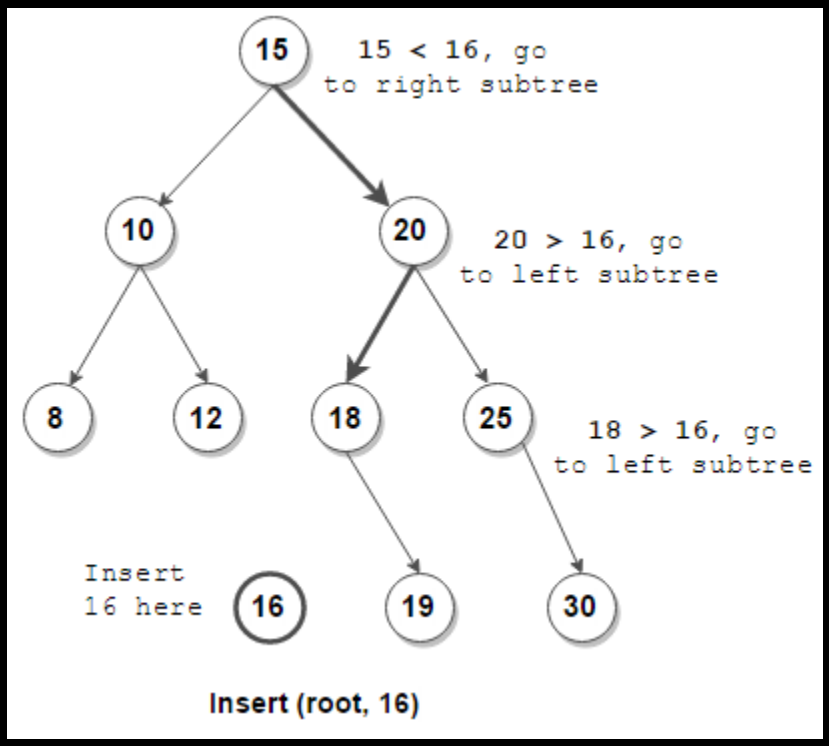
About

A tree in which each node (parent) has at most two-child nodes (left and right) is called binary tree. The top most node is called the root node. In a binary tree a node contains the data and the pointer (address) of the left and right child node.

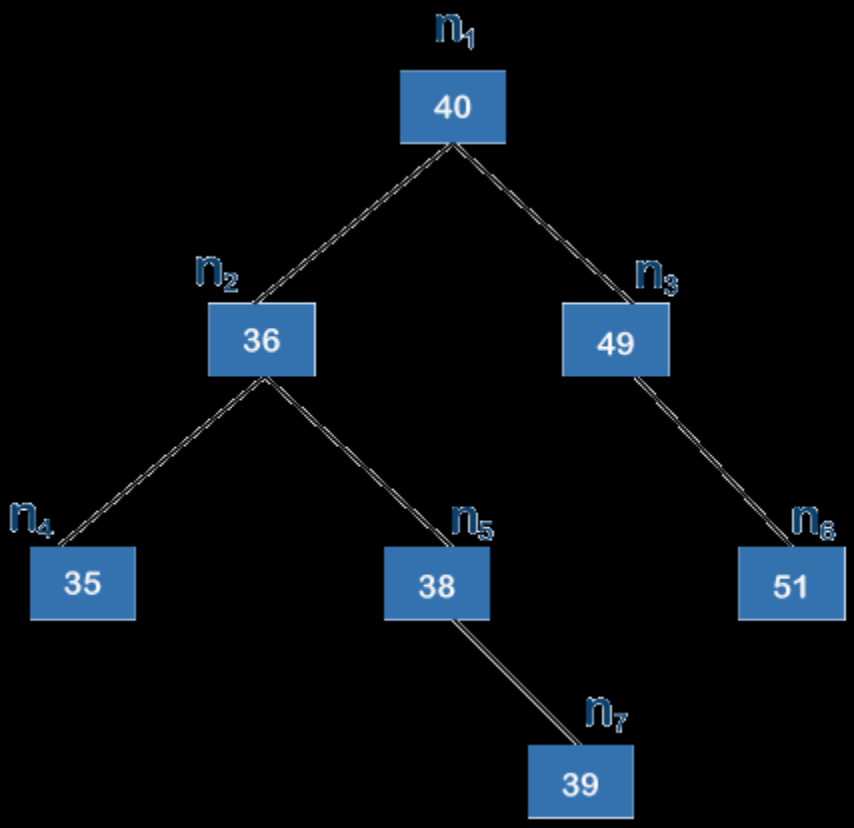


Explanation: Lets say the list is 15, 10, 8, 20, 12, 18, 25, …
We first take 15 as root as it first in the queue then making it as root taking second “10” compare with 15 if less put in left side of the root node then 20>15 so right side.. Now 8? So since it is less than 15 it will be below 15 now 15 has already 2 child right 20 and left 10 since it went to left it will be now compared with 10 so its small than 10 so it will below 10 left side. And so does 12 its greater than 10 so right of 10 as 10’s child node. 18>15 so goes to 20 again same problem 15 has 2 childs so now it will be compared to 20>18 so left of 20. And so on..

Operations

Example Data Set

[40, 36, 49, 35, 38, 51, 39]



Insert

Big O: O(log n)

Delete

Big O: O(log n)

3 Cases to be consider while deleting the Tree's node
Case 1: Node with 0 Child - No successor
Case 2: Node with one child - that one child becomes successor
Case 3: Node with 2 Childs – we need to get successor for this that is left-most right node.

Traversals (Pre Order)

Big O: O(log n)

In this case you will traverse and read data from root node with Visit Left Right **VLR** (First read node then move to Left node first then cover right nodes) In short top to bottom approach

Visit the root.
Traverse the left subtree
Traverse the right subtree

So as per Pre-Order Travesal will be: 40, 36, 35, 38, 39, 49, 51

Traversals (In Order)

Big O: O(log n)

In this case you will traverse left most bottom then read node then right most and then upwards **LVR** In short reading subtrees approach

Traverse the left subtree
Visit the root.
Traverse the right subtree

So as per In-Order Travesal will be: 35, 36, 38, 39, 40, 49, 51

Traversals (Post Order)

Big O: O(log n)

In this case you will traverse and read data left most bottom node then right most and then upwards **LRV** In short bottom to top approach

Traverse the left subtree
Traverse the right subtree
Visit the root.

So as per Post-Order Travesal will be: 35, 39, 38, 36, 51, 49, 40

Application of Tree

https://www.baeldung.com/cs/applications-of-binary-trees

References

- Binary Search Tree in Java - 2: Delete a node of binary search tree
- Recursive PreOrder traversal of a Binary Tree in Java
- Recursive Postorder traversal of a Binary Tree in Java
- Recursive Inorder traversal of Binary Tree in Java