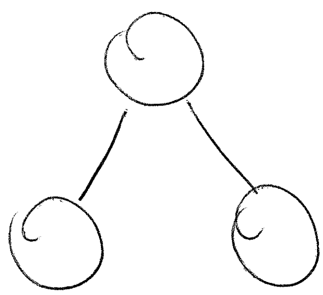
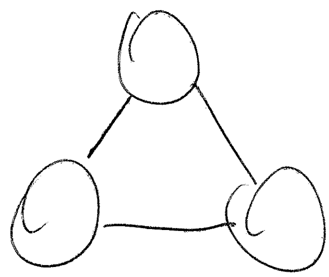
Trees

Vertex/Node + Edges.

Does not have cycles.



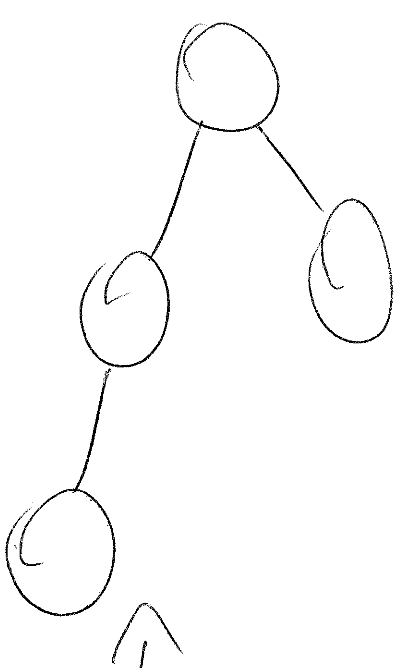
Tree



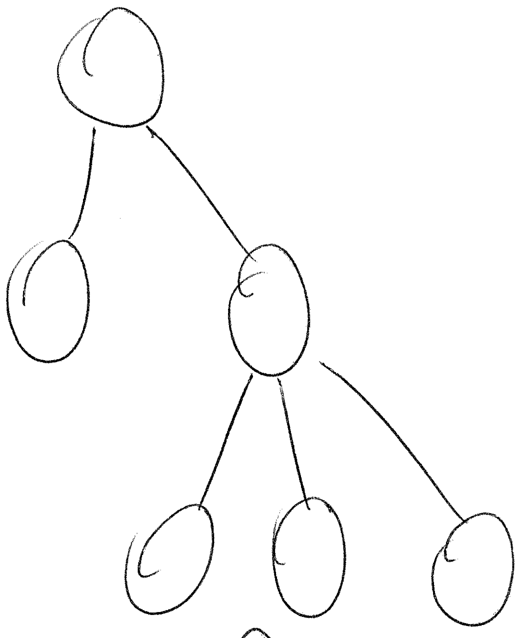
Not a tree

Binary Tree

Each node has max 2 children



Binary



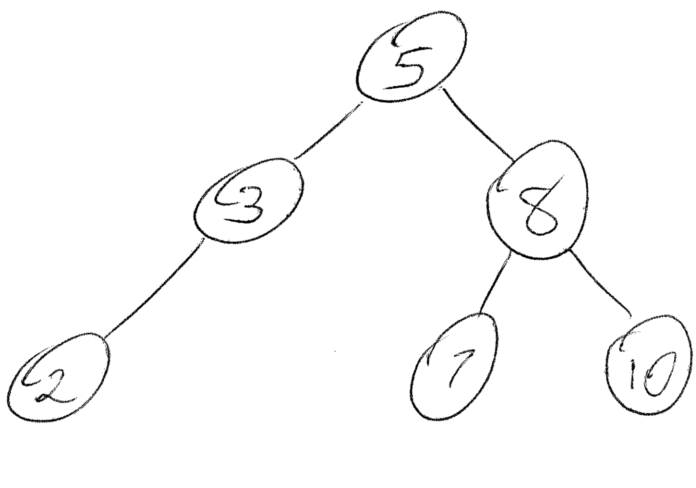
Not Binary

Definition

A Binary Tree is an empty node or a single node where the left and right pointers each point to a binary tree.

Binary Search Tree

A BST is an ordered Binary Tree. All items in the left subtree are less than the current node and all items in the right subtree are greater than or equal to the current node.



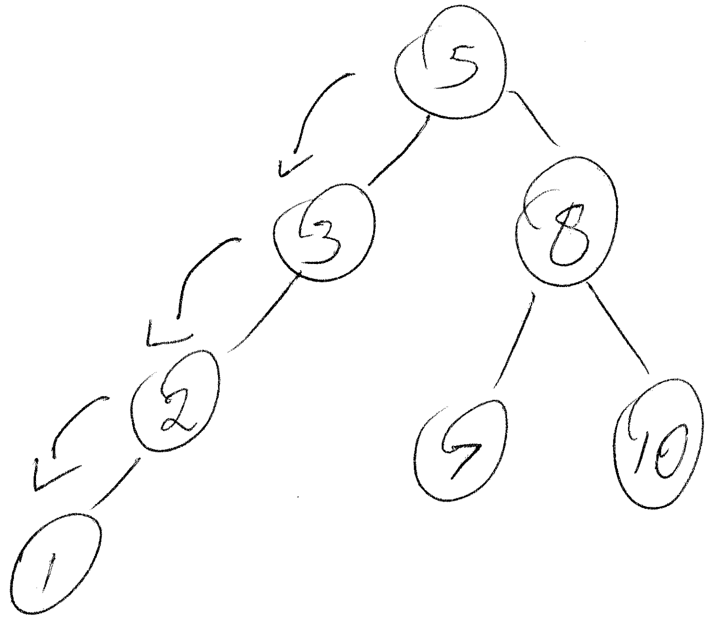
To insert 1 start at the root (5)

3

$1 < 5 \Rightarrow$ go left.

$1 < 3 \Rightarrow$ go left.

$1 < 2 \Rightarrow$ go left $\rightarrow$ empty space so insert to the left of 2



Find (7)

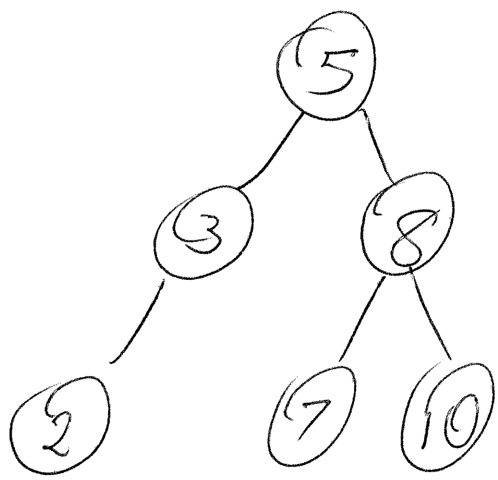
```
curr = root
if (curr.v == 7) return true;
else if (curr.v < 7)
    return find(7, curr.left); //search left
}else{
    return find(7, curr.right); //search right.
}
```

Min

As far left as possible.

Max

As far right as possible.



In-Order-Traversal

print left
print current
print right.

2, 3, 5, 8, 7, 10

Pre-Order-Traversal

print current
left
right

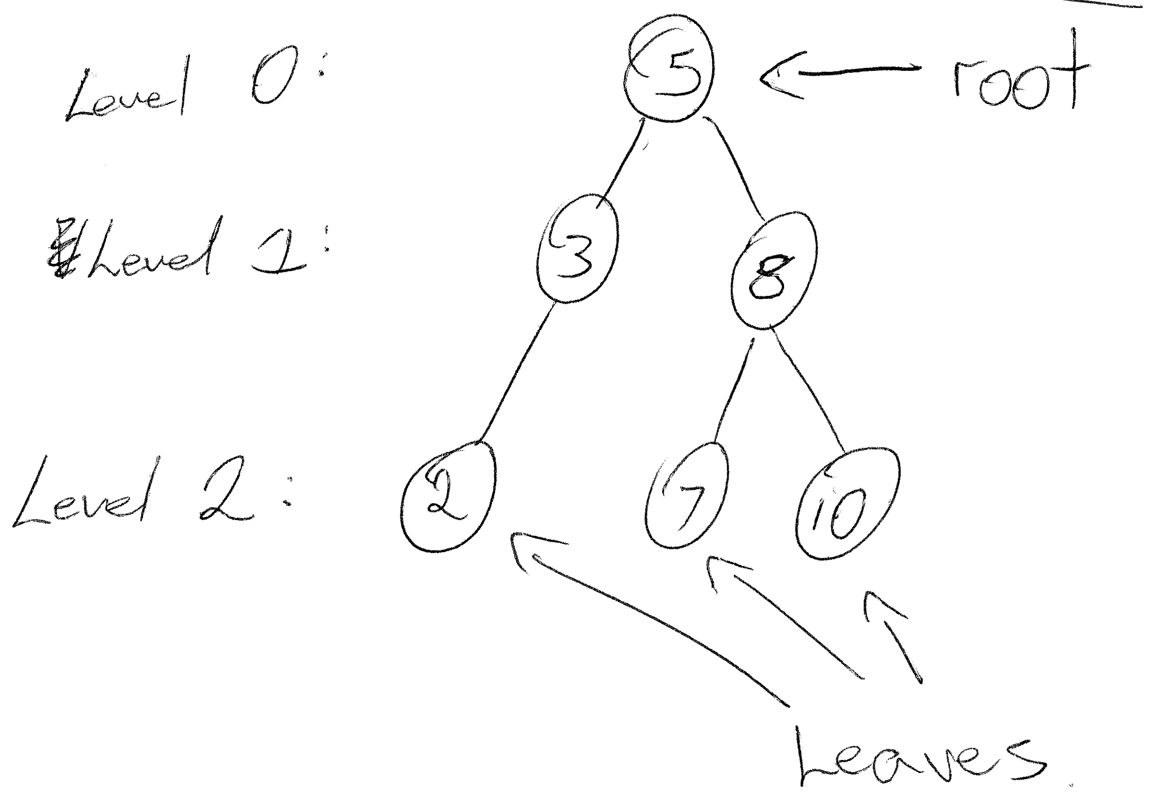
5, 3, 2, 8, 7, 10

Post-Order Traversal

left
right
current

2, 3, 7, 10, 8, 5

Structure and Definitions

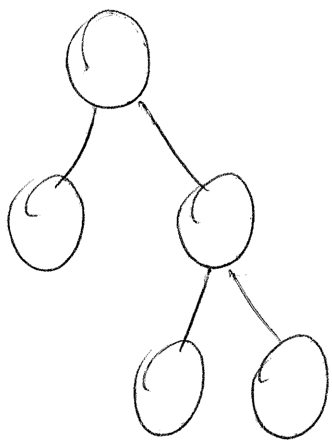


Depth of a node: number of edges from the node to the root.

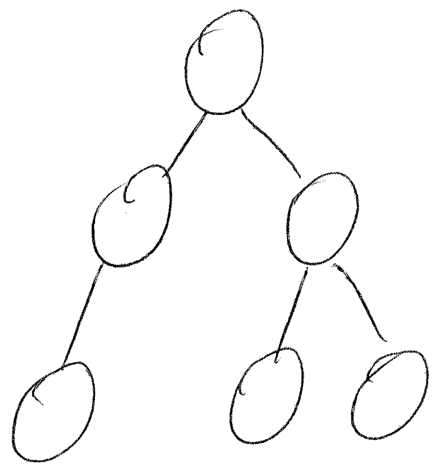
Height of a node: number of edges from the node to the deepest descendant leaf

Height of the tree: Height of the root.

Full Binary Tree: every node has exactly 0 or 2 children (internal or leaf node).

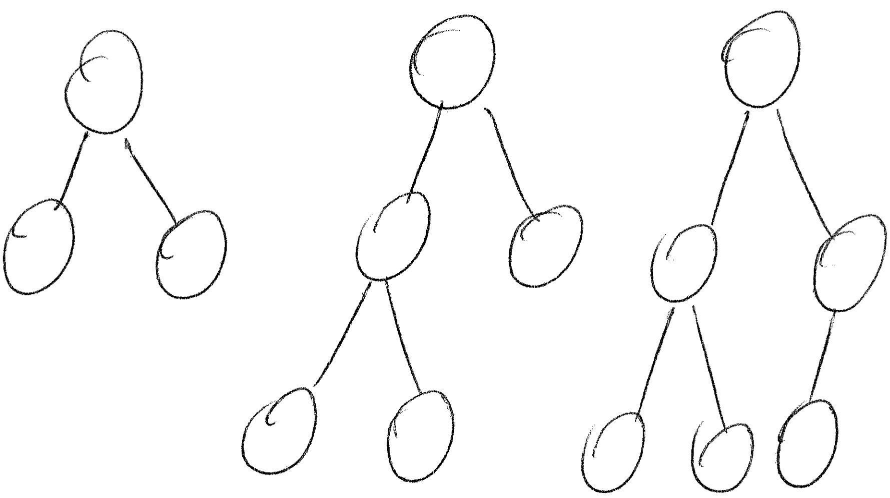


Full



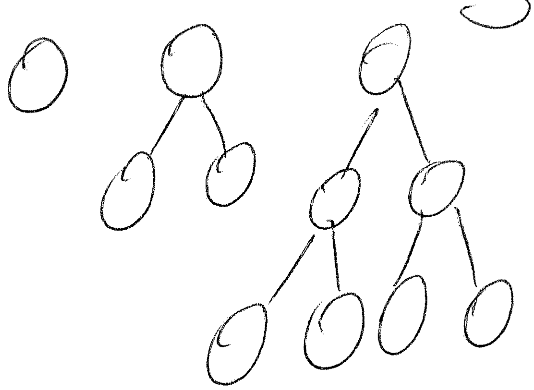
Not full.

Complete Binary Tree: Completely filled on every level except possibly the last.



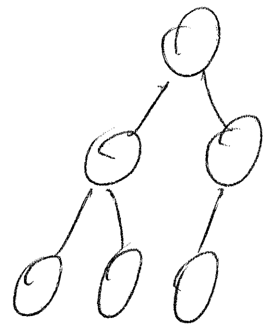
All complete.

Perfect: Every level is completely filled.

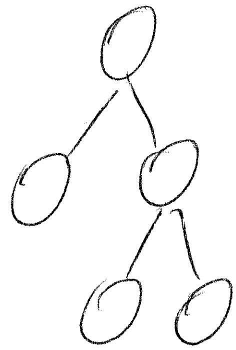


etc...

Does complete ⇒ full? No



Does full ⇒ complete? No

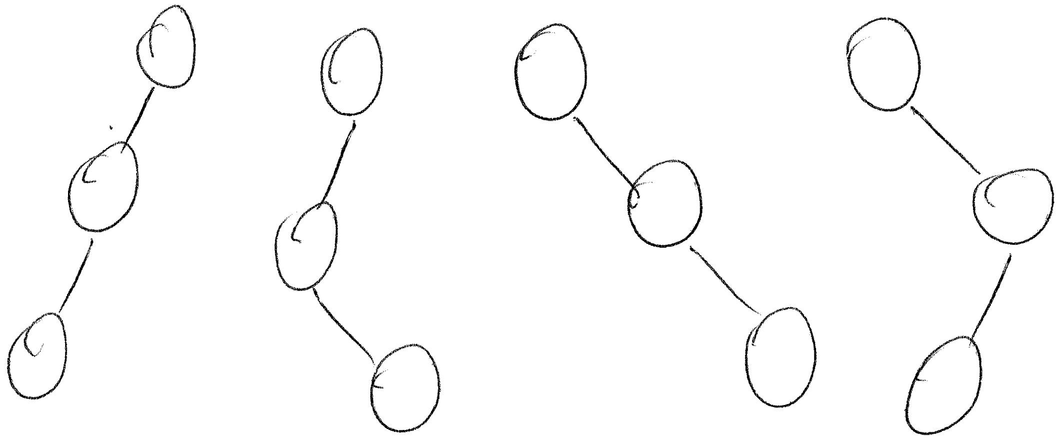


Perfect ⇒ complete and full.

Height of a tree with n nodes and height h.

Worst Case

Each node has exactly 1 child.



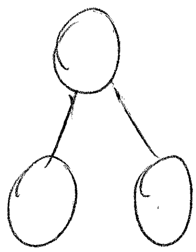
$h=2$

$h=n-1$

$$h = O(n)$$

Best Case

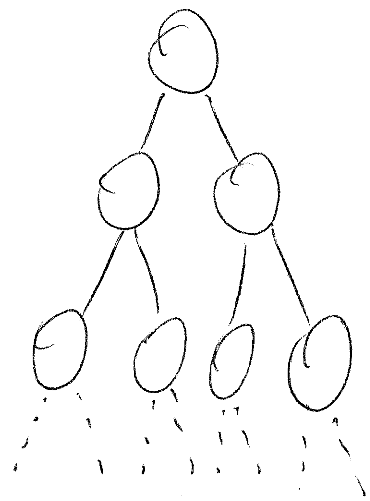
Complete



$h=1$

$h = O(\lg n)$

Nodes in a perfect tree of height h.



Level 0 has 1 node.

Level 1 has 2 nodes

Level 3 has 4 nodes.

etc...

Each level has twice as many nodes as the previous.

$$n = \sum_{i=0}^{h} 2^i = 2^{h+1} - 1$$ (Prove me with induction)

Number of leaves $= 2^h = (n+1)/2$

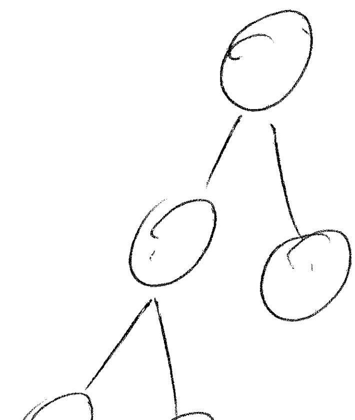
If $n = 2^{h+1} - 1$

$\Rightarrow h = \lfloor \lg n \rfloor$ (floor operation - round down)

$h = O(\lg n)$

Number of leaves in a complete BST.

Number of leaves, $l = \left\lfloor \frac{n+1}{2} \right\rfloor$



$h = 2$

$n = 5$

$l = 5$

Full Binary Tree

Nodes, height, leaves, internal nodes
n, h, l, I.

If there are I internal nodes, then:

1. $l = I + 1$
2. $N = 2I + 1$
3. $I = (N-1)/2$
4. $l = (N+1)/2$
5. $N = 2l - 1$
6. $I = l - 1$

Prove

I internal nodes $\Rightarrow l = I + 1$ for a non-empty tree, T.

Base Case

$I = 0$, no internal nodes.

$\Rightarrow T$ has 1 node, the root because it is non-empty.

$\Rightarrow l = 1$ as the root is a leaf.

Base Case Holds. ✓

Inductive ~~Step~~ Hypothesis.

Suppose for some $k \geq 0$, $l = I + 1 \ \forall I \in \{0, 1 \ldots k\}$

I.E. every full tree that has $0 \leq I \leq k$, has $l = I + 1$.

Inductive Step

let T be a full binary tree with $k+1$ internal nodes.

$k \geq 0 \Rightarrow k+1 \geq 1 \Rightarrow$ Root has at least 1 child. but the tree is full, so the Root must have 2 children.

Let these subtrees be L and R.

Every Internal node in L is an internal node in T. Similarly for R.

$\Rightarrow I = I_L + I_R + 1$

as the root is an internal node.

I_L: Internal nodes of L.

I_R: Internal nodes of R.

Similarly, every leaf in L is a leaf in T.

$$l = l_L + l_R$$

Now T had $k+1$ internal nodes and the root is an internal node.

$$\Rightarrow I_L < k+1 \; ; \; I_R < k+1$$

i.e. $I_L \leq k$; $I_R \leq k$.

But by the Inductive hypothesis

$$l_L = I_L + 1 \; ; \; l_R = I_R + 1$$

$$\Rightarrow l = l_L + l_R = I_L + 1 + I_R + 1$$

But $I = I_L + I_R + 1$

$$\Rightarrow l = I + 1$$

$\therefore$ By the principal of strong mathematical induction, $l = I + 1 \;\; \forall \, I \geq 0$.

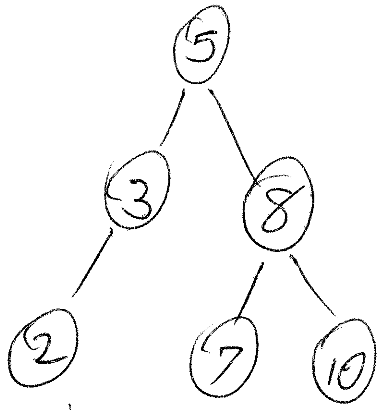
$\square$.

Deleting from a BST

There are a few cases:

1. Deleting a leaf.

Delete(2)



Go to the parent of the leaf, and delete the leaf, set the parent's pointer to null.

2. Node has 1 child.

Delete(3).

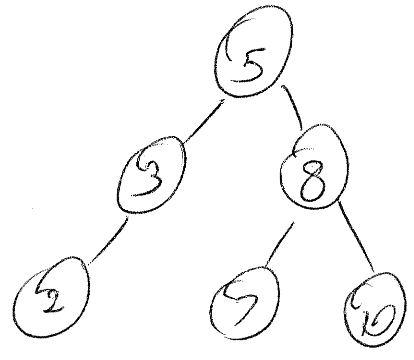
Go to the parent of 3, set its relevant child pointer to point to 3's child. Delete 3 (remember to keep a tmp pointer).

Case 3 — The node has 2 children.

→ Need to replace it with a node that is bigger than the left but smaller than the right child.

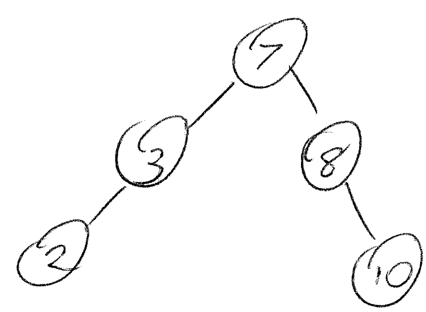
→ Take the min value from the right subtree (1 step right, then as far left as possible.)
and put that value in the place of the node to be deleted.

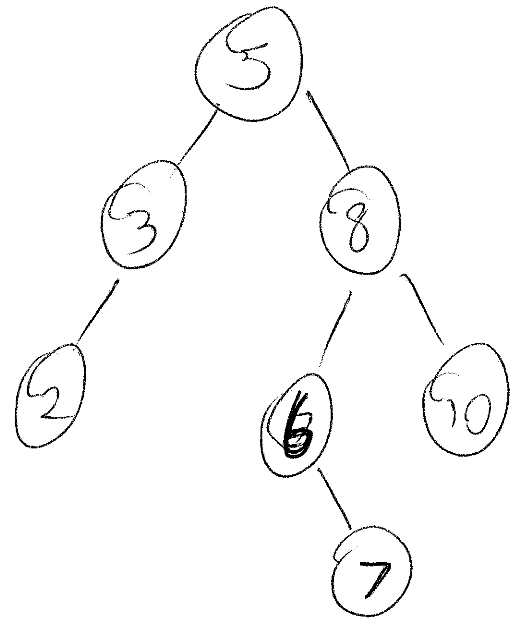
→ Now delete the value from the right subtree using cases 1 or 2



Delete (5)

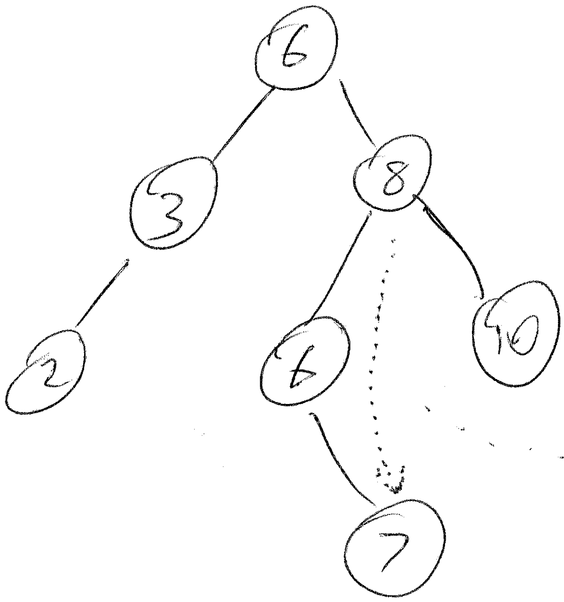
⇒





Delete 5

Move 6 to replace 5.



Now delete 6 from the tree rooted at 8.