CSE 105:
Data Structures and Algorithms-I
(Part 2)

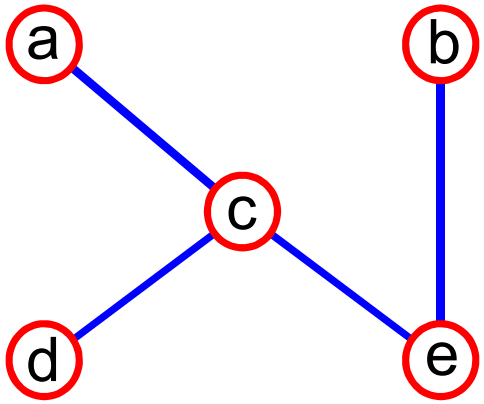
Instructor

Dr Md Monirul Islam

Graphs and Trees: Representation and Search

Tree: Definition

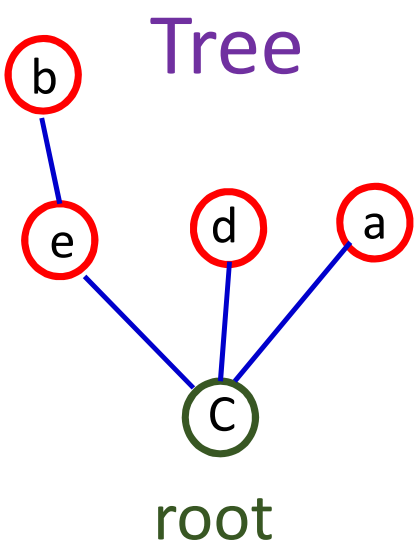
- A tree is a special type of graph!
- A tree is a graph that is connected and acyclic.
- A tree consists of one or more nodes
- a free tree has NO special node.

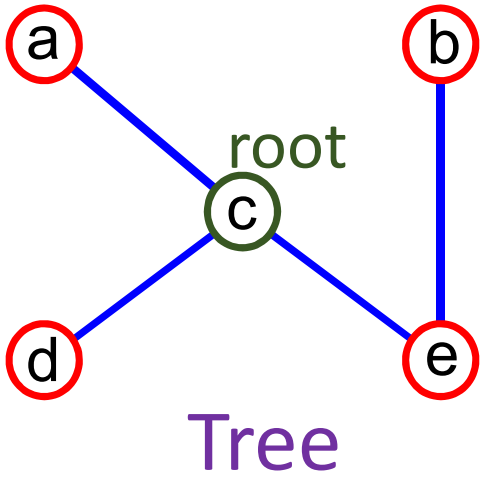


Tree

Rooted Tree: Definition

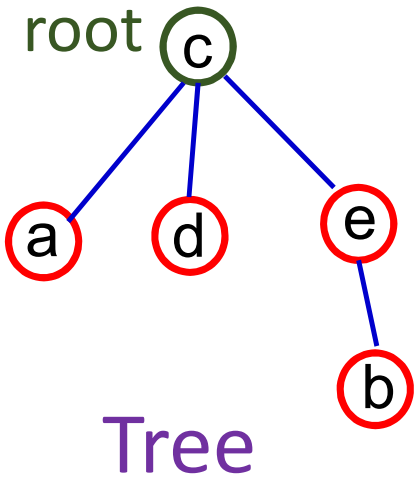
- a rooted tree has a special node, e.g., first node or root.
- every node has a parent except the root
- every node has zero or more children

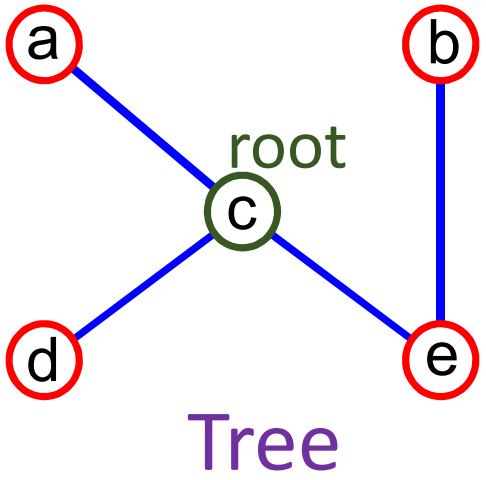




Rooted Tree: Definition

- a rooted tree has a special node, e.g., first node or root.
- every node has a parent except the root
- every node has zero or more children





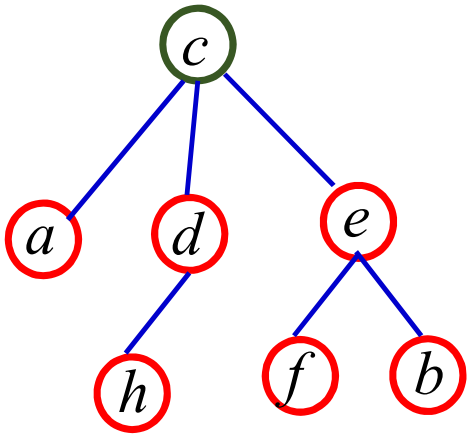
Rooted Tree: Definition

degree of a node:

No. of children of a node

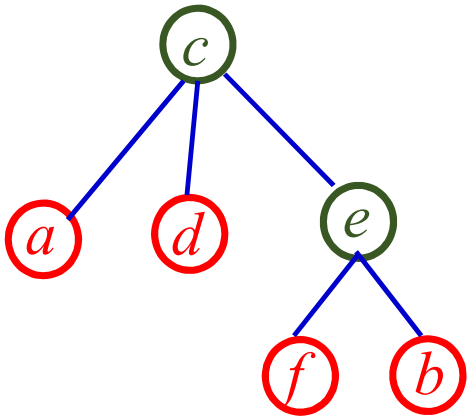
degree (c) =3, degree (e) =2 , degree (d) =1

others have degree 0



Rooted Tree: Definition

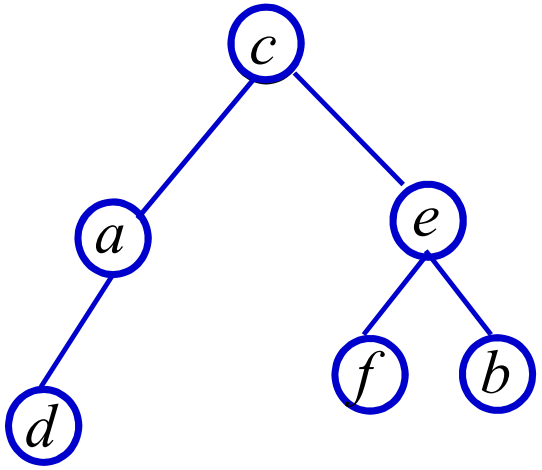
- Leaf nodes: nodes with degree 0: *a, d, f, b*
- Internal nodes: other nodes : *c, e*



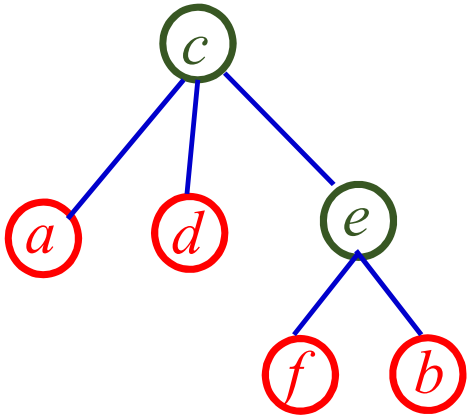
Binary Tree: Definition

all nodes have at most 2 children

Binary tree



Binary tree

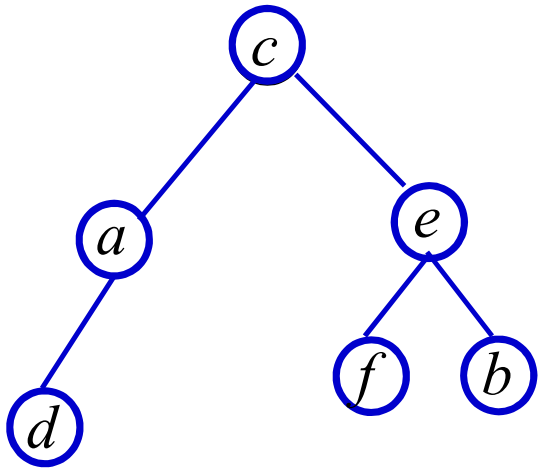


Non-binary tree

Binary Tree: Definition

- A binary tree is made up of a finite set of nodes
- This set either is empty or consists of a node called the root together with two binary trees, called the left and right subtrees
- Subtrees are disjoint from each other and from the root

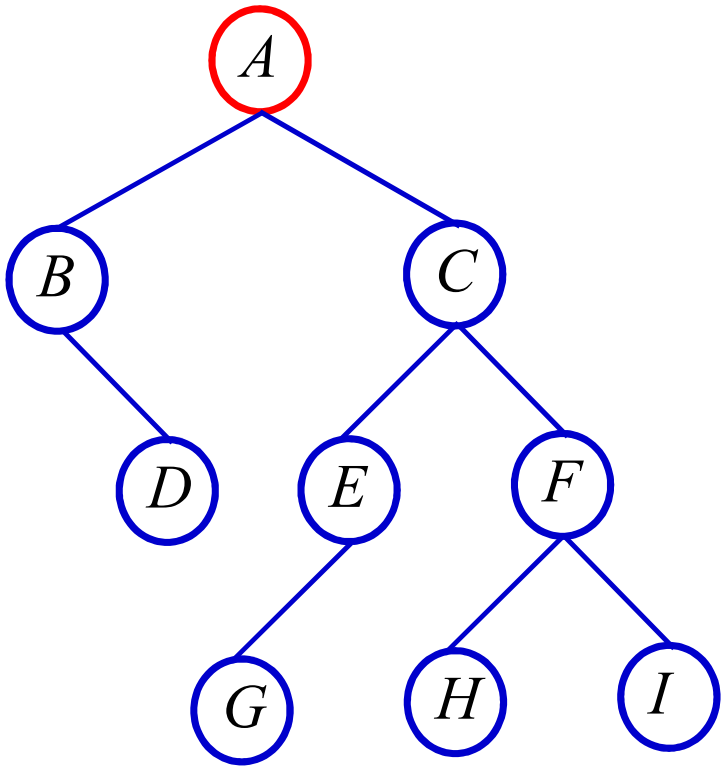
Binary tree



Binary tree

Binary Tree: Elements

One or more Nodes, with a special node called root



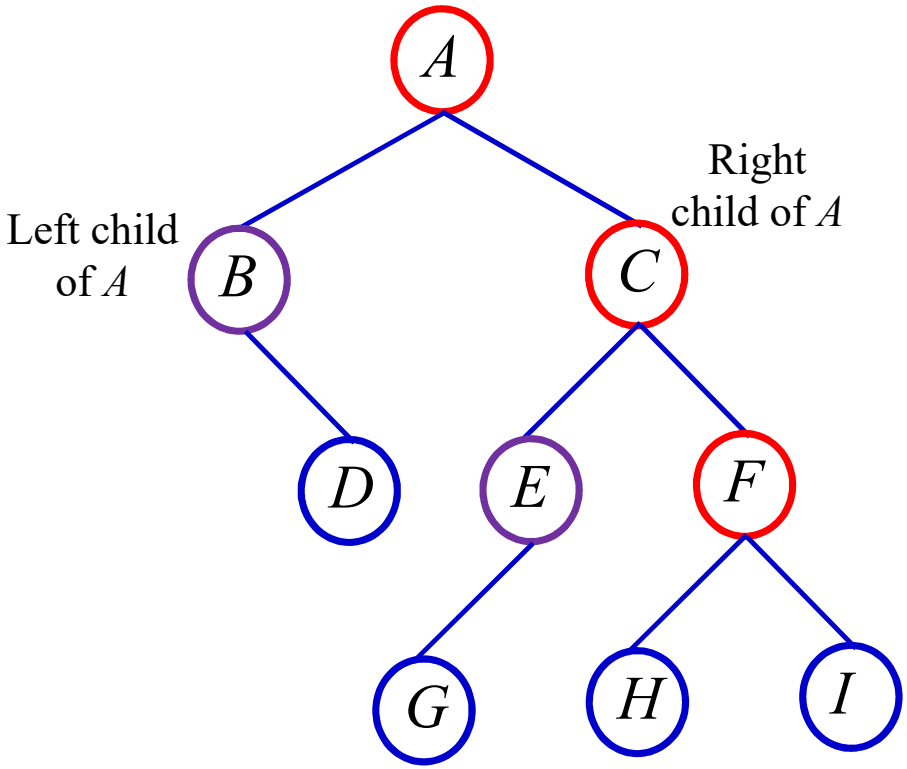
Binary Tree: Elements

Children:

Every node has 0, 1 or 2 children

Leaf nodes: have no child

Internal nodes: have 1 or 2 children



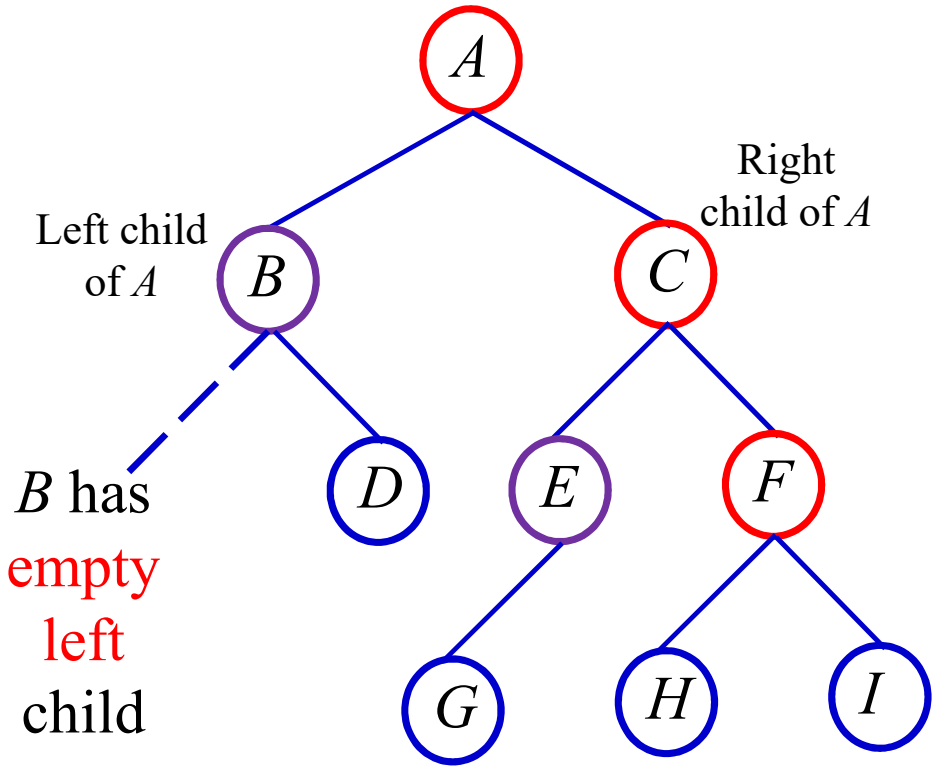
Binary Tree: Elements

Children:

Every node has 0, 1 or 2 children

Leaf nodes: have no child

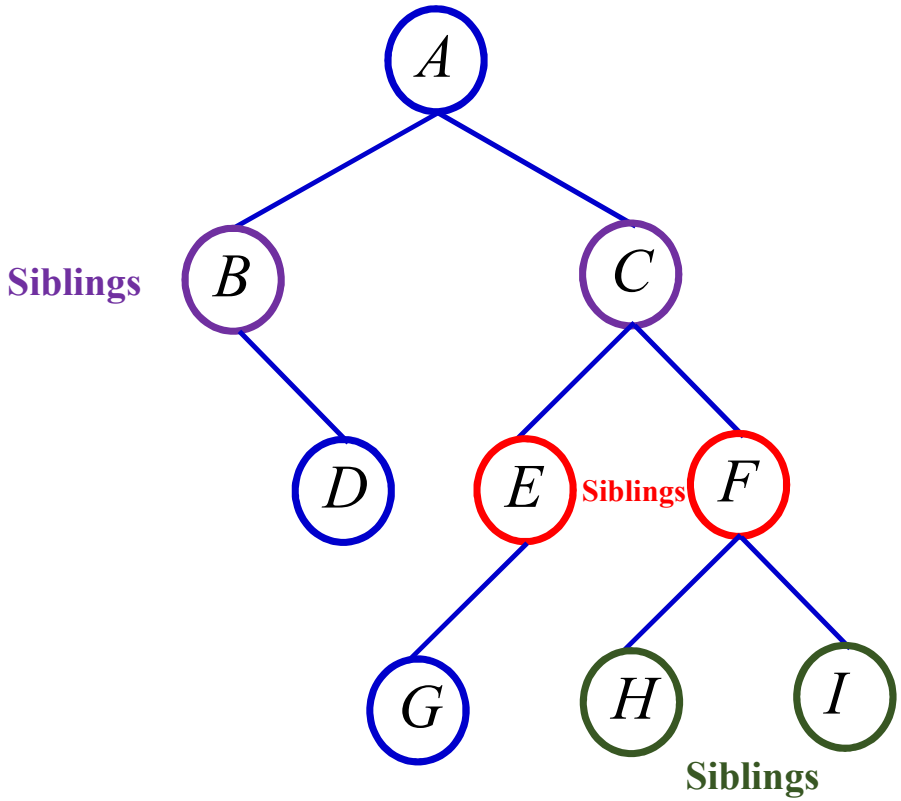
Internal nodes: have 1 or 2 children



Binary Tree: Elements

Siblings:

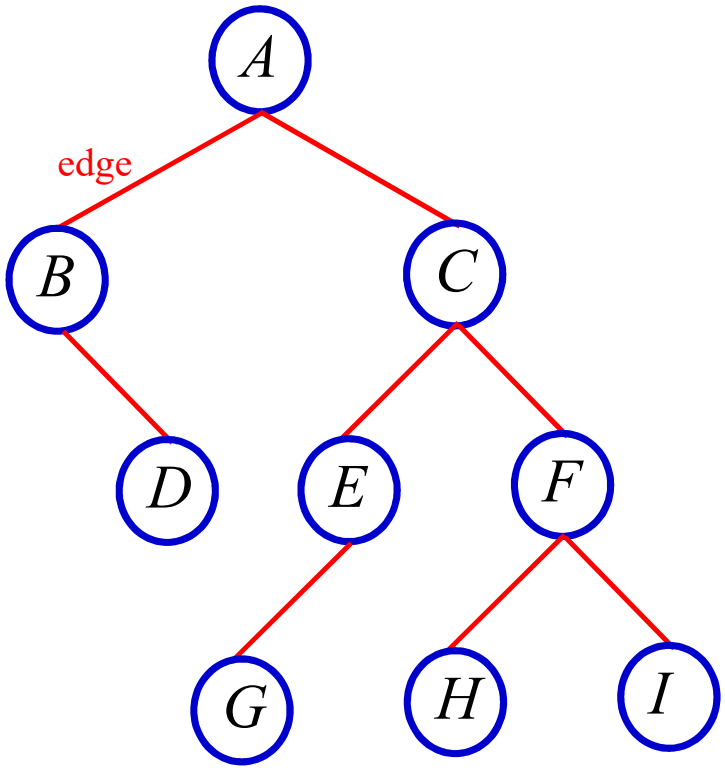
Immediate children of same parent



Binary Tree: Elements

Edge:

Each node is connected to each of its children by an edge

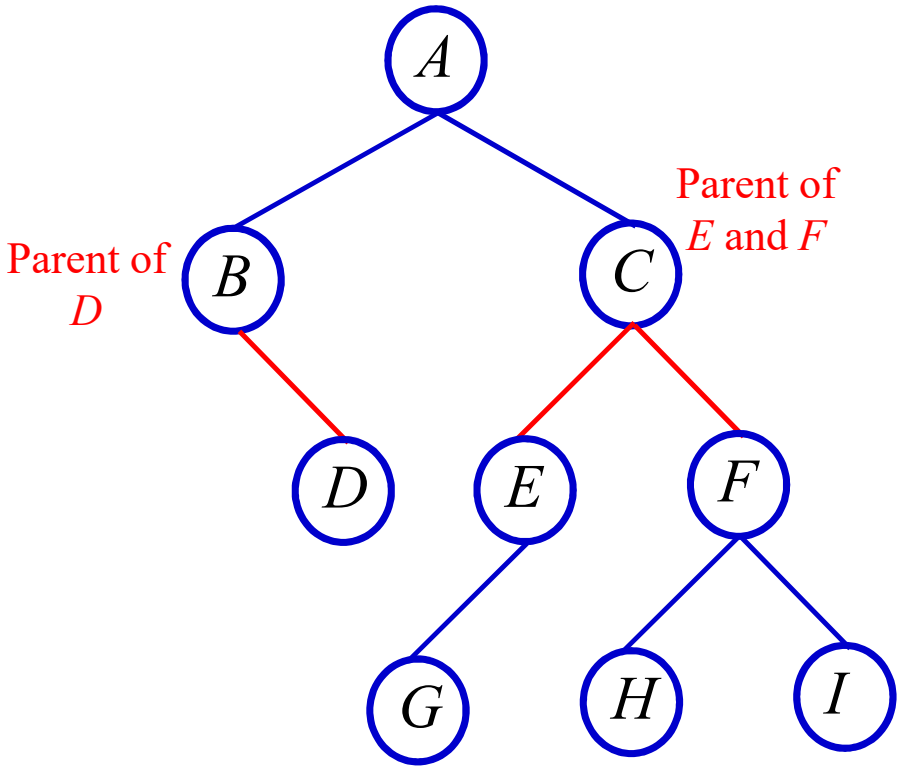


Binary Tree: Elements

Parent:

Root has no parent.

Others have parent



Binary Tree: Elements

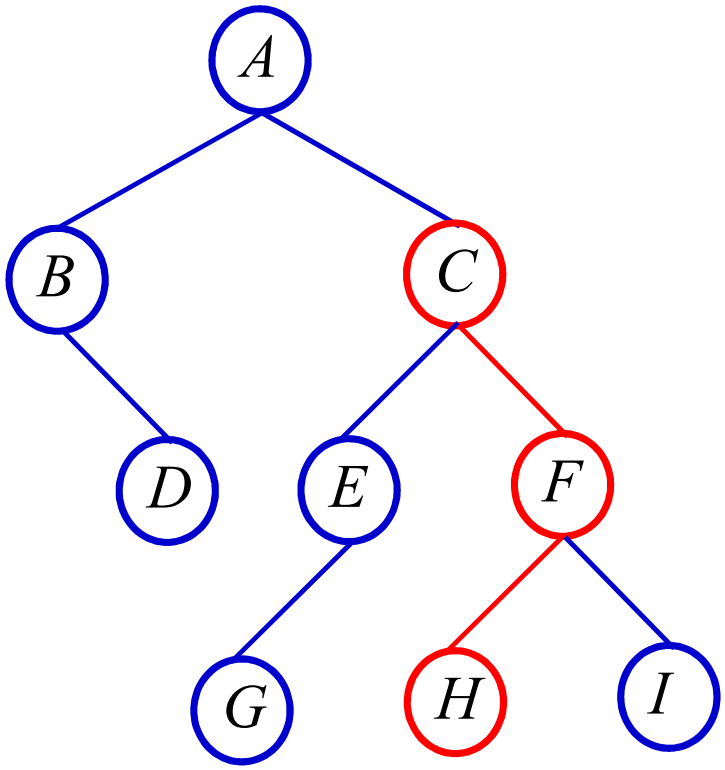
Path:

If $n_1, n_2, n_3, ..., n_k$ is a sequence of nodes

where n_i is the parent of n_{i+1} for $1 \leq i < k$,
then this sequence is a path

Each path has a length (No. of edges)

Length of path *C-F-H* is 2



Binary Tree: Elements

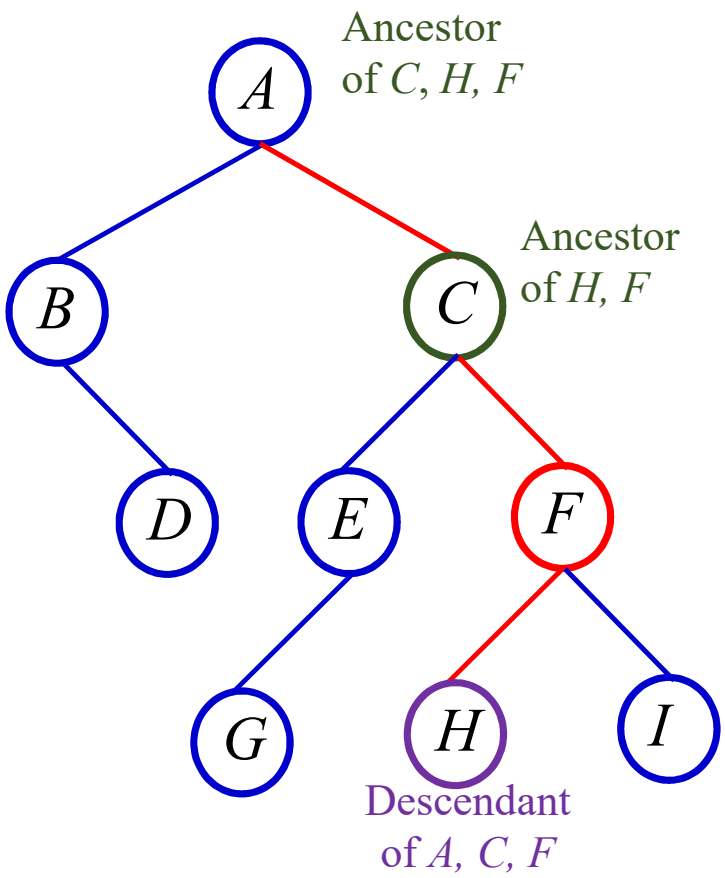
Ancestor/Descendant:

If there is a path from n_i to parent of n_{i+1}

Then,

n_i is a ancestor of n_{i+1}

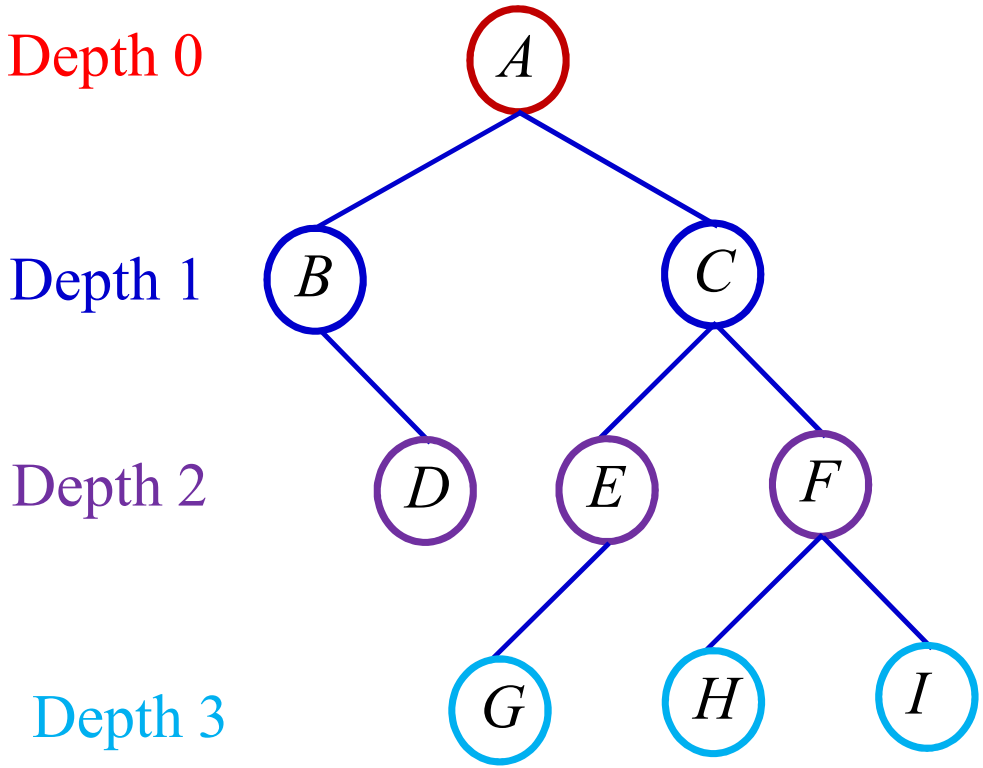
n_{i+1} is a descendant of n_i



Binary Tree: Elements

Depth of a node (u):

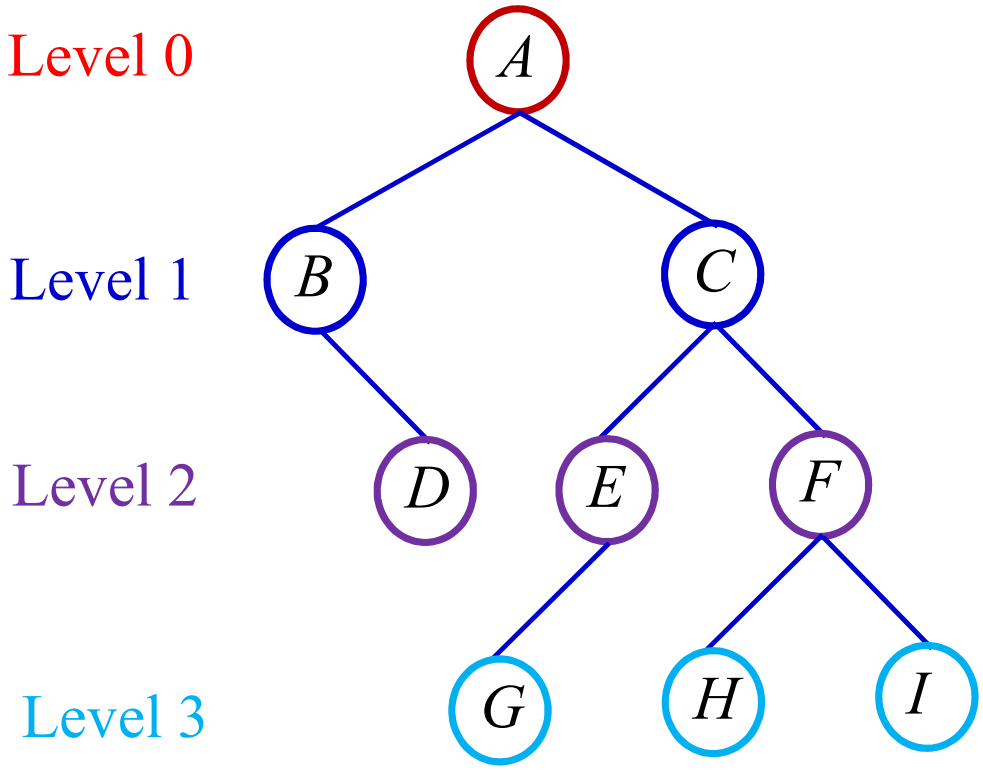
Length of path from root to node u



Binary Tree: Elements

Level of a node:

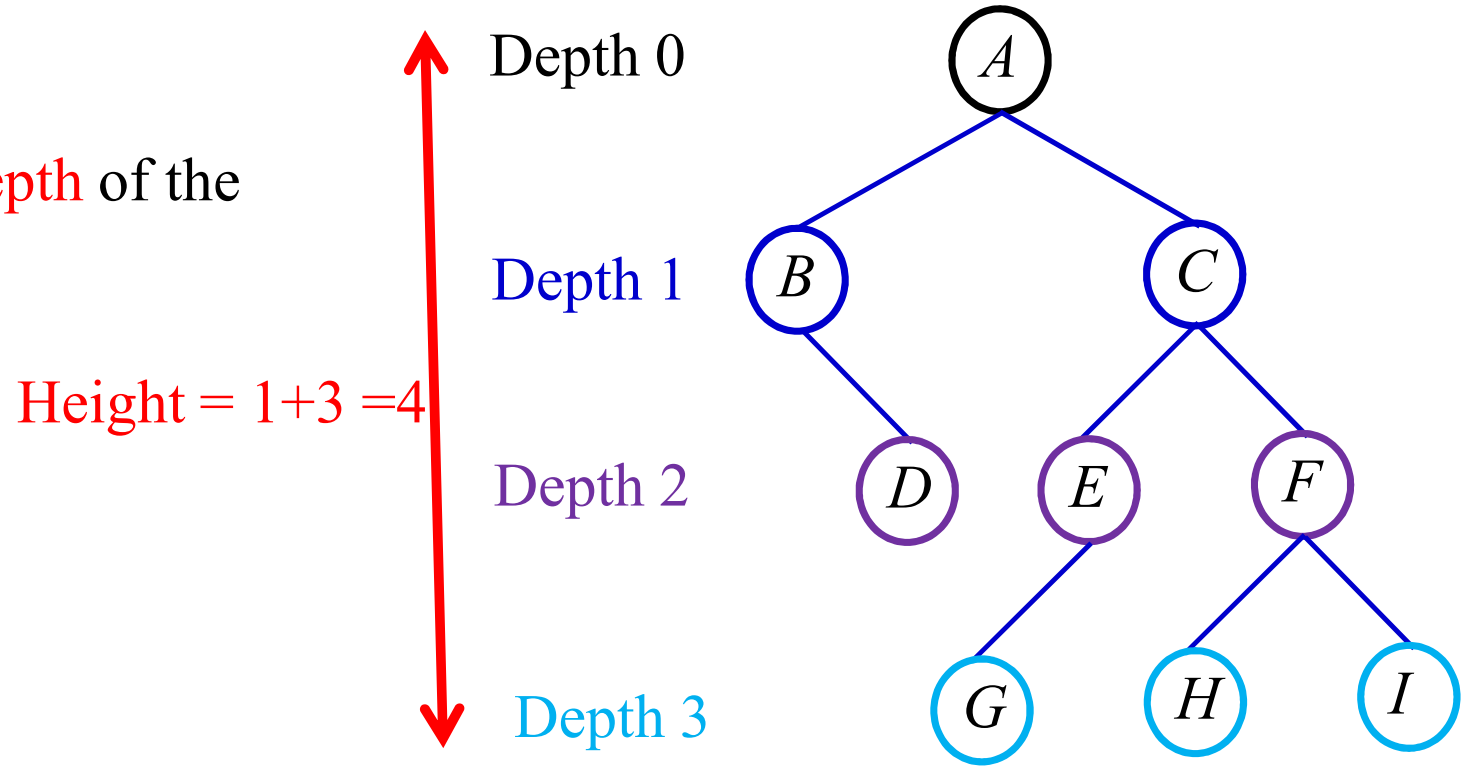
All nodes of depth *n* are at level *n*



Binary Tree: Elements

Height of a tree:

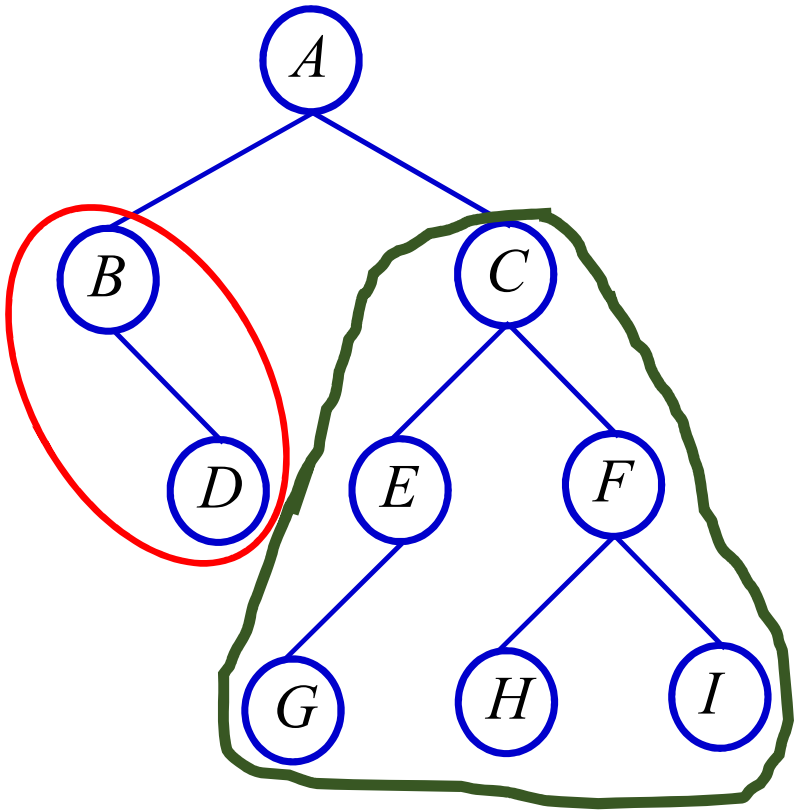
One more than the depth of the deepest node



Binary Tree: Elements

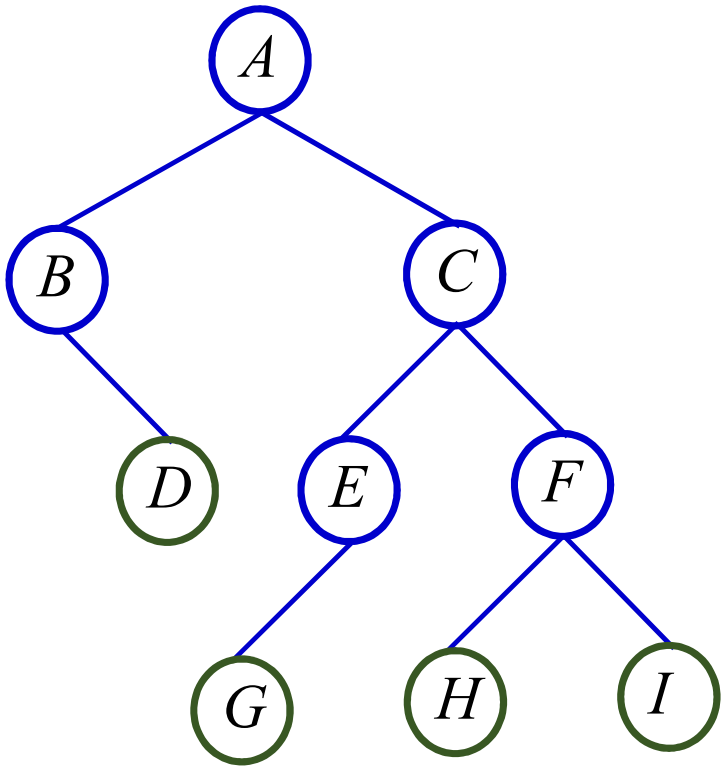
Subtree:

Subset of a tree; **Left** and **right** subtrees



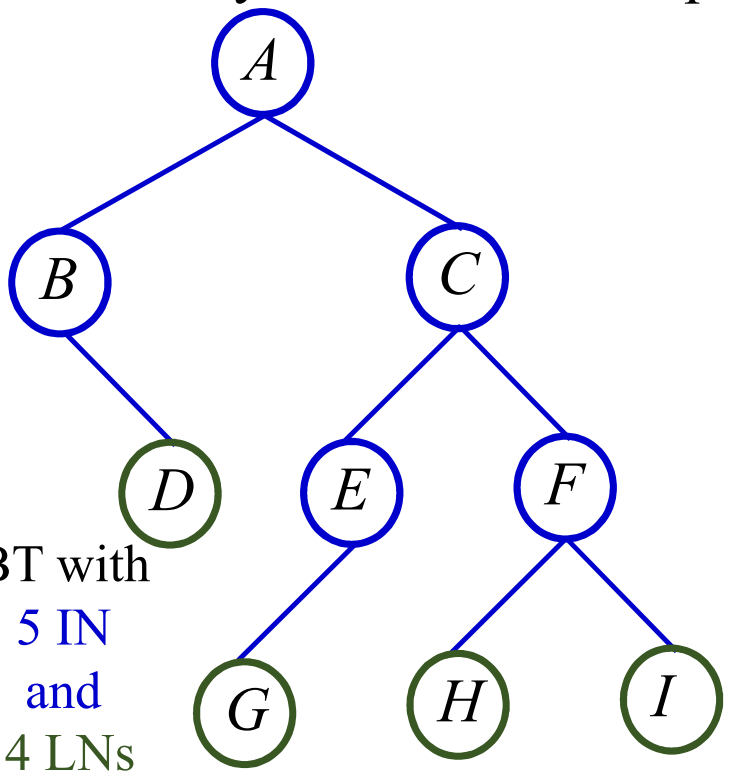
Information storage in Binary Trees

- Some implementations store data
  - at all nodes
  - only at the leaf nodes, using the internal nodes to provide structure to the tree.
- Space requirements may vary for
  - internal nodes
  - leaf nodes.

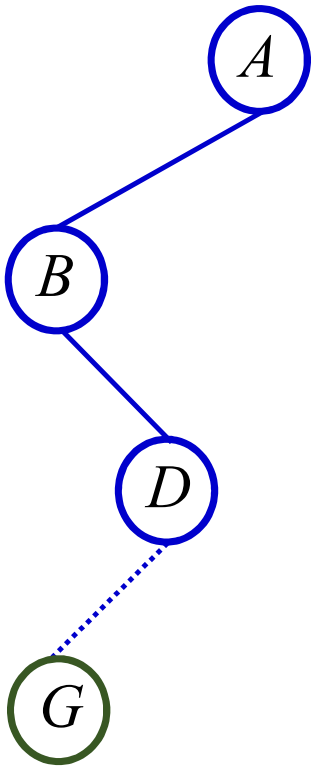


Information storage in Binary Trees

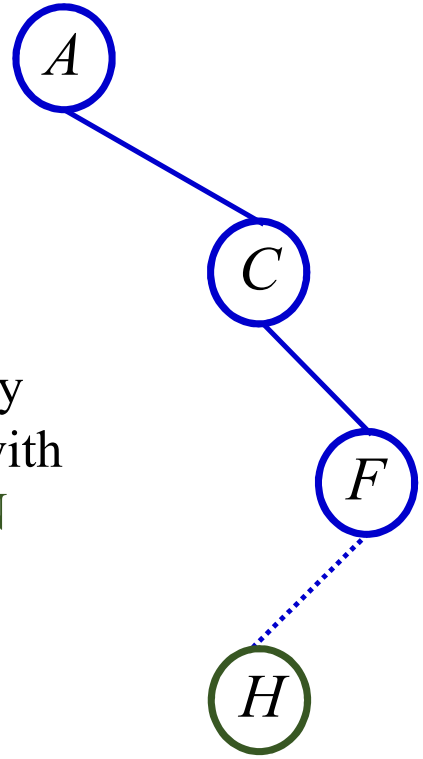
The number of internal nodes and leaf nodes vary based on the shape of the tree.



BT with 5 IN and 4 LNs

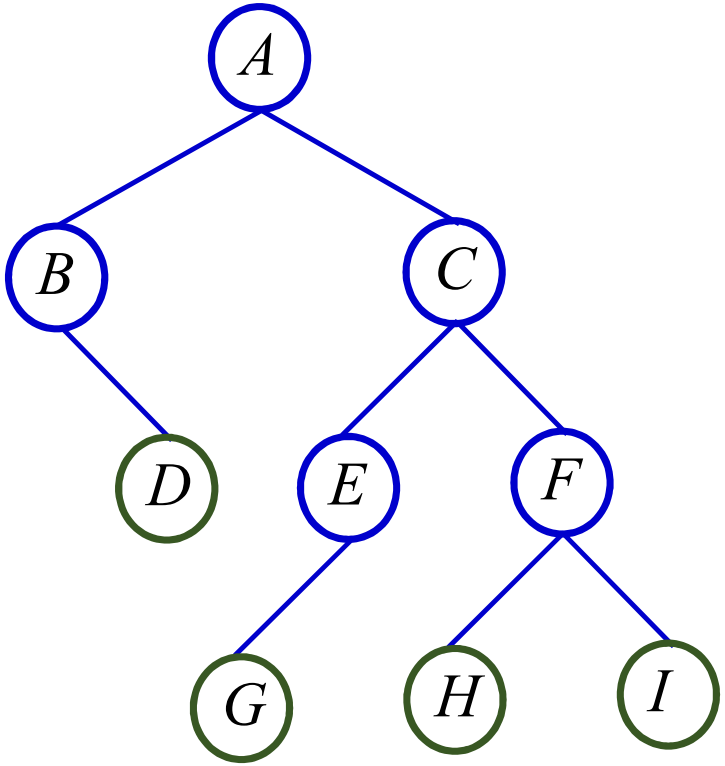


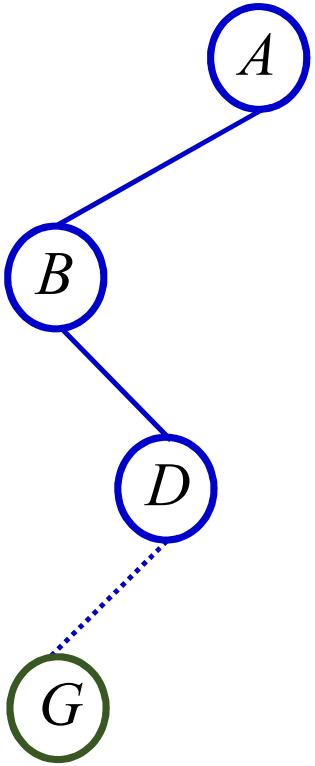
Binary Trees with **1 LN**

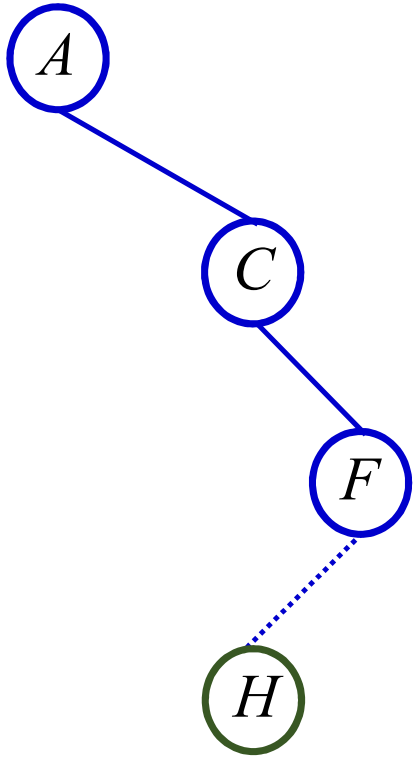


Information storage in Binary Trees

For space analysis and other requirements
special class of binary trees are used



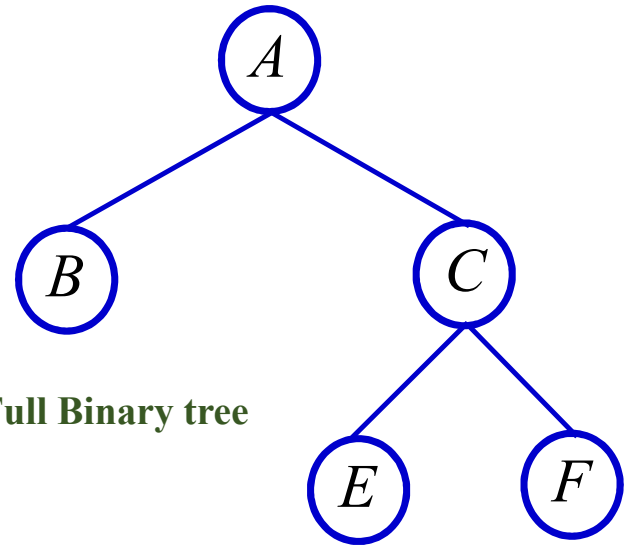




Full and Complete Binary Trees

Full Binary tree

Each node is either a leaf or internal node with exactly two non-empty children.

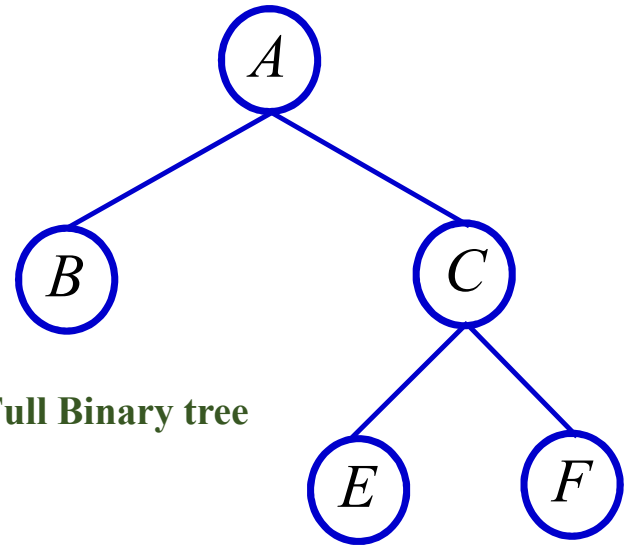


Full Binary tree

Full and Complete Binary Trees

Full Binary tree

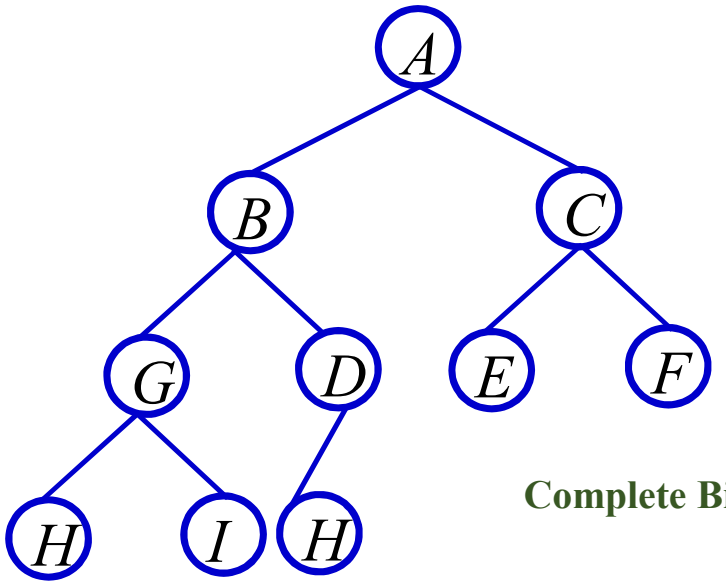
Each node is either a leaf or internal node with exactly two non-empty children.



Full Binary tree

Complete Binary Tree

If the tree has *d* levels, then all levels except possibly level *d*-1 are completely full. The bottom level has all nodes starting from the left side.

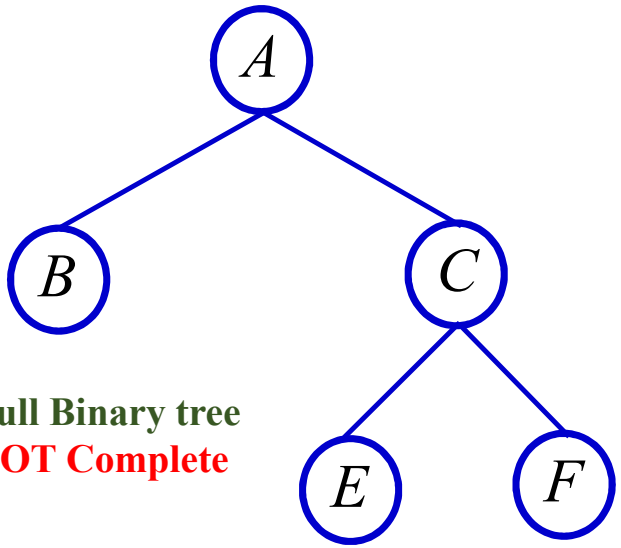


Complete Binary tree

Full and Complete Binary Trees

Full Binary tree

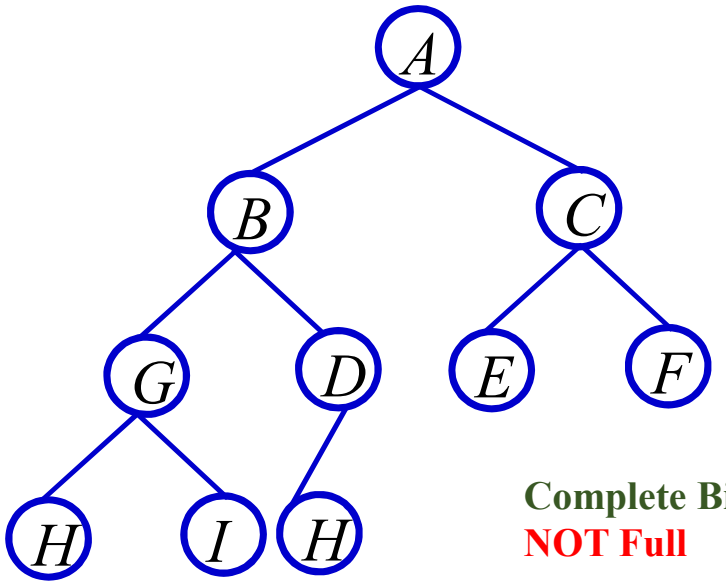
Each node is either a leaf or internal node with exactly two non-empty children.



Full Binary tree
NOT Complete

Complete Binary Tree

If the tree has *d* levels, then all levels except possibly level *d*-1 are completely full. The bottom level has all nodes starting from the left side.

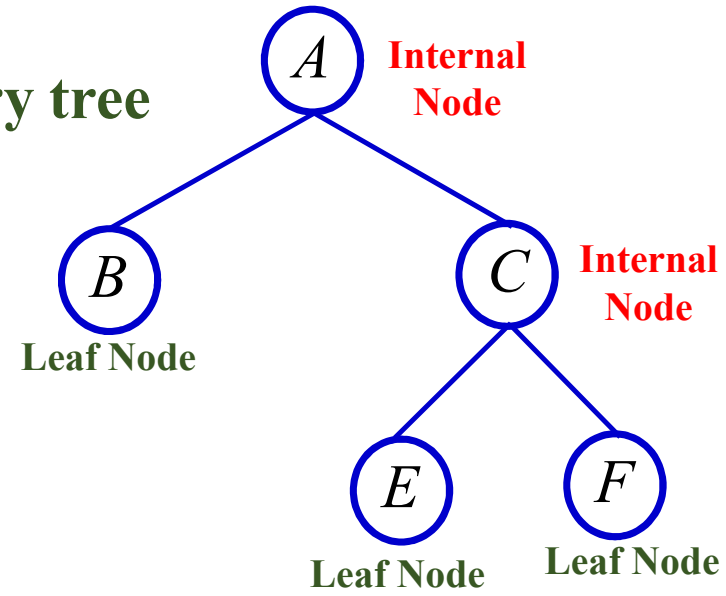


Complete Binary tree
NOT Full

Full Binary Tree Theorem

Theorem: The number of leaves in a non-empty full binary tree is one more than the number of internal nodes.

Full Binary tree



LN = 1 + IN

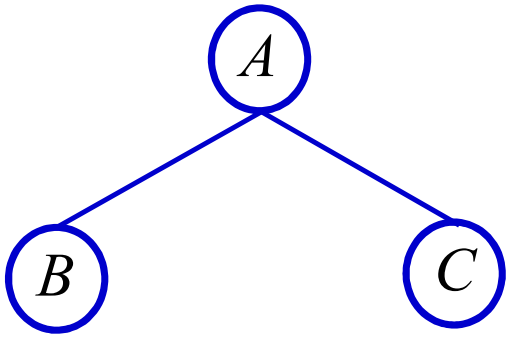
Full Binary Tree Theorem

Theorem: The number of leaves in a non-empty full binary tree is one more than the number of internal nodes.

Proof (by Mathematical Induction):

Base case: A full binary tree with 1 internal node must have two leaf nodes.

Induction Hypothesis: Assume any full binary tree **T** containing n-1 internal nodes has n leaves.



Full Binary Tree Theorem

Induction Step:

We want to prove the following here:

If n-1 internal nodes mean n leaves

Then n internal nodes would mean n+1 leaves

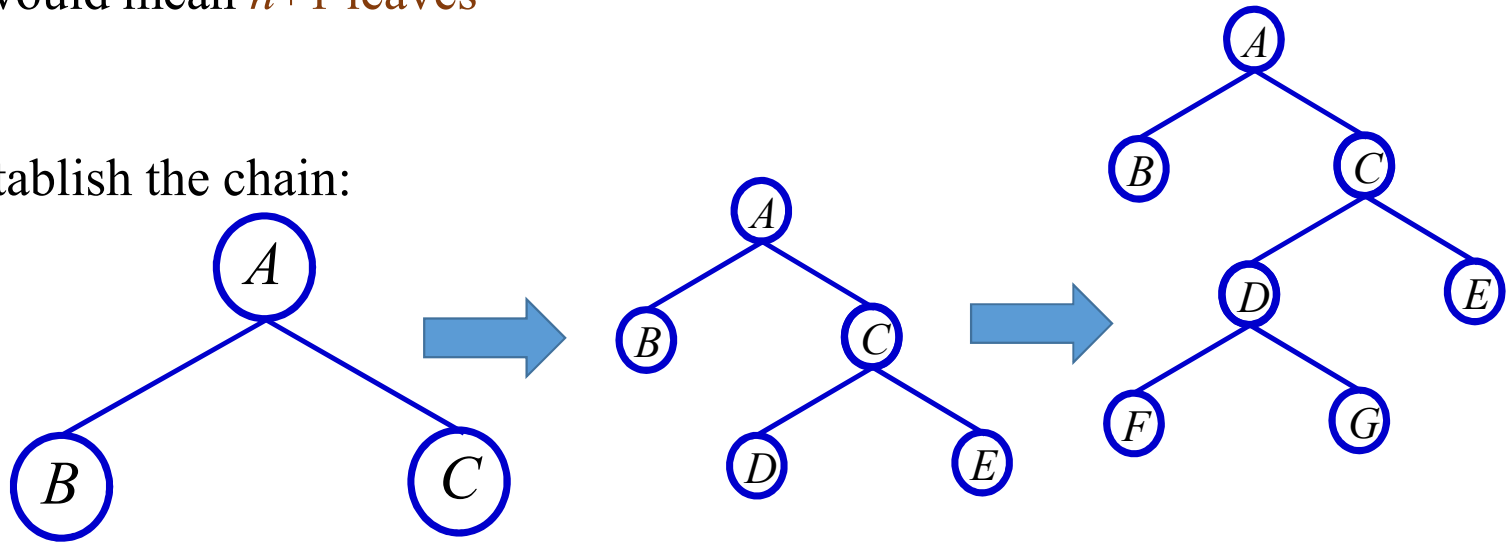
[Recall that this will establish the chain:

1 IN mean 2 L

So, 2 IN mean 3 L

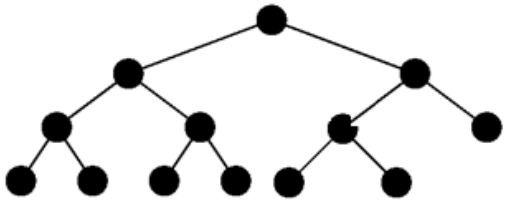
So, 3 IN mean 4

And so on…]



Full Binary Tree Theorem

Induction Step: Given tree **T** with n internal nodes

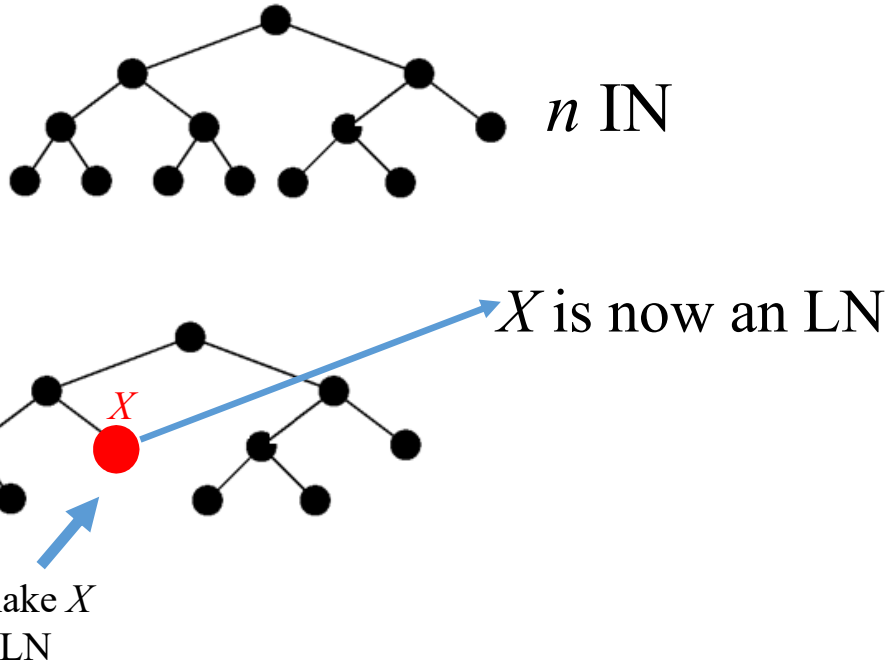


n IN

Full Binary Tree Theorem

Induction Step: Given tree **T** with n internal nodes,

pick internal node X with two leaf children. Remove X's children, call resulting tree **T′**.

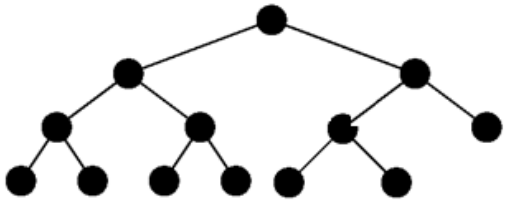


Full Binary Tree Theorem

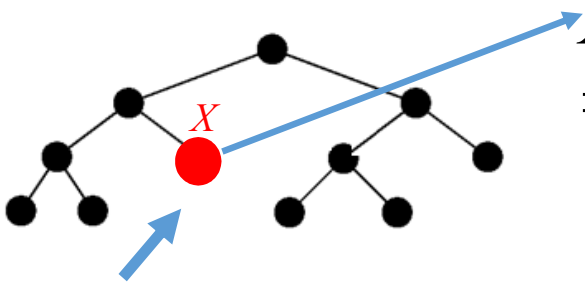
Induction Step: Given tree **T** with n internal nodes,

pick internal node X with two leaf children. Remove X's children, call resulting tree **T′**.

Clearly, **T′** is a full binary tree with n-1 internal nodes



n IN



X is now an LN

⇒now, n-1 INs

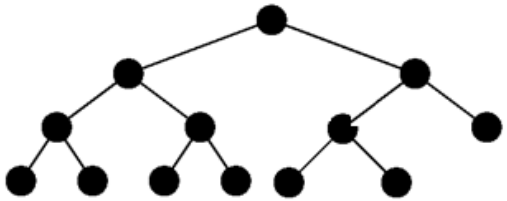
make X a LN

Full Binary Tree Theorem

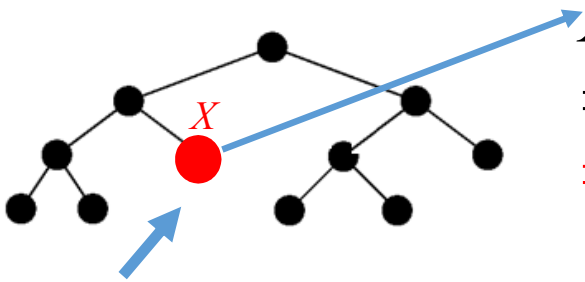
Induction Step:

Clearly, **T′** is a full binary tree with n-1 internal nodes

By induction hypothesis, it has n leaves.



n IN



X is now an LN

⇒now, n-1 INs

⇒n LN (IH)

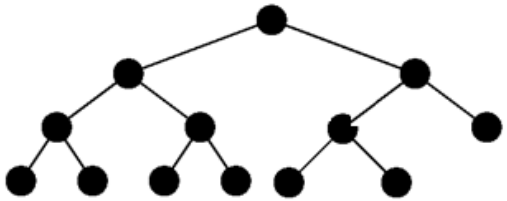
make X a LN

Full Binary Tree Theorem

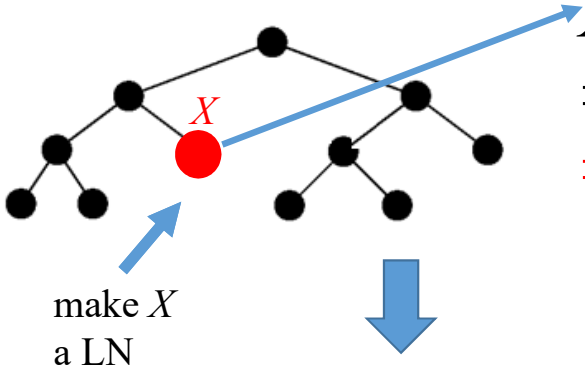
Induction Step:

Now:

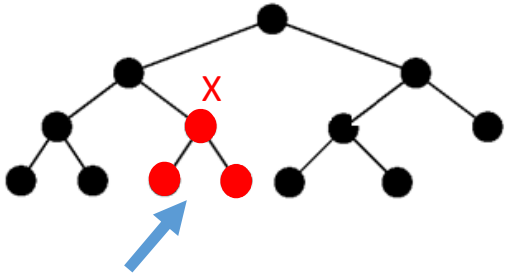
Restore *X*'s two children.
The number of internal nodes has now gone up by 1 to reach *n*.



n IN



X is now an LN
⇒now, *n*-1 INs
⇒*n* LN (IH)



X is now an IN
⇒now, *n* INs

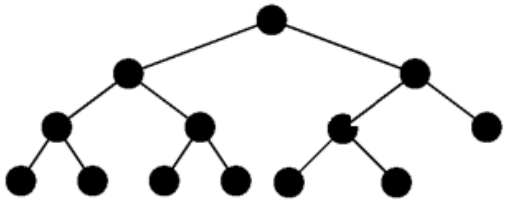
Full Binary Tree Theorem

Induction Step:

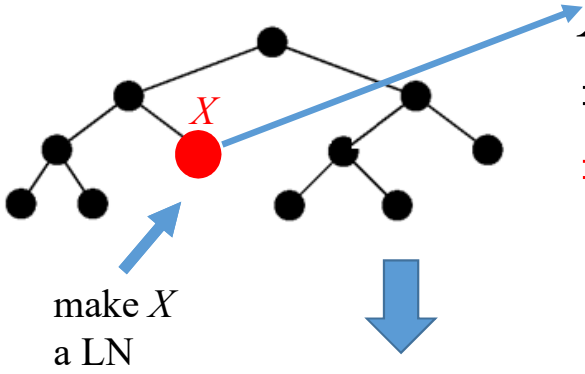
Now:

Restore *X*'s two children. The number of internal nodes has now gone up by 1 to reach *n*.

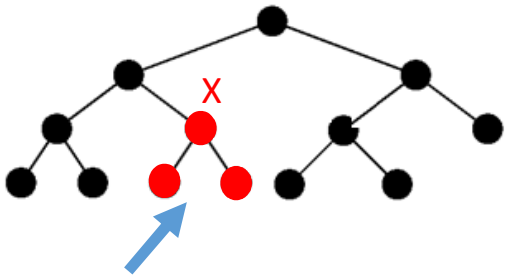
The number of leaves has also gone up by 1.



n IN



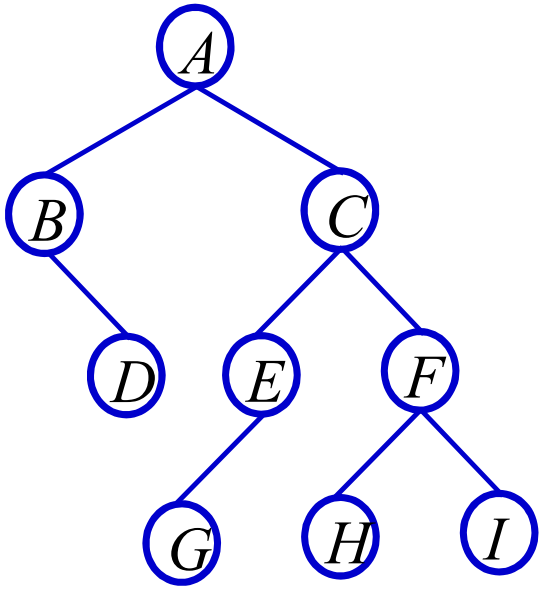
X is now an LN
⇒now, *n*-1 INs
⇒*n* LN (IH)



X is now an IN
⇒now, *n* INs
X: 1 LN became IN
2 LNs added
LN = ***n*** -1 +2 = ***n*** + **1**

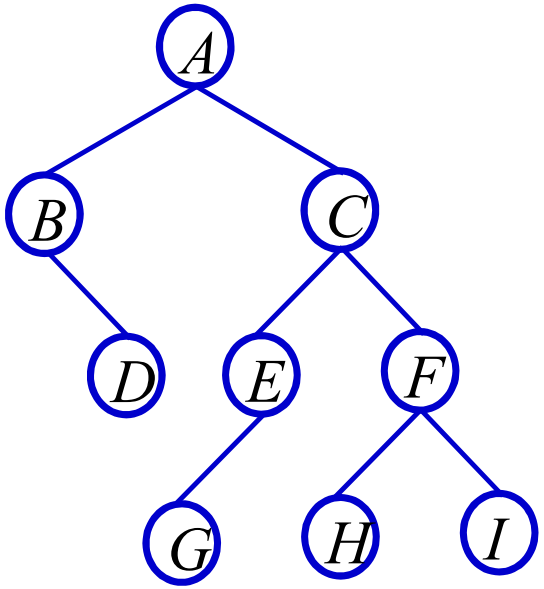
Full Binary Tree Corollary

Theorem: The number of empty subtrees in a non-empty binary tree is one more than the number of nodes in the tree.



Full Binary Tree Corollary

Theorem: The number of empty subtrees in a non-empty binary tree is one more than the number of nodes in the tree.



To understand the theorem assume the following binary tree data structure

```
// binary tree node in C
struct BTnode {
    int data;
    struct BTnode *left, *right;
}
```

Full Binary Tree Corollary

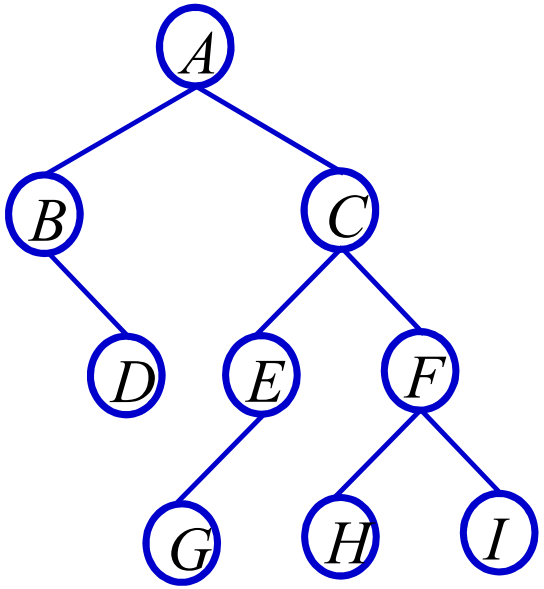
Theorem: The number of empty subtrees in a non-empty binary tree is one more than the number of nodes in the tree.

Single node binary tree

Null pointers or empty subtrees

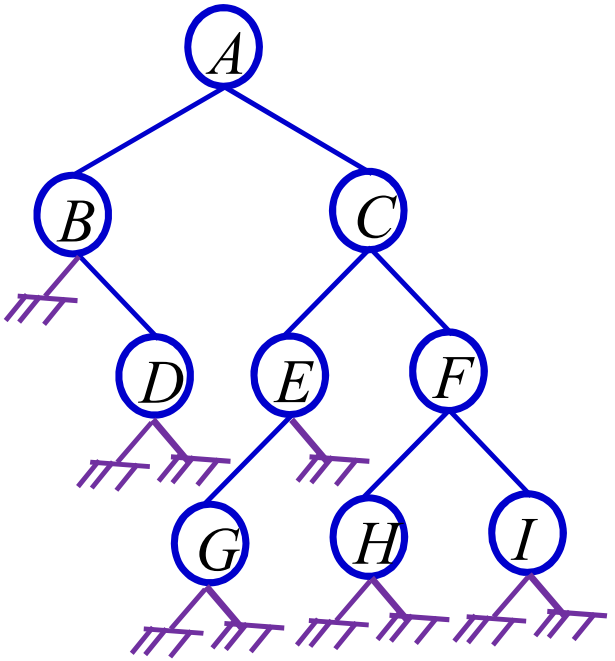
Full Binary Tree Corollary

Theorem: The number of empty subtrees in a non-empty binary tree is one more than the number of nodes in the tree.



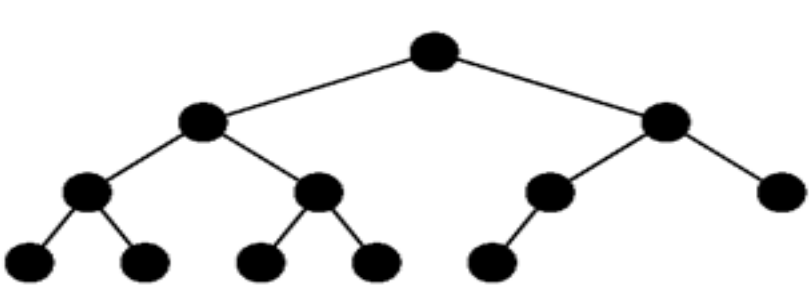
9 Nodes

10 empty subtrees

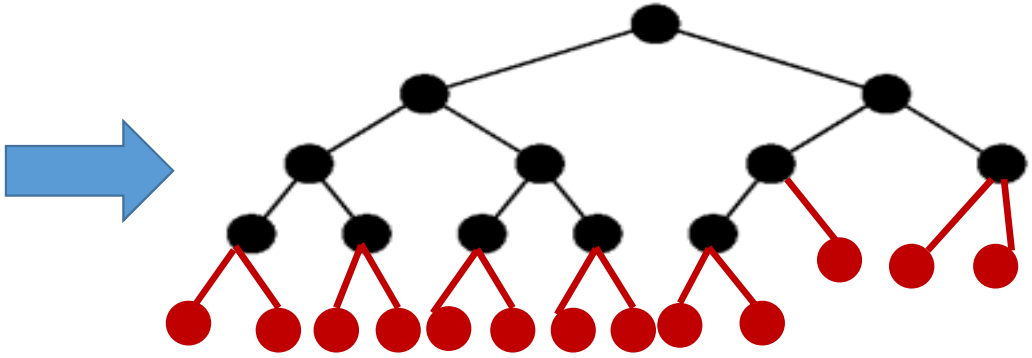


Full Binary Tree Corollary- Proof

- Take an arbitrary binary tree T and replace every empty subtree with a leaf node. Call the new tree T0.



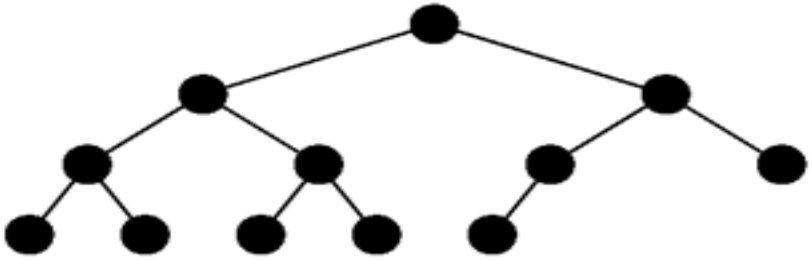
BT: T



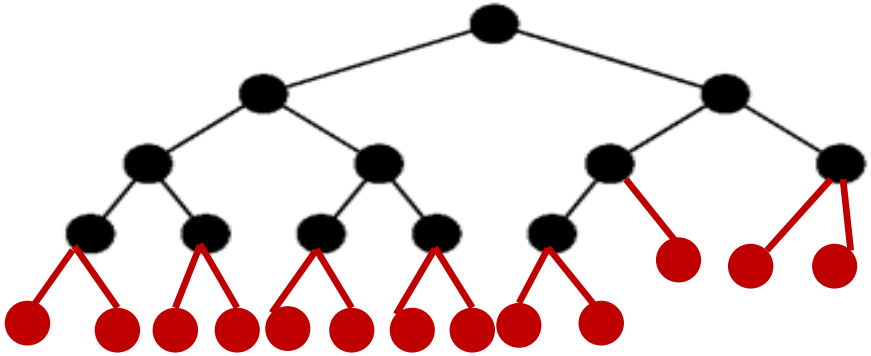
BT: T0

Full Binary Tree Corollary- Proof

All nodes originally in T will be internal nodes in T0



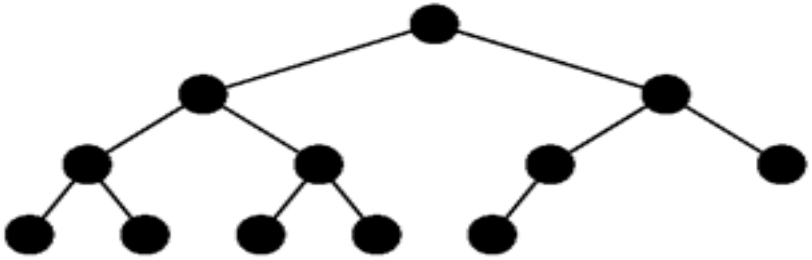
T: Some blacks are internal nodes
Others are leaf nodes



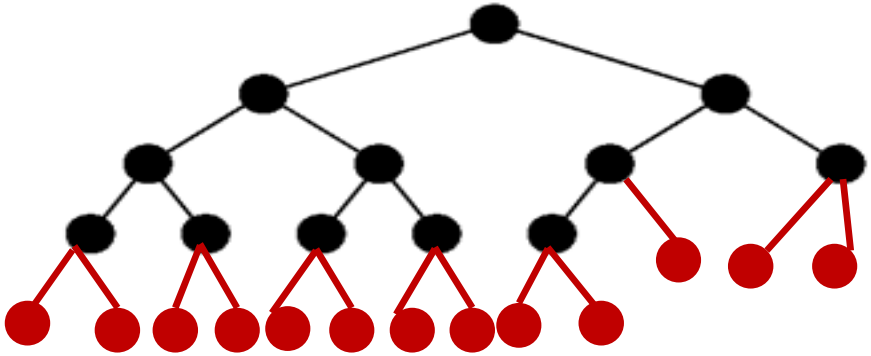
T0: All blacks are internal nodes

Full Binary Tree Corollary- Proof

All IN in T have two children in T0
All LF in T have two children in T0



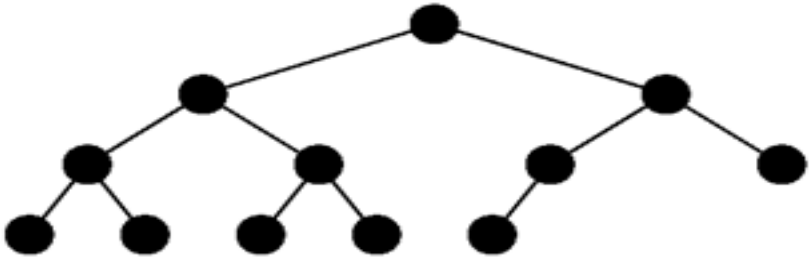
Some blacks are internal nodes
Others are leaf nodes



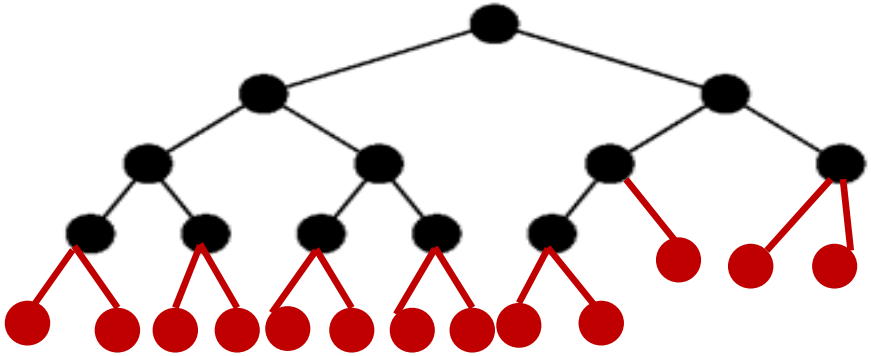
Full Binary tree

Full Binary Tree Corollary- Proof

The Full Binary Tree Theorem tells:
the number of leaves in a full binary tree is one more than the number of internal nodes.



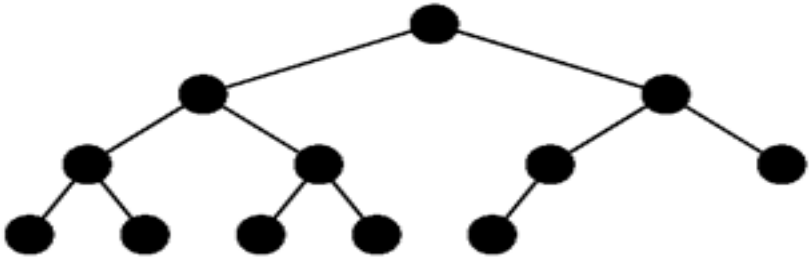
Some blacks are internal nodes
Others are leaf nodes



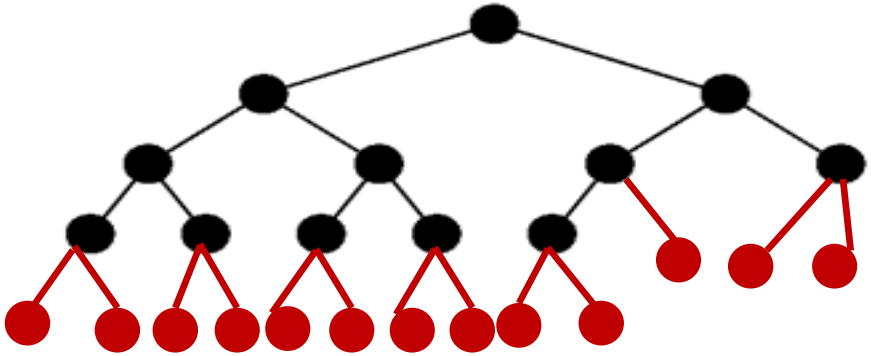
Full Binary Tree Theorem:
LN = 1+ IN

Full Binary Tree Corollary- Proof

leaves in T0 are newly added.
Their number is one more than the nodes of T



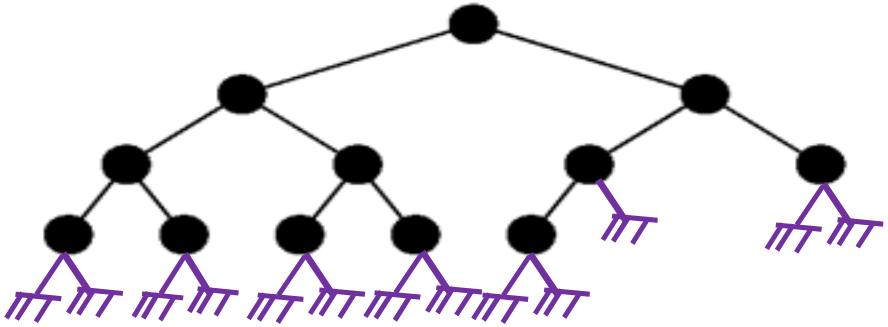
T



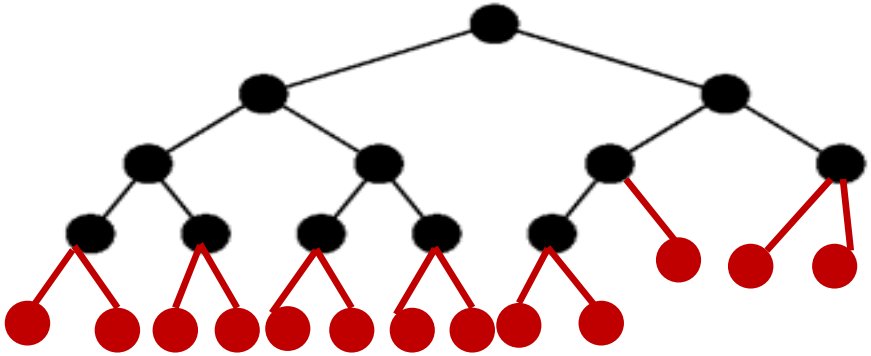
T0: LN = 1+ IN

Full Binary Tree Corollary- Proof

leaves in T0 are empty subtree in T.
Therefore, the number of empty subtrees in T is one more than the number of nodes in T.



T

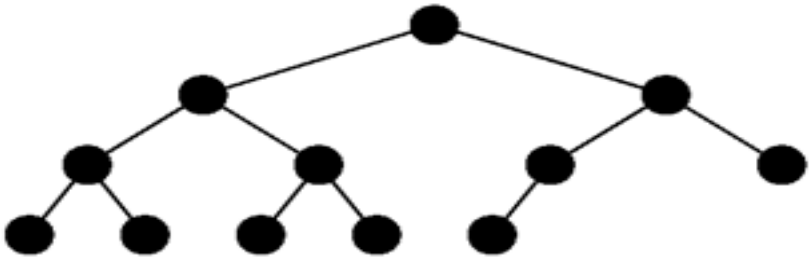


T0: LN = 1+ IN

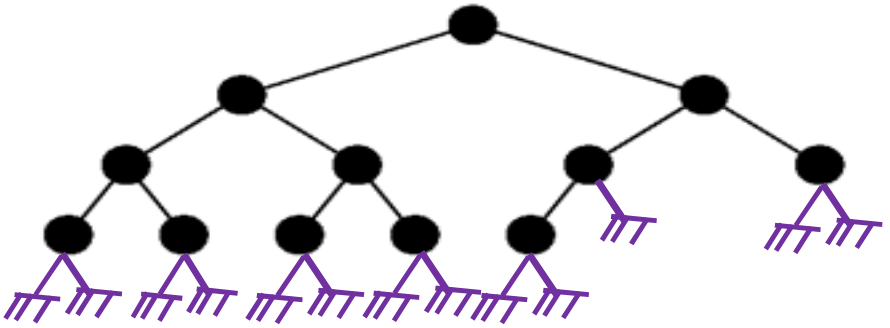
Full Binary Tree Corollary- Proof (2)

Let T have *n* nodes

Every node has two children: empty or non-empty



T: *n* nodes



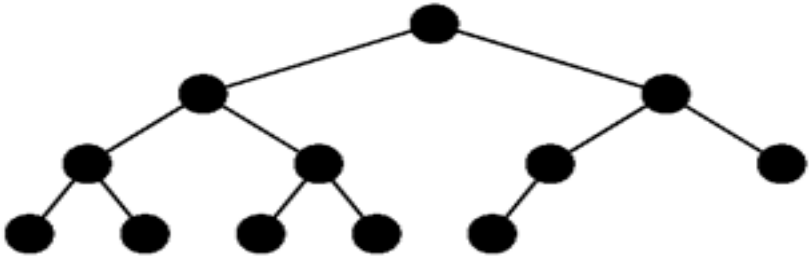
Another view of T

Full Binary Tree Corollary- Proof (2)

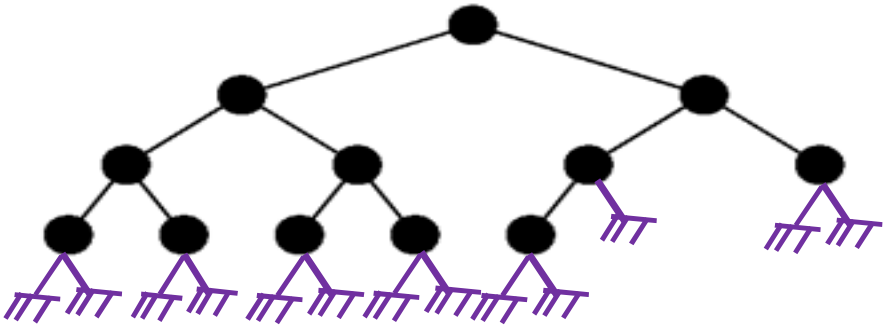
Let T have *n* nodes

Every node has two children: empty or non-empty

Total children :2*n*



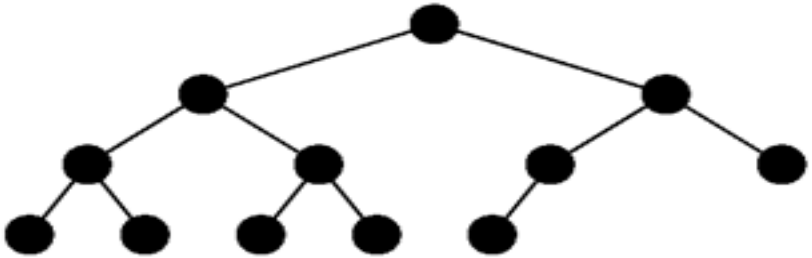
T: *n* nodes



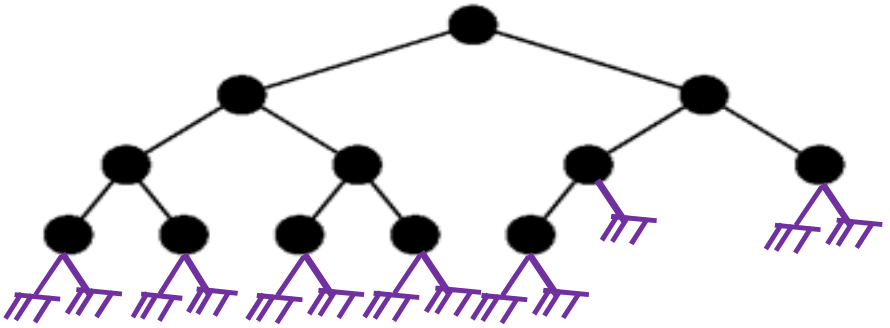
Another view of T: 2*n* Children

Full Binary Tree Corollary- Proof (2)

Except root, every node has parent.
n -1 nodes have parents: they are nonempty children



T: *n* nodes

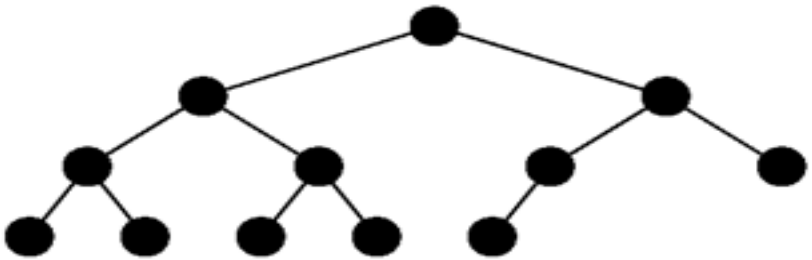


Another view of T: 2*n* Children

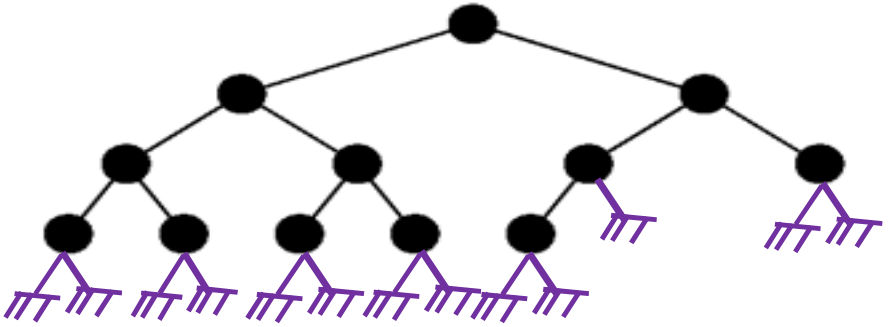
Full Binary Tree Corollary- Proof (2)

Except root, every node has parent.
n -1 nodes have parents: they are nonempty children

Number of empty children (empty subtrees) = $2n - (n\text{-}1) = n + 1$



T: n nodes



Another view of T: $2n$ Children
n+1 empty children