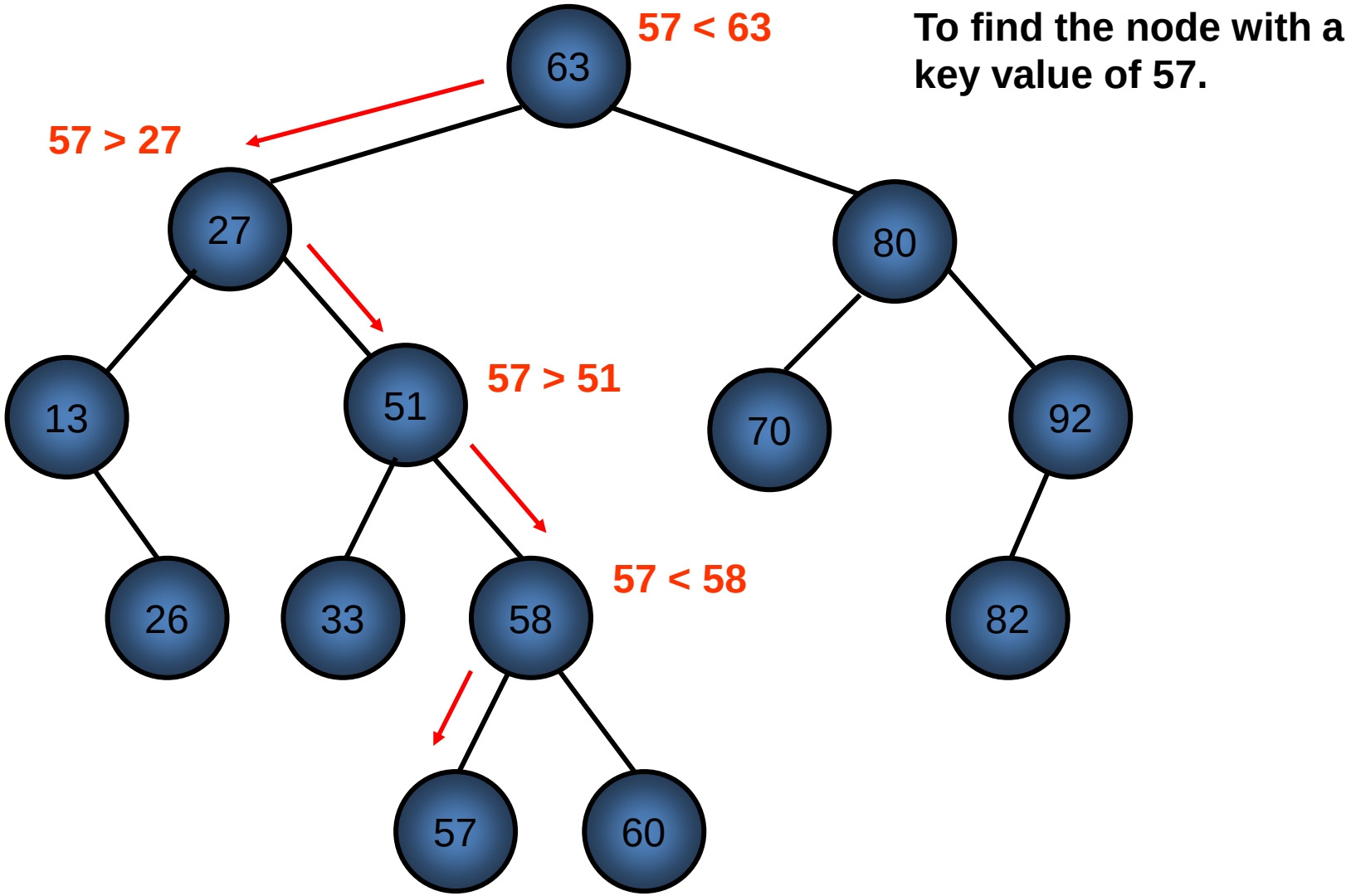
insert (I, T) § true

```
if empty(T) then
        root(T) = I
        right(T) = null
        left(T) = null
    else if I < root(T)
        insert(I, left(T))
    else if I > root(T)
        insert(I, right(T))
    else if I = root(T)
        error
```

Searching The Binary Tree

- Similar to insertion.
- Key value compared with that of Root’s and left or right path followed down tree until matching key value found.
- Typically the Find method returns the matching node or null if not found.

Search Example



To find the node with a key value of 57.

BST Algorithm for Search

```
search(T, I) § boolean
if empty(T)
      return false
else if I = root(T)
      return true
else if I < root(T)
      return (search(left(T), I))
else
      return (search(right(T), I))
```

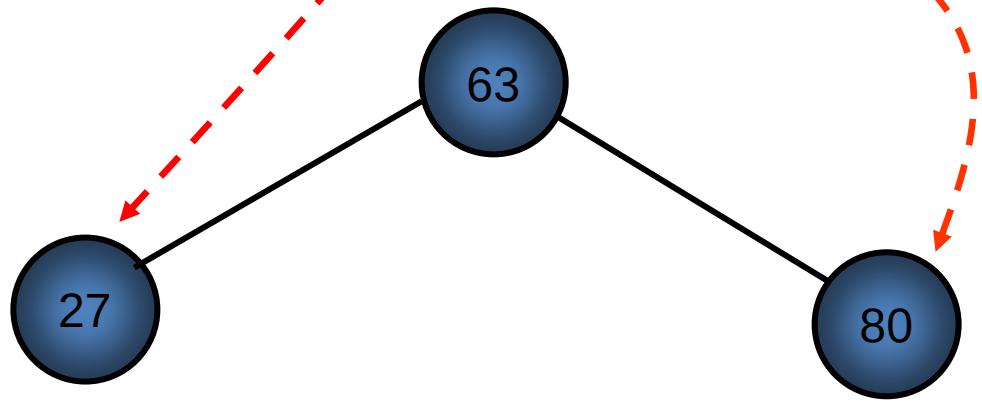
Traversing A Binary Tree

- Visit each node in a specific order.
- Usually nodes would be visited in ascending order based on their key value.
- A recursive method is declared. The method has a single node as an argument.
- When first invoked the method is passed the root node.
- The method
  1. Calls itself passing the left node as an argument.
  2. Visits the node
  3. Calls itself passing the right node as an argument

Traversing Code

```
private void DisplayInOrder(Node localRoot) {
        if (localRoot != null) {
            DisplayInOrder(localRoot.LeftChild);
            localRoot.Print();
            DisplayInOrder(localRoot.RightChild);
        }
}
```

DisplayInOrder(Root);

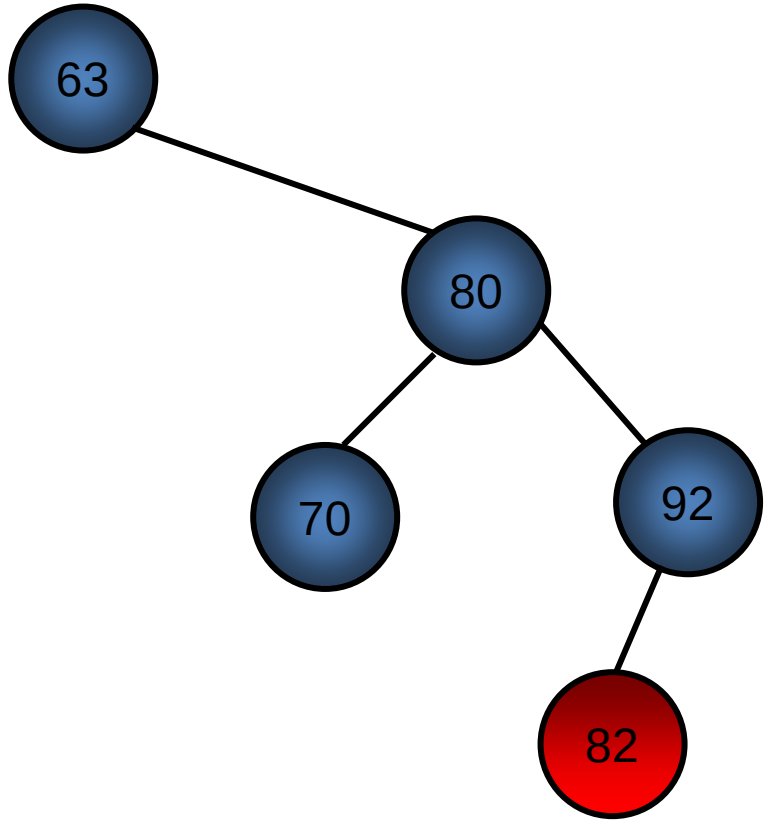


Deleting A Node

- Find the node to be deleted using the logic within the search method.
- Three cases to consider
  1. The node is a leaf
  2. The node has one child
  3. The node has two children

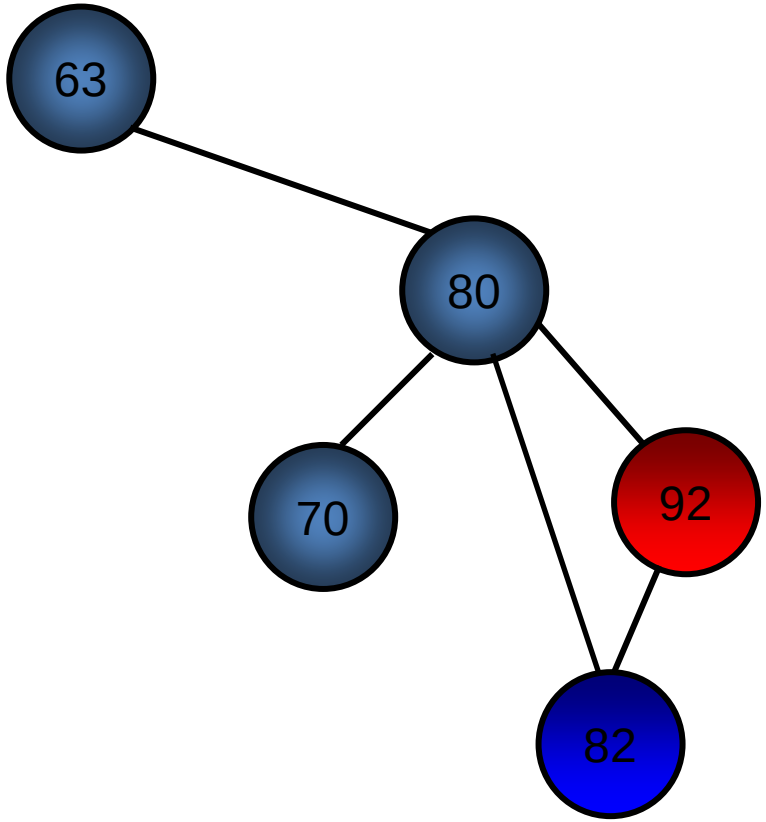
Delete A Leaf Node

- Set the parents reference to the deleted node to null.
- The node itself will be removed when garbage collection is activated.



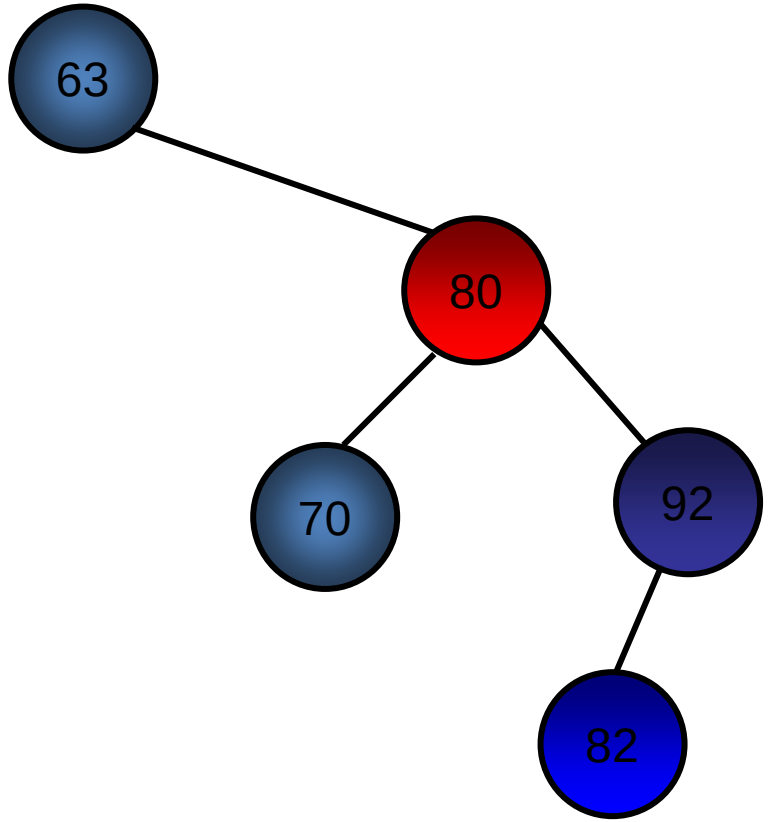
Delete A Node With One Child

- Connect the parent directly to the only child.
- Change the parent's reference to that of the child.

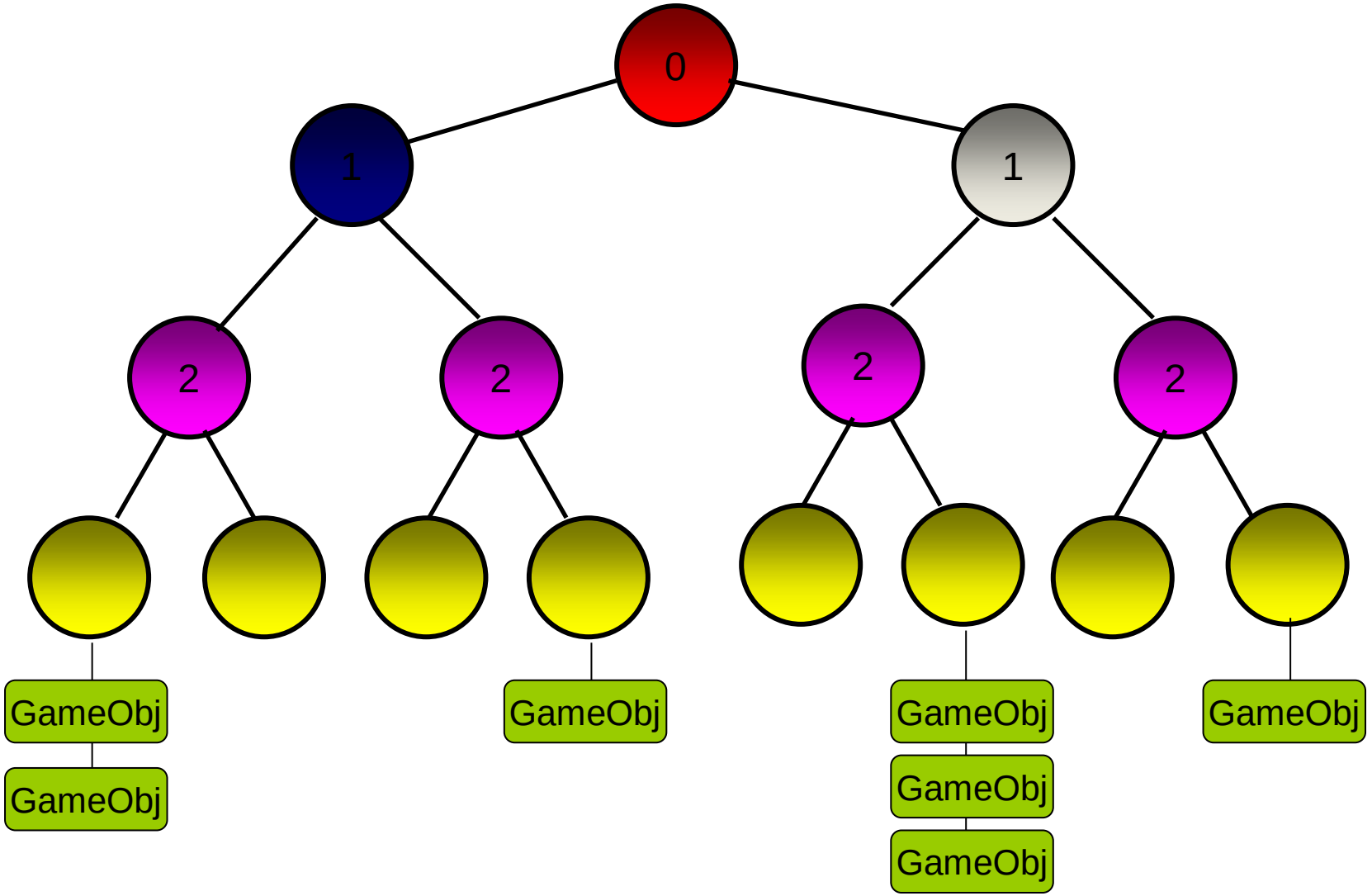


Delete A Node With Two Children

- Replace the node with its in order successor.
- The successor is the node with the next largest key value.



BSP Tree Example



Performance

- If the order of insertion and deletion is comparatively random, a binary search tree may remain fairly balanced, such that average height over all inputs of size *n* is O(log *n*)
- If the order of insertion is fully ordered, then the tree may degenerate into looking like a linear structure of O(*n*)
- The answer may be to have a self balancing tree (AVL Tree)
- A binary search tree is said to be *AVL balanced* if , for every node, the heights of its two sub trees differ at most by one

Activity

- Attempt Exercises 3, 4 and 5 to implement the insert, find and delete algorithms.