Binary Search Tree

Ayushi Dwivedi
Sourav Girish Walke

Agenda

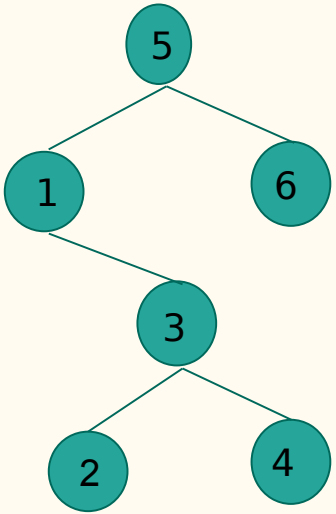
- What is a Binary search Tree?
- Time Complexity
- Insertion
- Deletion
- Traversal
- Summary

What is a Binary Search Tree?

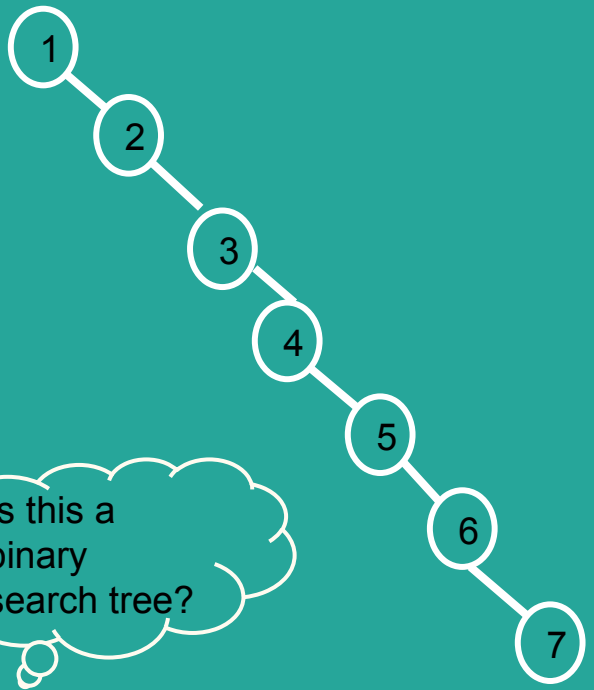
Binary Search Tree

Binary search tree is node based binary tree, which has following characters:

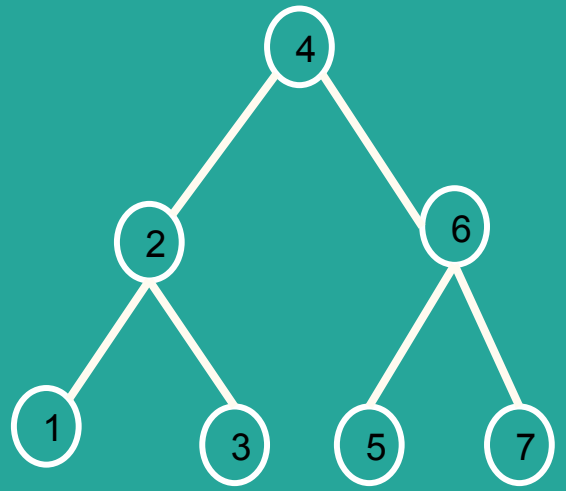
1. Left subtree to a node contains all the nodes with lesser key value than node’s key
2. Right subtree to a node contains all the nodes with greater key value than node’s key
3. Left and right subtree must also be a binary search tree



Quiz Time!

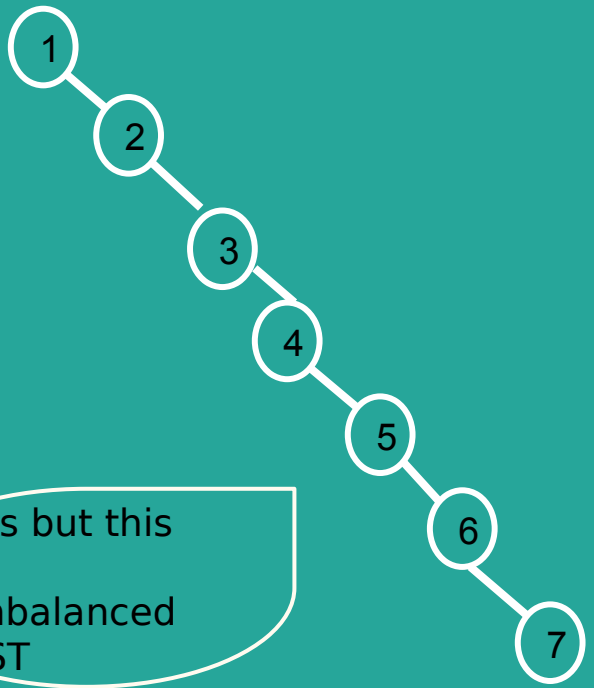


Is this a binary search tree?

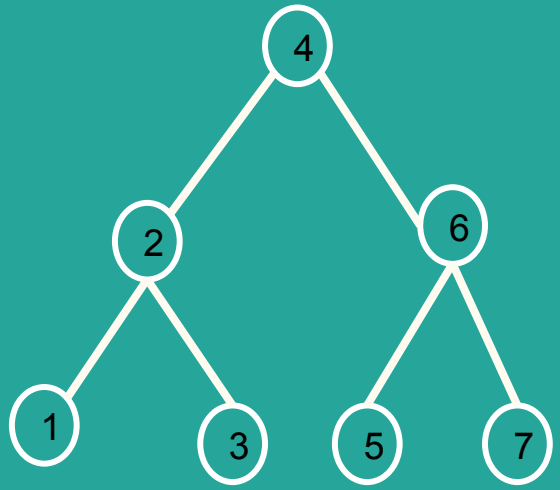


Is this a binary search tree?

Answer



Yes but this is imbalanced BST

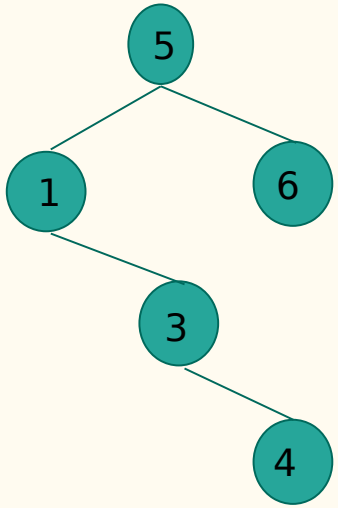


Yes, but this one is balanced BST

How to perform search in Binary Search Tree?

Steps are:

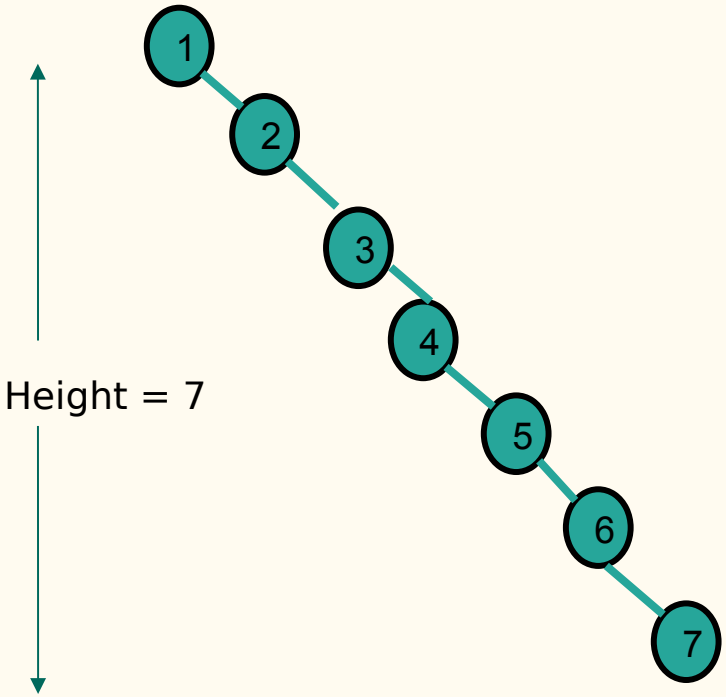
1. Compare the search key with root, if matches then return
2. If key is lesser than root then search in left sub tree
3. If key is greater than root then search in right subtree



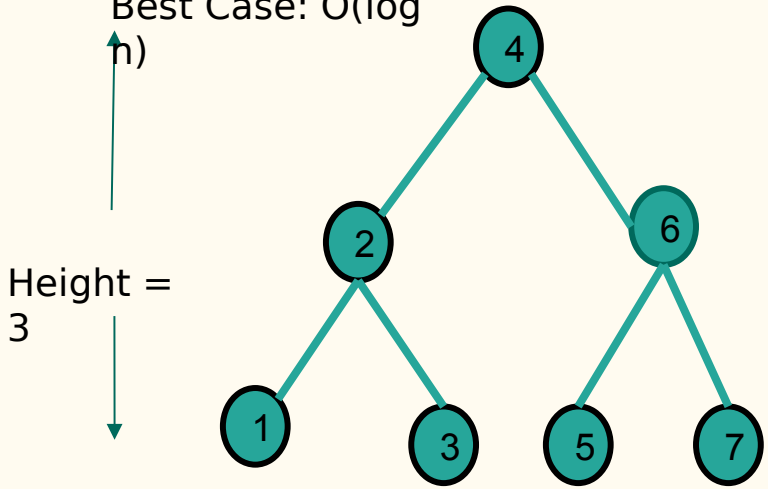
The search is similar to binary search, hence the tree is named as Binary Search Tree

Time Complexity?

Worst Case: O(n)



Best Case: O(log n)



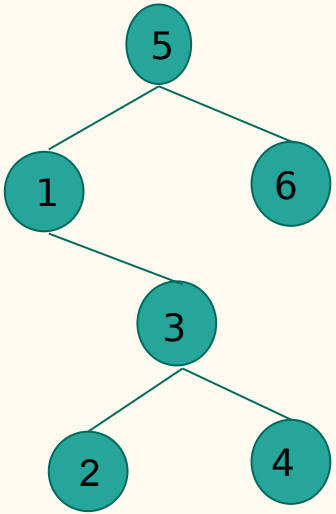
In general scenario, we can say that Time Complexity is O(H), where H is height of the tree

Importance of Binary Search Tree

Binary search tree is stored by following specific order, thus operations like

1. Searching an element
2. Finding minimum and maximum values
3. Insertion into a tree

   can be done faster than any other data structure

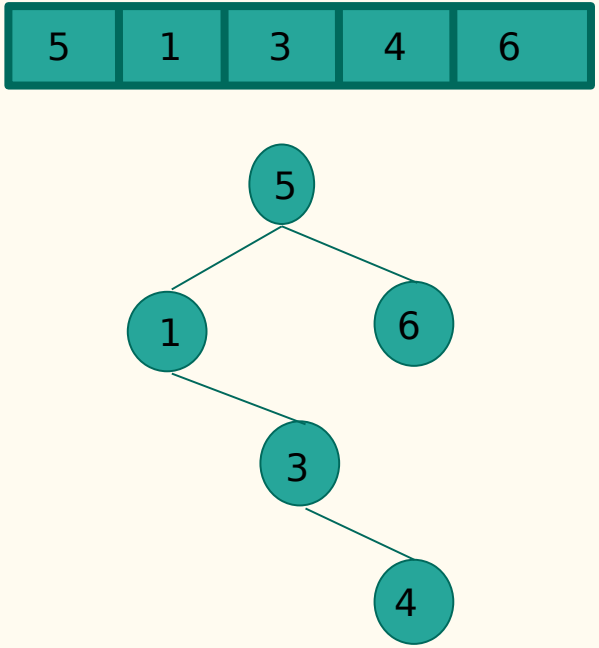


Insertion

Insertion

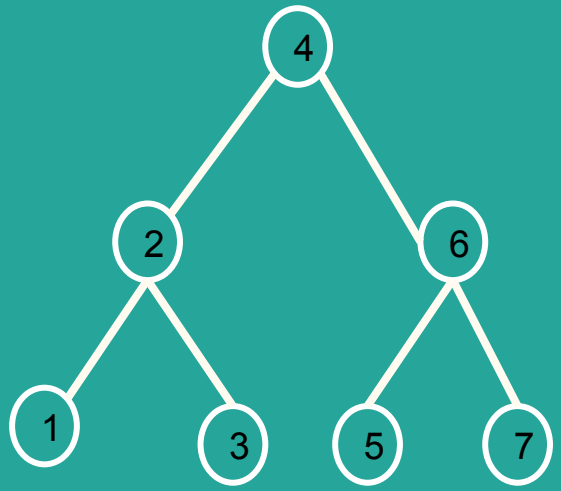
New key is always inserted at the leaf node of the Binary Search Tree. Steps for insertion are:

1. Start from the root node
2. Compare the inserting key with root node, if lesser than root then recursively call insertion on left subtree or in case it is greater then recursively call insertion on right subtree
3. On reaching the end, insert the key after comparing to leaf



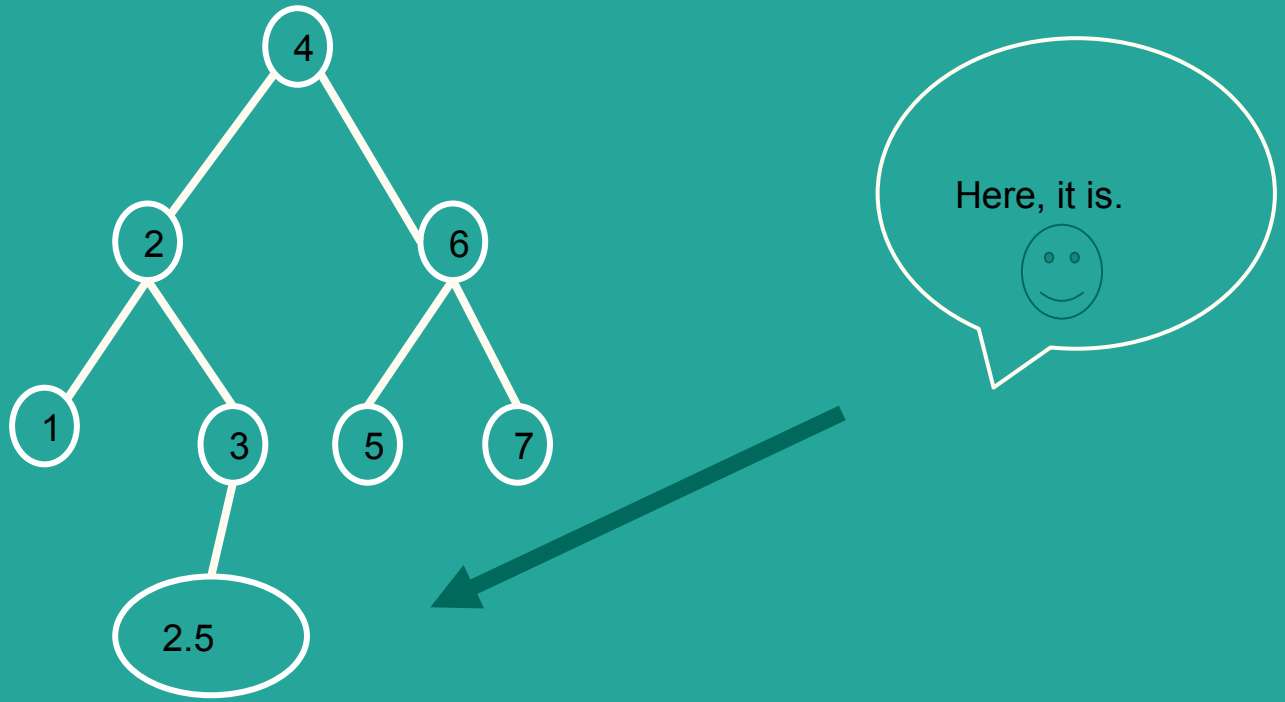
Time Complexity: O(h); h = height of tree

Quiz Time!



Where are you going to insert 2.5 in this tree?

Answer



Deletion

Deletion

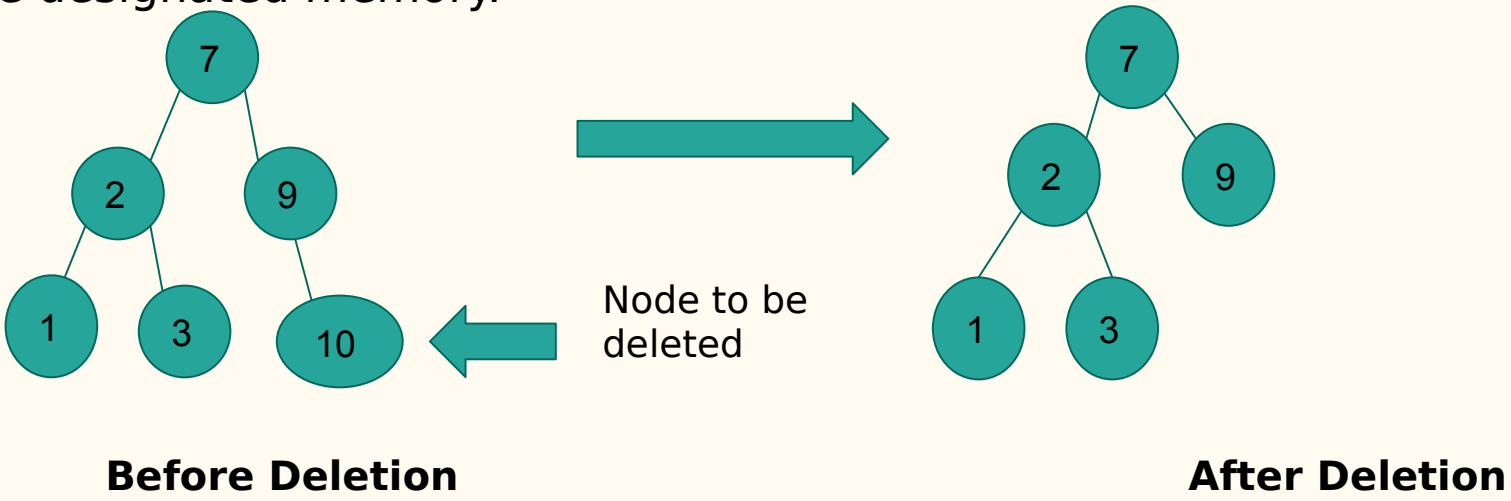
Deletion on a Binary search tree deals with deleting a node from the tree. This operation must be performed such that the **property of the binary search trees is not violated.**

There are three possible scenarios to perform deletion on a binary search tree:

- The node to be deleted is the leaf node
- The node to be deleted has only one child
- The node to be deleted has two children.

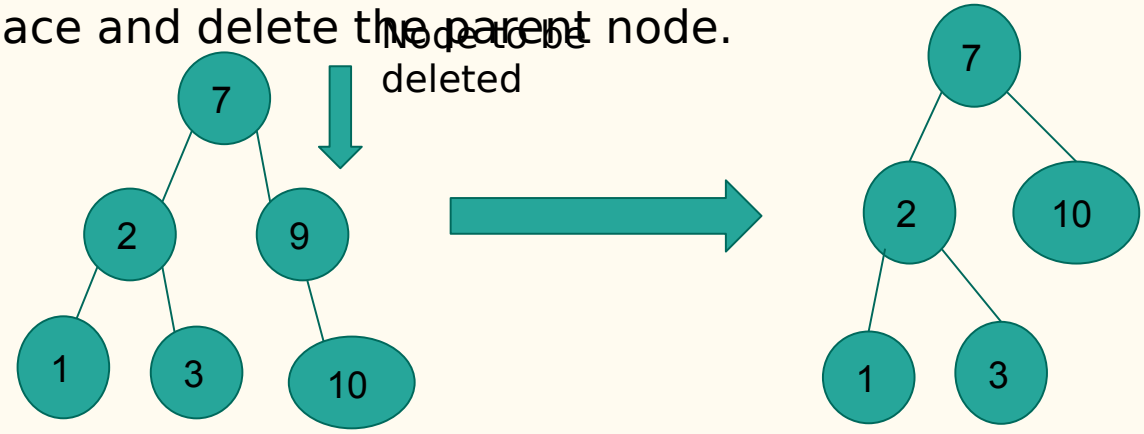
Scenario 1: Deleting a leaf node

When we are supposed to delete a leaf node then we simply traverse to that node and delete it i.e assign NULL to the leaf node and clear the designated memory.



Scenario 2: Deleting a node with One Child

When we are supposed to delete a node with one child we first traverse to that node and copy the child of that node in the parent's place and delete the parent node.



Before Deletion **After Deletion**

Scenario 3: Deleting a node with Two Children

When we are supposed to delete a node with two children

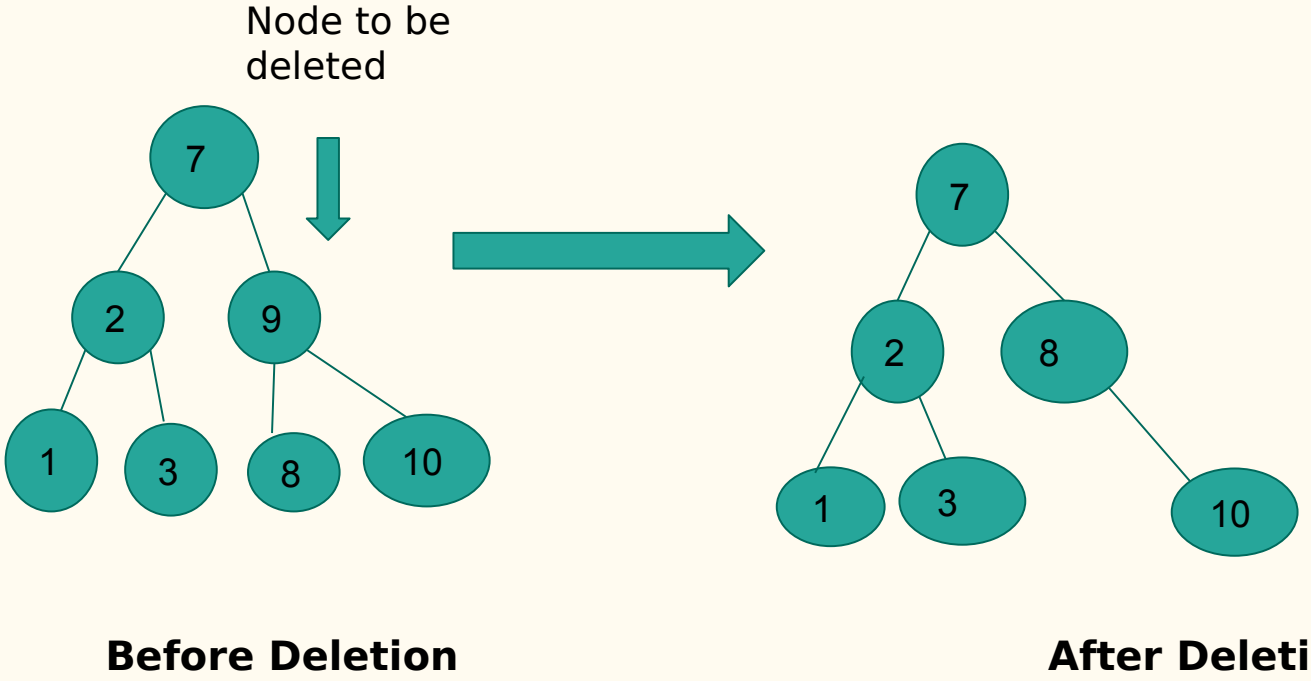
- We first traverse the tree and find out it’s **inorder successor/predecessor** depending upon the given tree.

Inorder Successor is a node with minimum value in the right subtree of the root node.

Inorder Predecessor is a node with maximum value in the left subtree of the root node.

- After finding this, we need to copy the content of that node and replace it with the content of the node to be deleted.

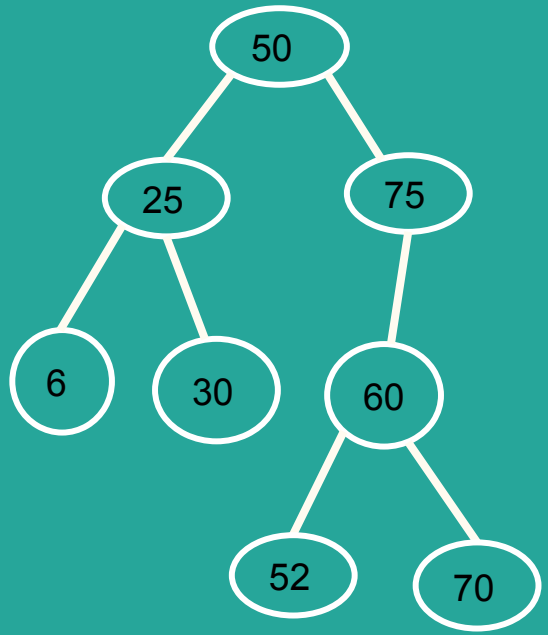
Example for Scenario 3



Before Deletion

After Deletion

Quiz Time!



What is the inorder successor if the node to be removed is 50 ?

Yup, it is 52.

Traversal

Traversing a Binary Search Tree

Traversing a Binary Search tree is the process of visiting each node in the tree exactly once in a particular order.

There are three methods of traversing a tree:

- Inorder Traversal - Left, Root and Right
- Preorder Traversal - Root, left and right
- Postorder Traversal - Left, Right and Root

Summary

- A **binary search tree (BST)** is a binary tree. The key difference is that a BST only allows you to store nodes with lesser value on the left side and nodes with greater value on the right.
- The binary search tree is built on the idea of **Binary search** algorithm, which allows for fast lookup, insertion and removal of nodes.
- The advantage of using a binary-search tree is that, if the tree is reasonably balanced (shaped more like a "full" tree than like a "linear" tree) the insert, lookup, and delete operations can all be