

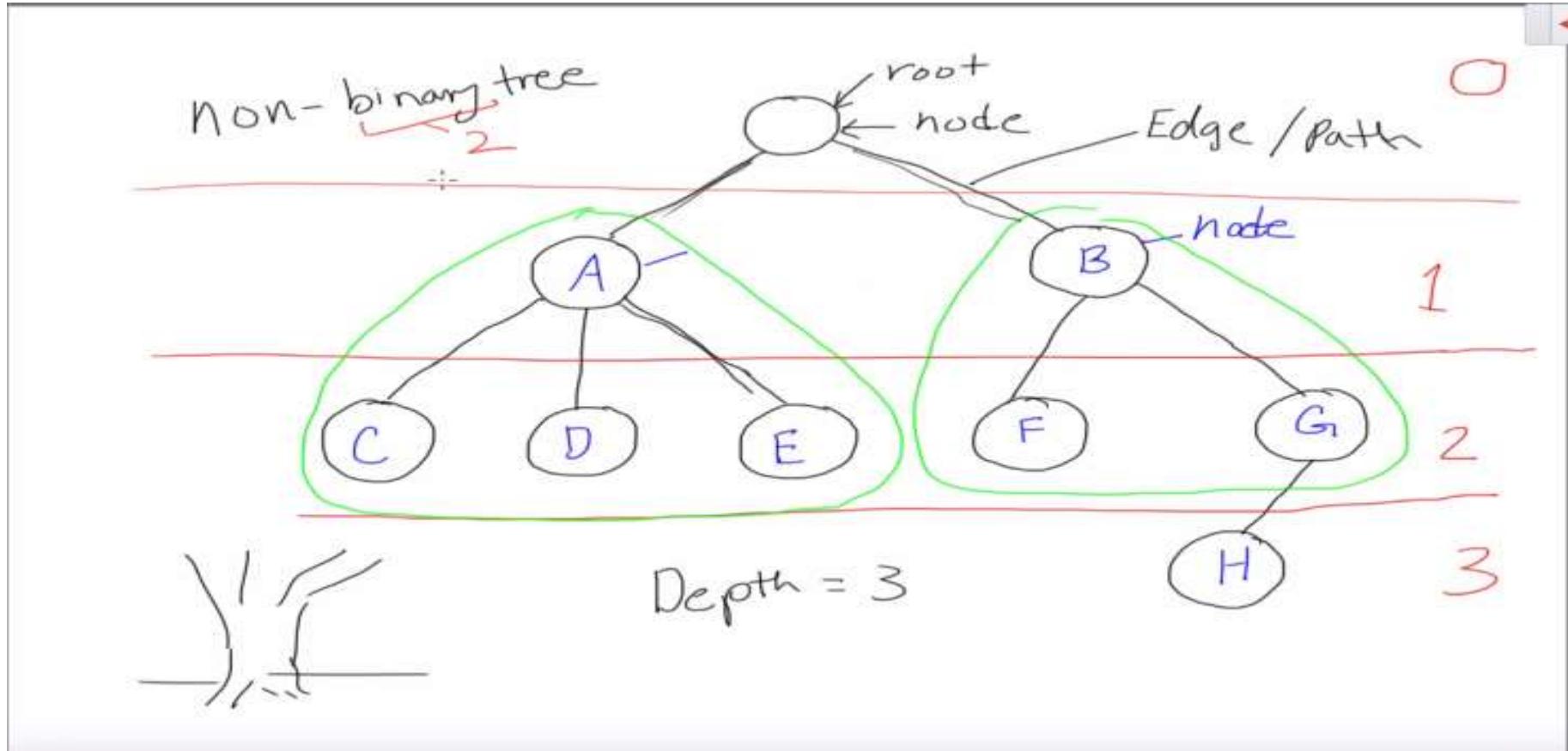
Binary Search Tree

- In computer science, a binary search tree (BST), also called an ordered or sorted binary tree, is a rooted binary tree whose internal nodes each store a key greater than all the keys in the node's left subtree and less than those in its right subtree. A binary tree is a type of data structure for storing data such as numbers in an organized way. Binary search trees allow binary search for fast lookup, addition and removal of data items . The order of nodes in a BST means that each comparison skips about half of the remaining tree, so the whole lookup takes time proportional to the binary logarithm of the number of items stored in the tree. This is much better than the linear time required to find items by key in an (unsorted) array

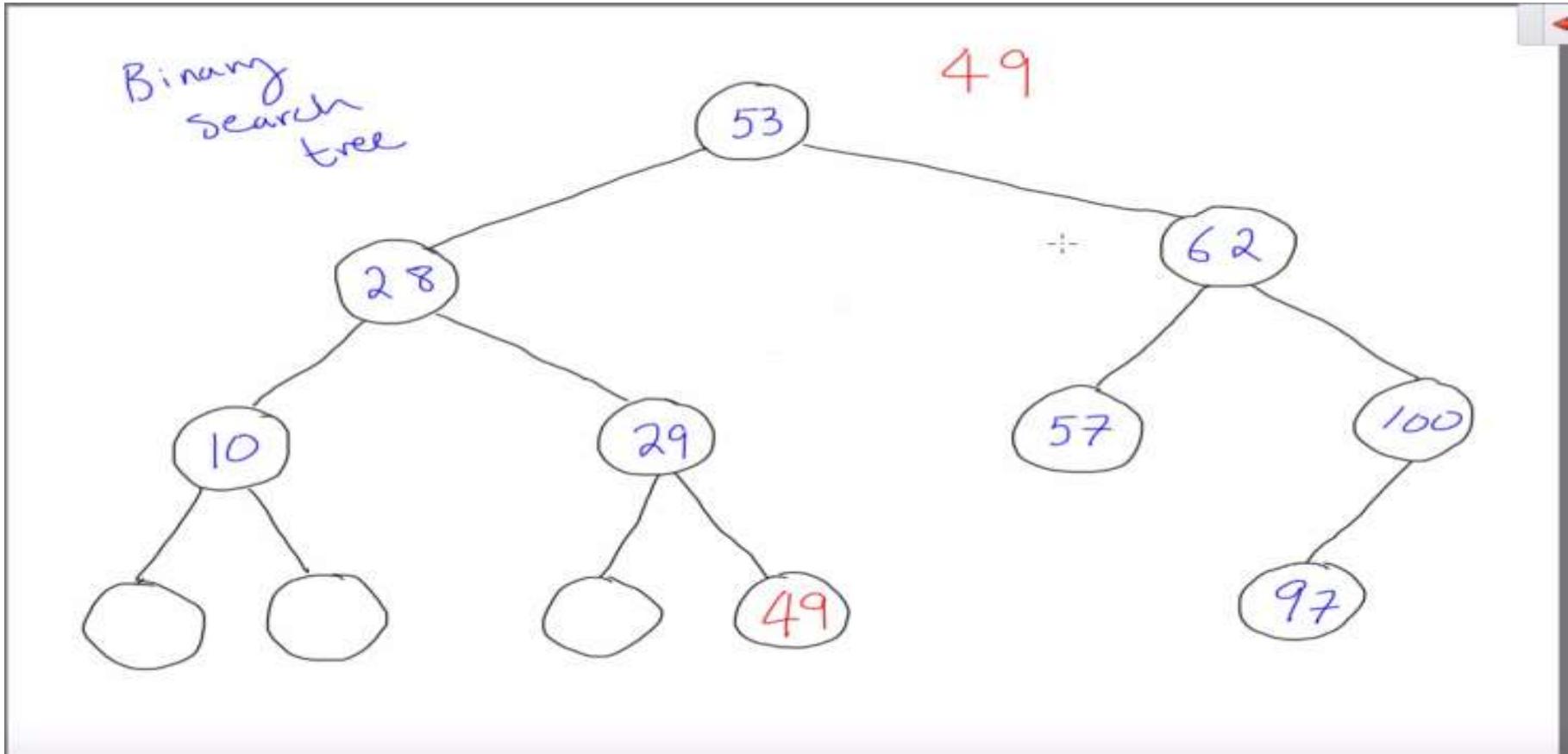
Complexity

Algorithm	Average	Worst case
Space	$O(n)$	$O(n)$
Search	$O(\log n)$	$O(n)$
Insert	$O(\log n)$	$O(n)$
Delete	$O(\log n)$	$O(n)$

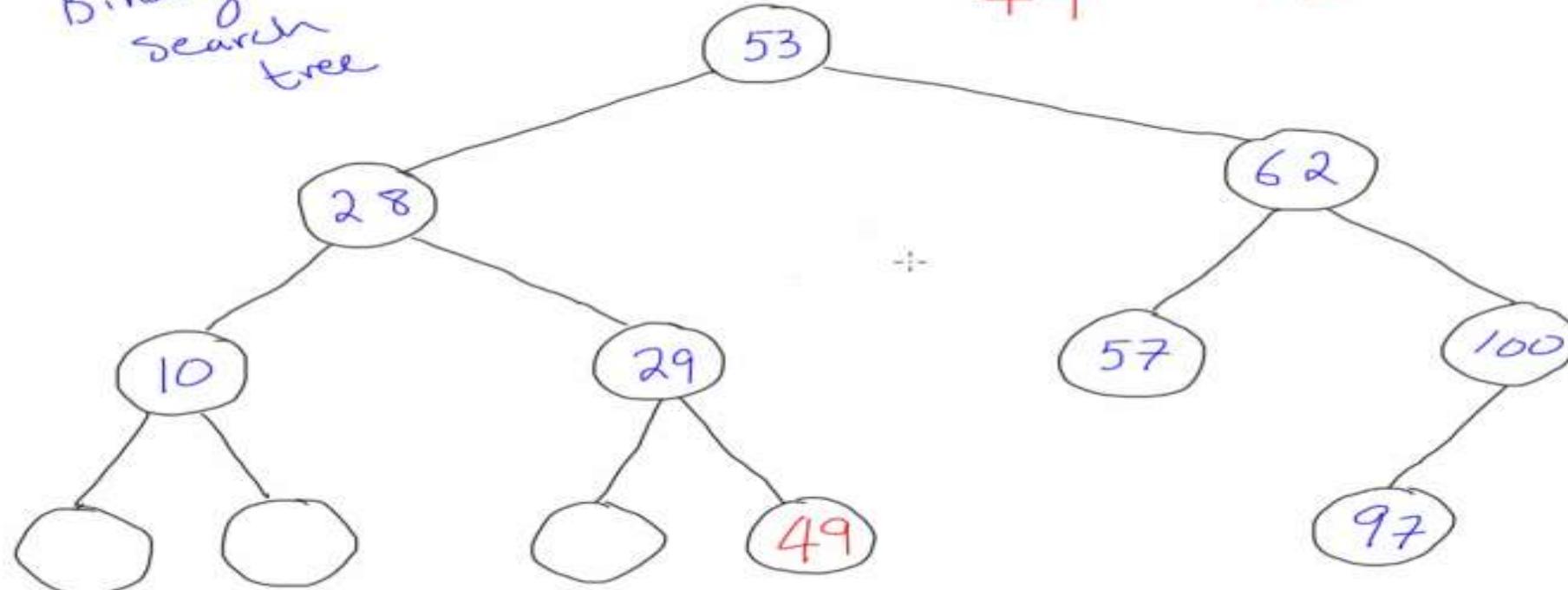
Non-Binary Tree Example



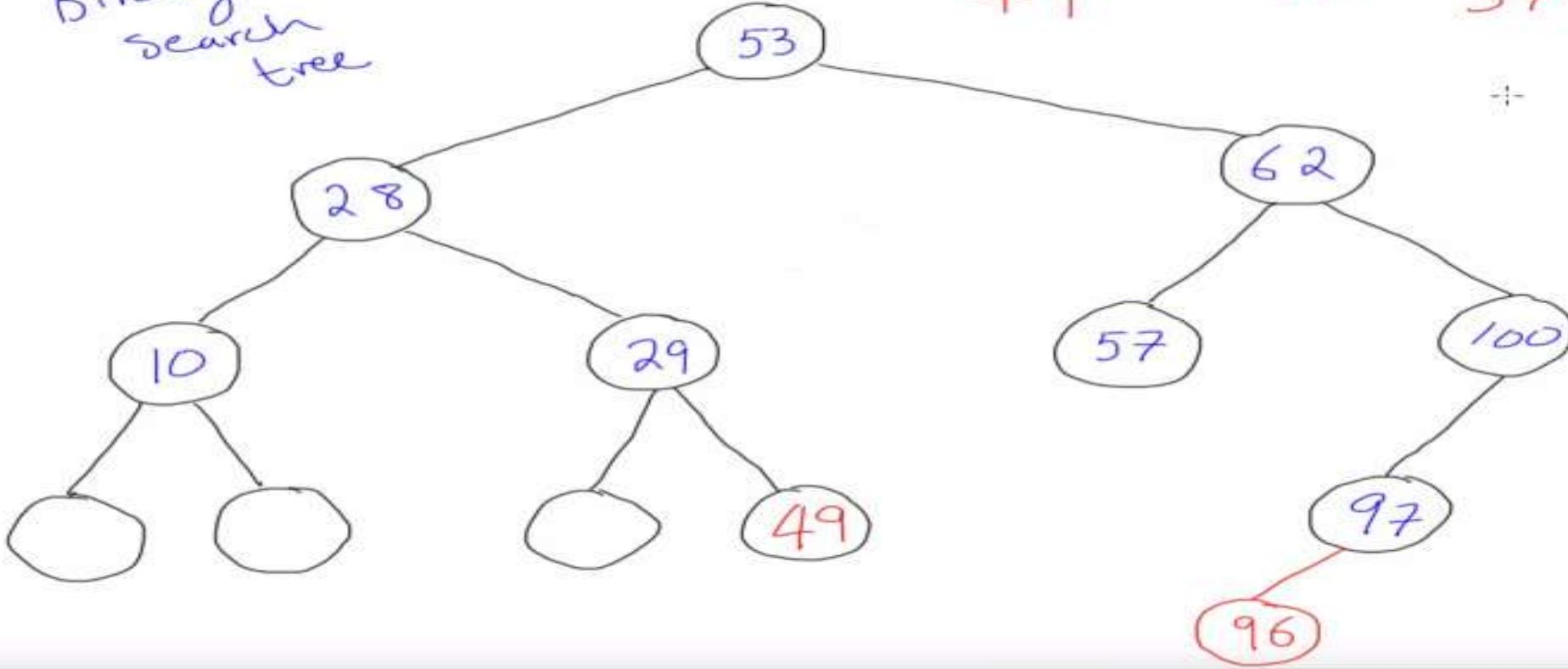
Binary Tree Example



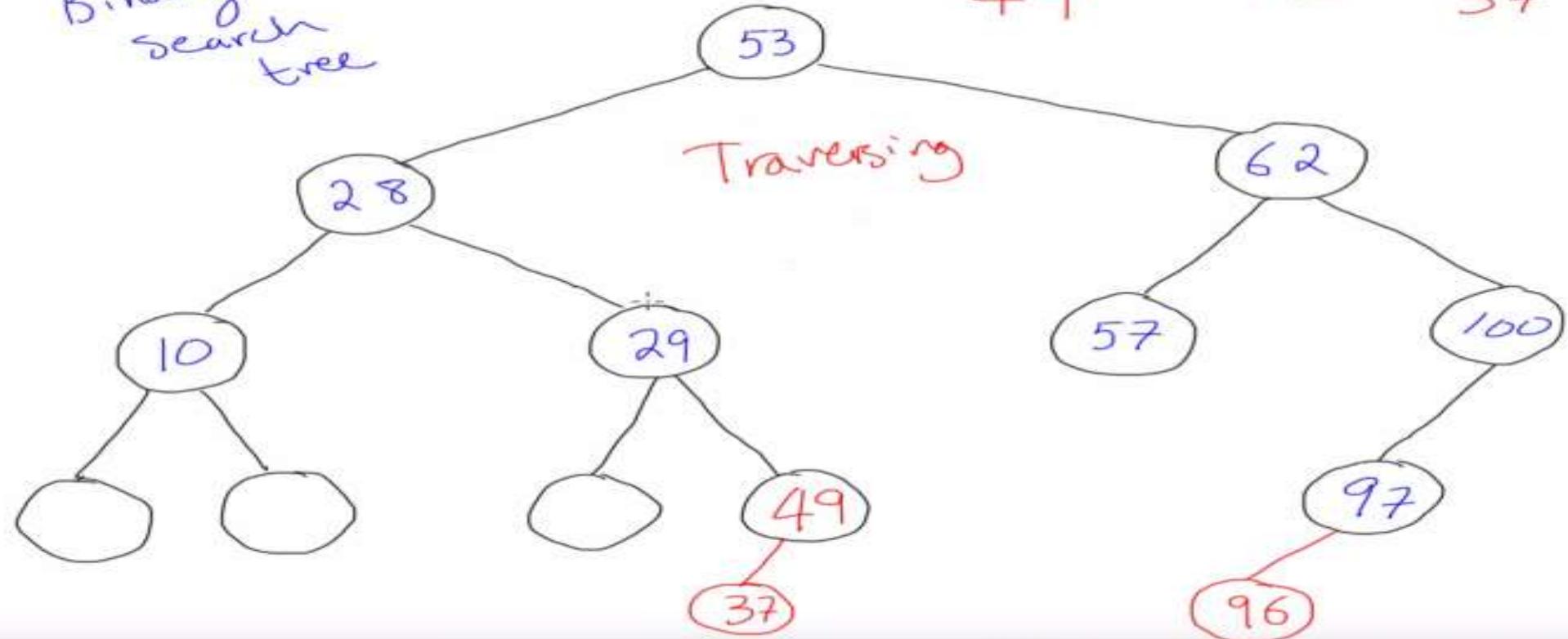
Binary
search
tree



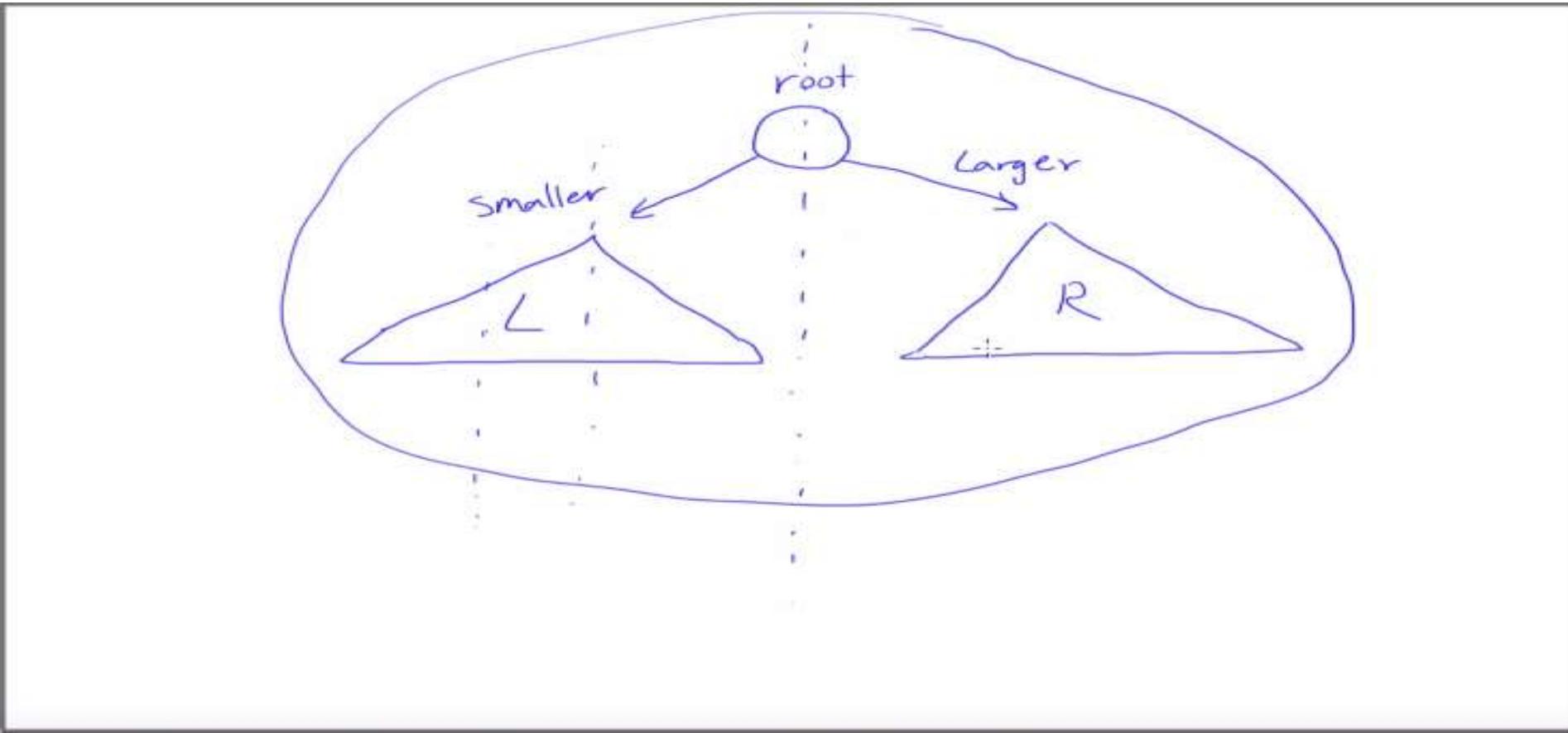
Binary
Search
Tree



Binary
search
tree



Overview



worse

Search(x) $O(n)$

Insert(x) $O(1)$

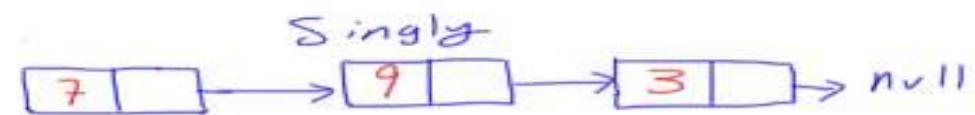
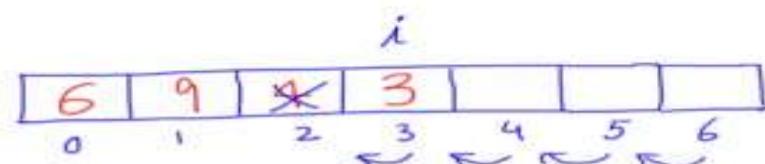
remove (x) $O(n)$

Search (x) $O(n)$

insert+ (x) $O(1)/O(n)$

remove (x) $O(n)$

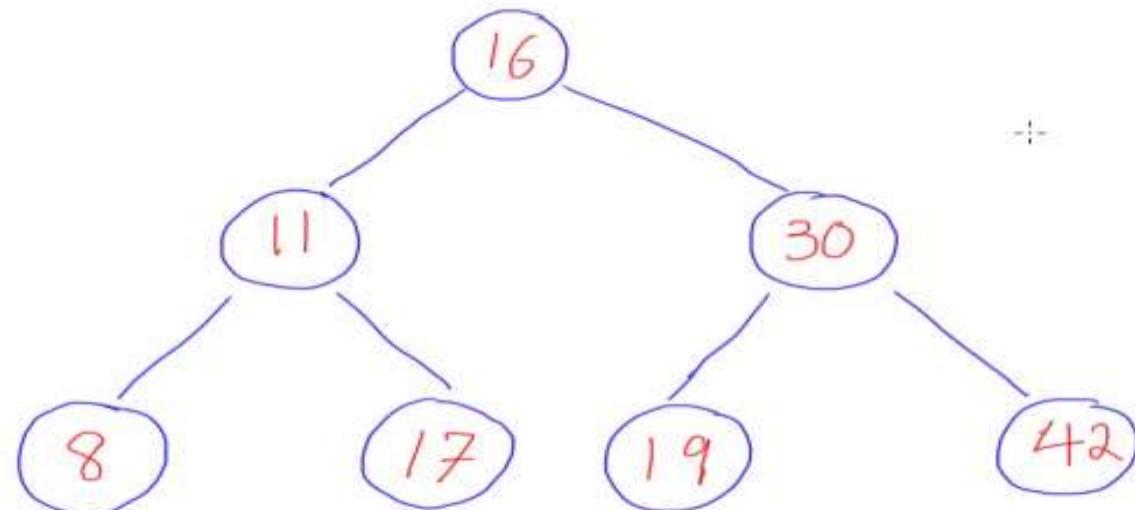
$a[i+1] = \infty$



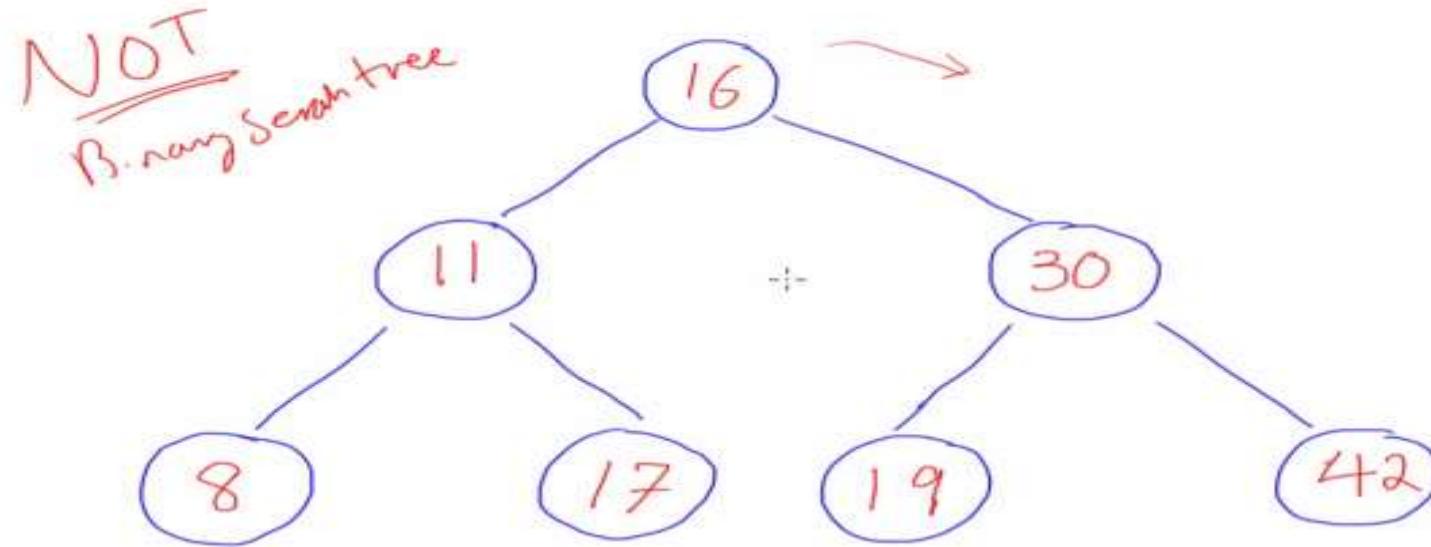
$O(\log n)$

Balanced ↗

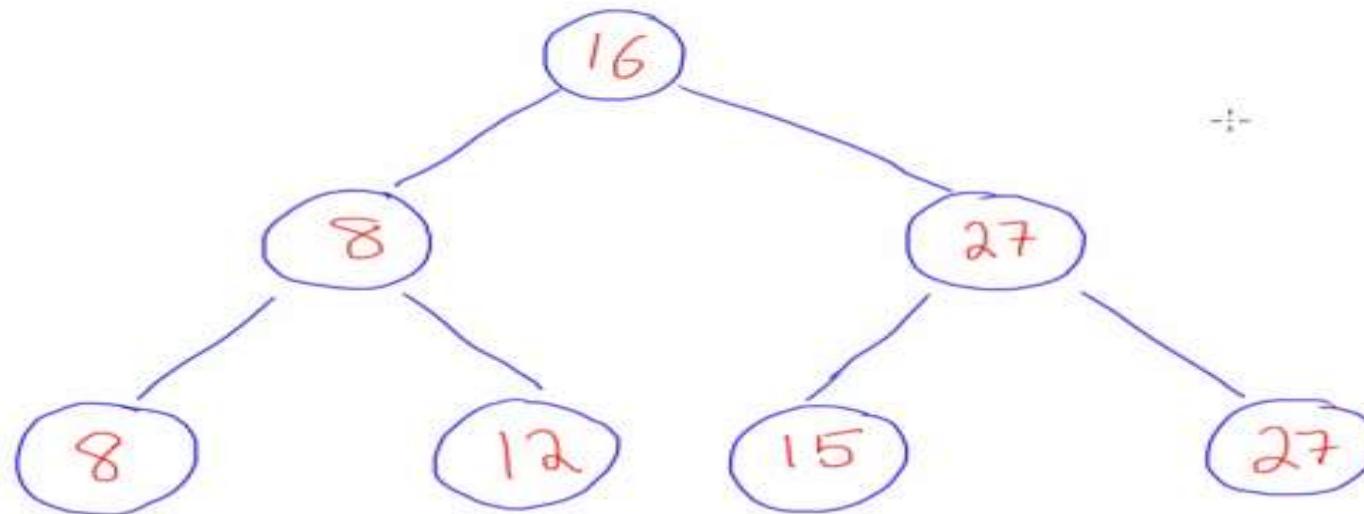
Q) Is it a Binary Search Tree ?



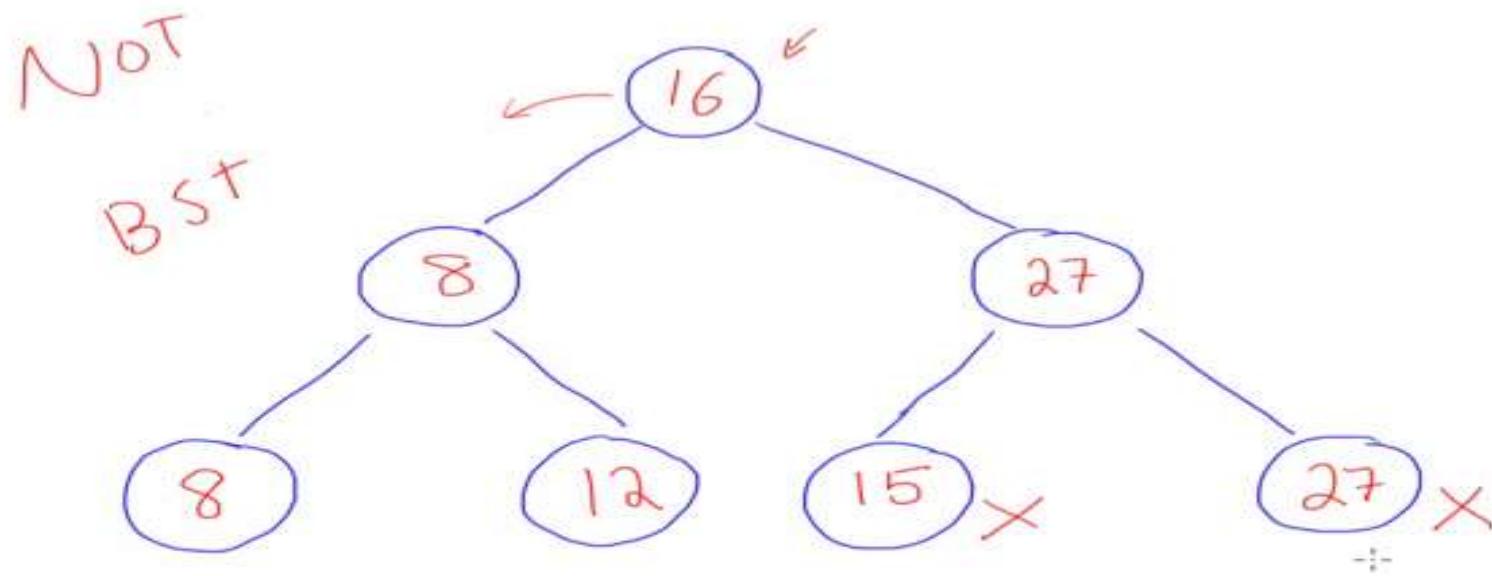
Answer



Q) Is it a Binary Search Tree ?

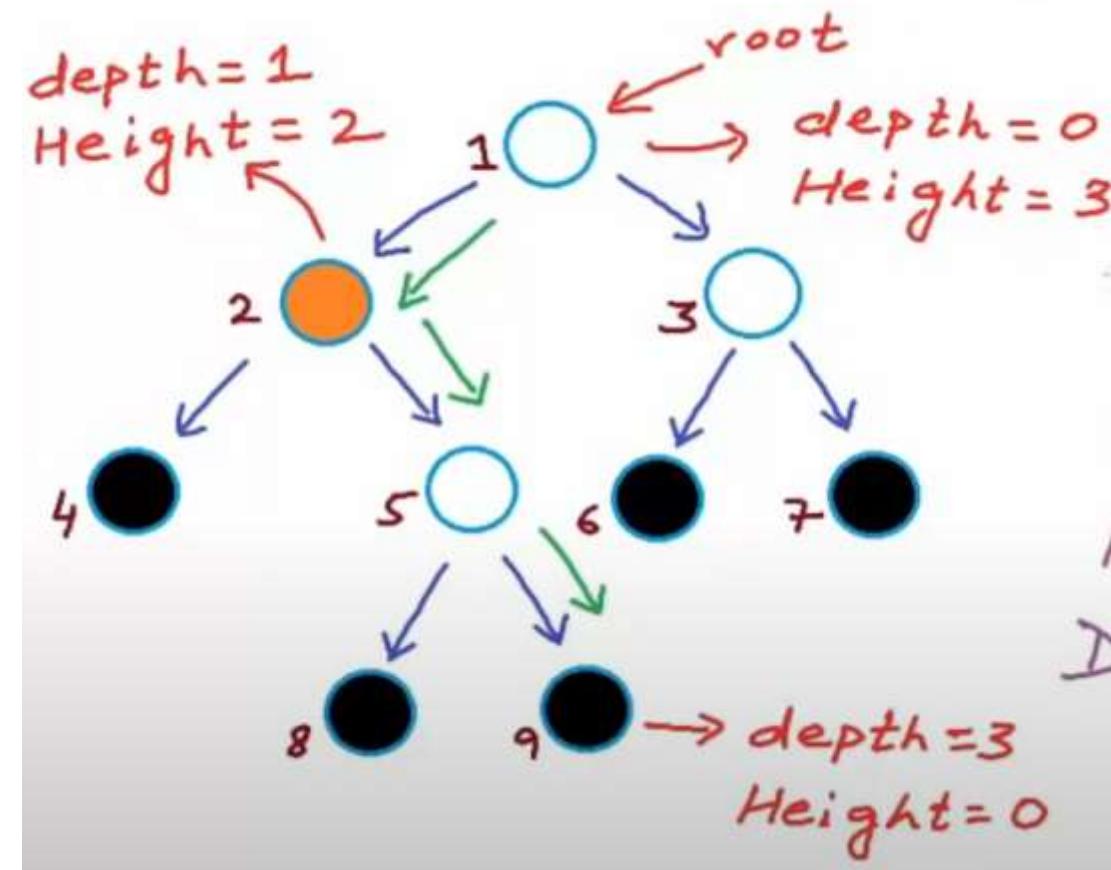


Answer

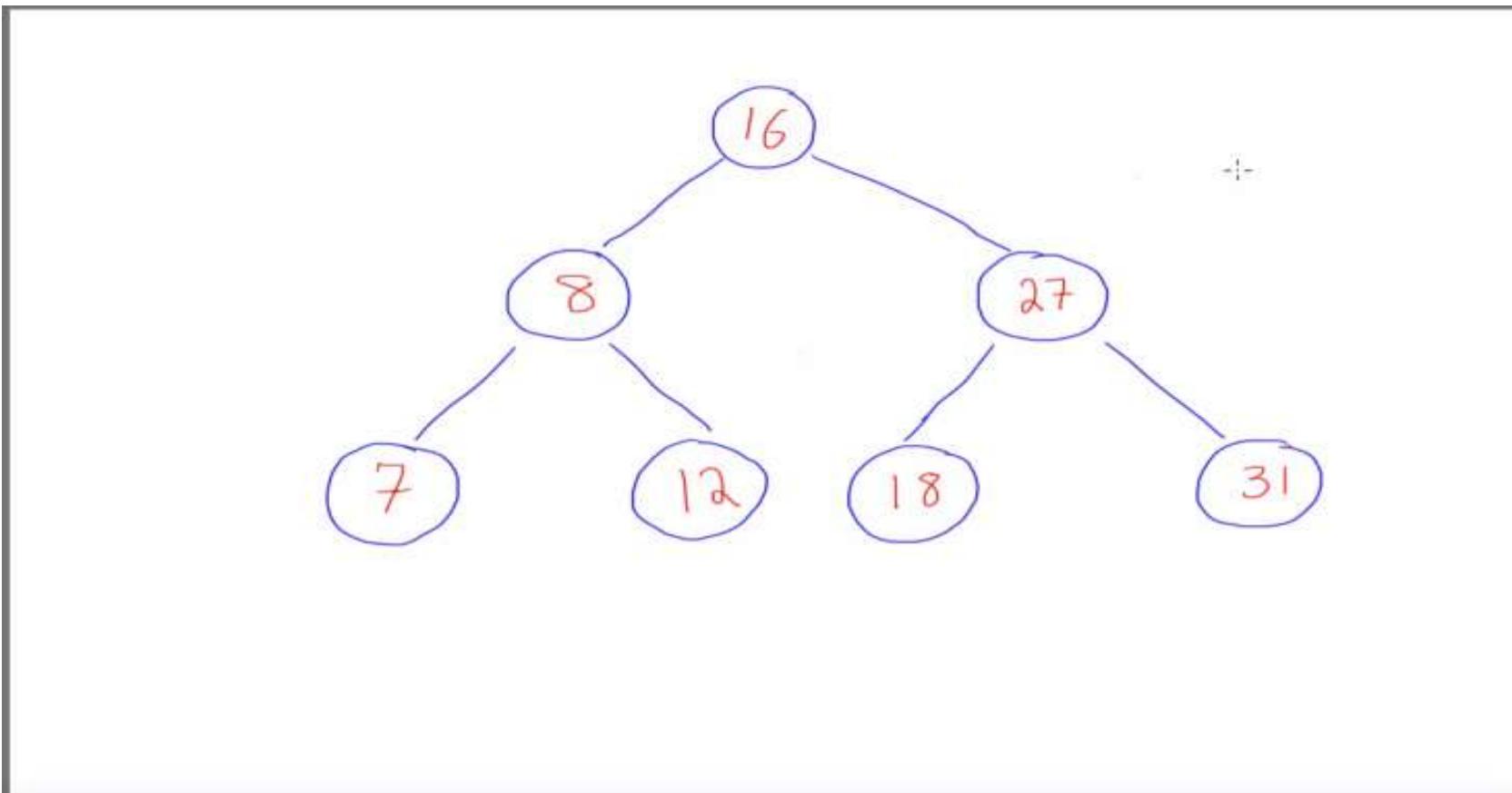


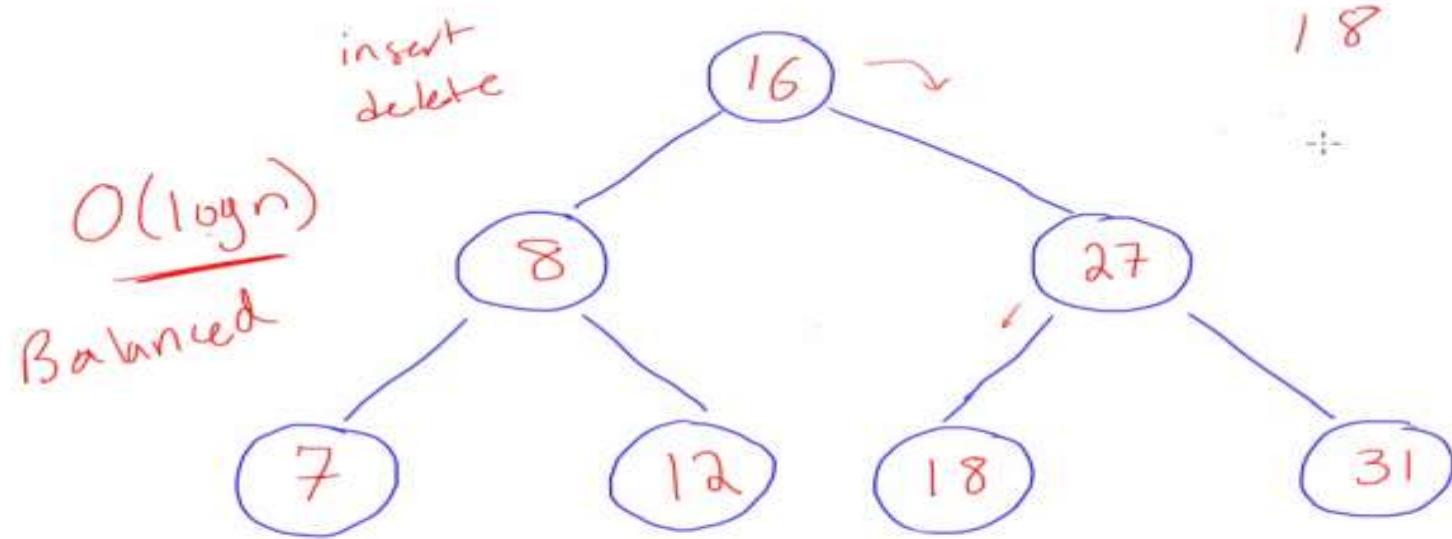
Balanced Binary Search Tree

- For all the nodes , the difference between the heights of the left and right subtrees must not be greater than one.
- Height of Binary Search Tree can be calculated by counting number of edges in longest path from the root to the leaf node.
 - *Height of Tree = Height of Root
 - *Height of Tree with 1 Node (Leaf Node) = 0
- Depth of a Node = No of edges in path from root to that Node.

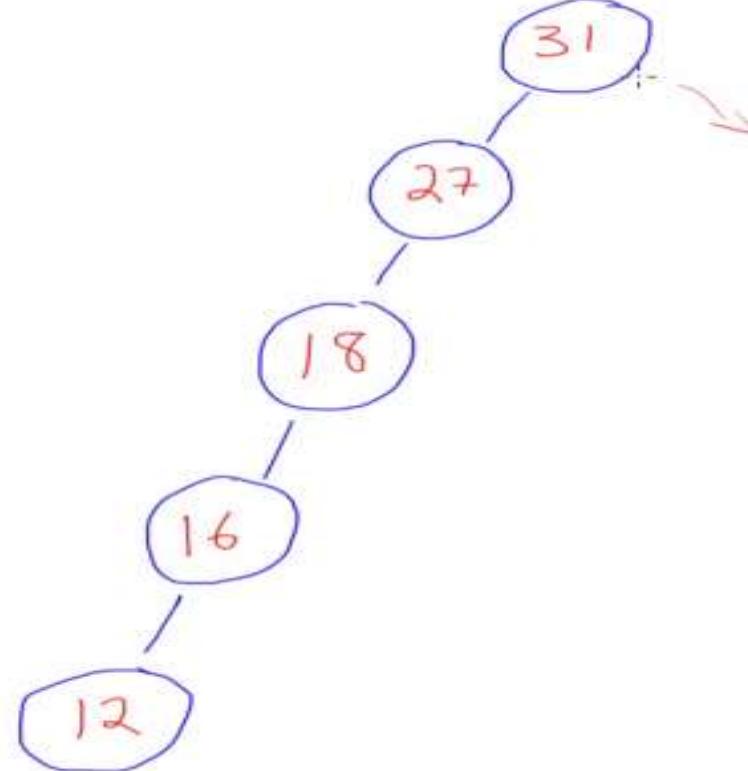


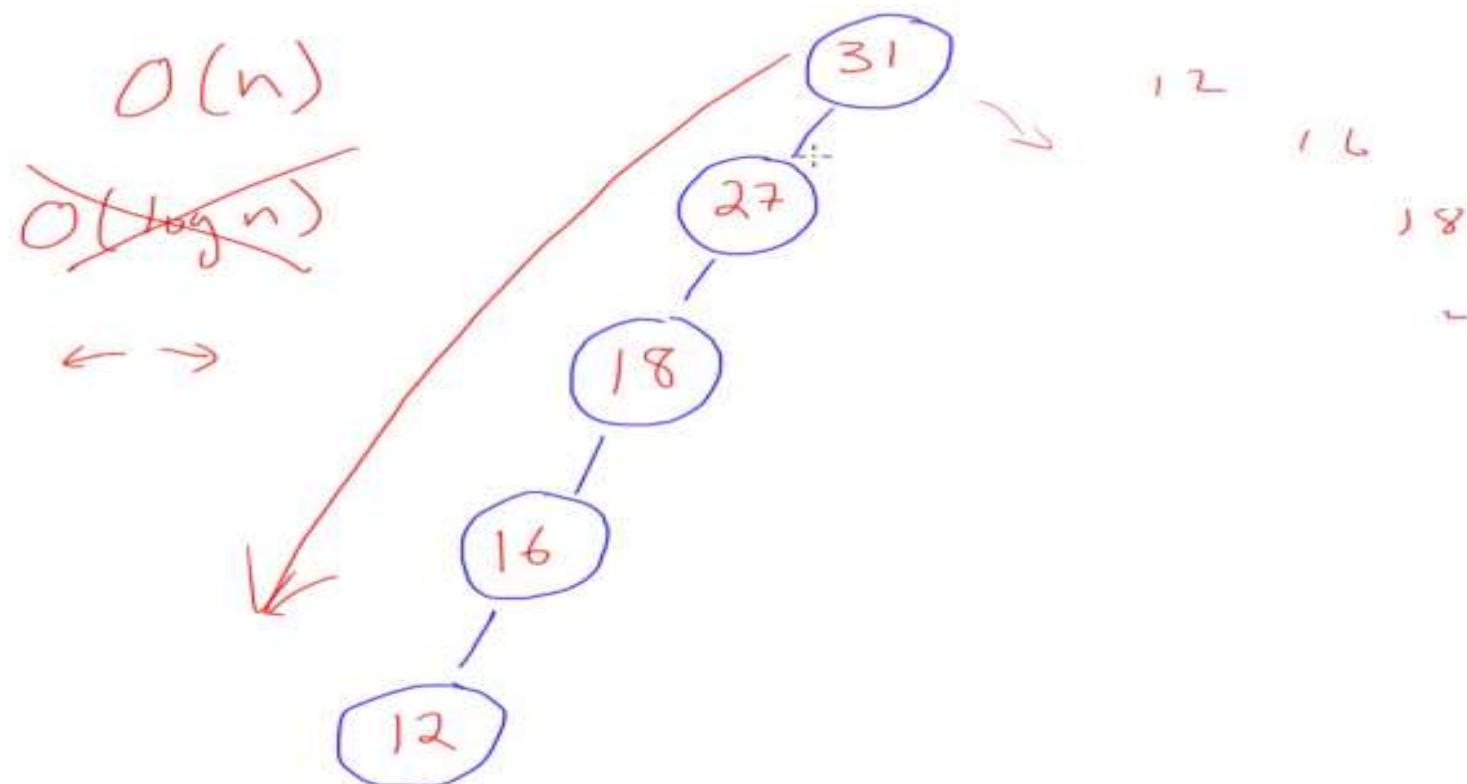
Balanced Binary Search Tree



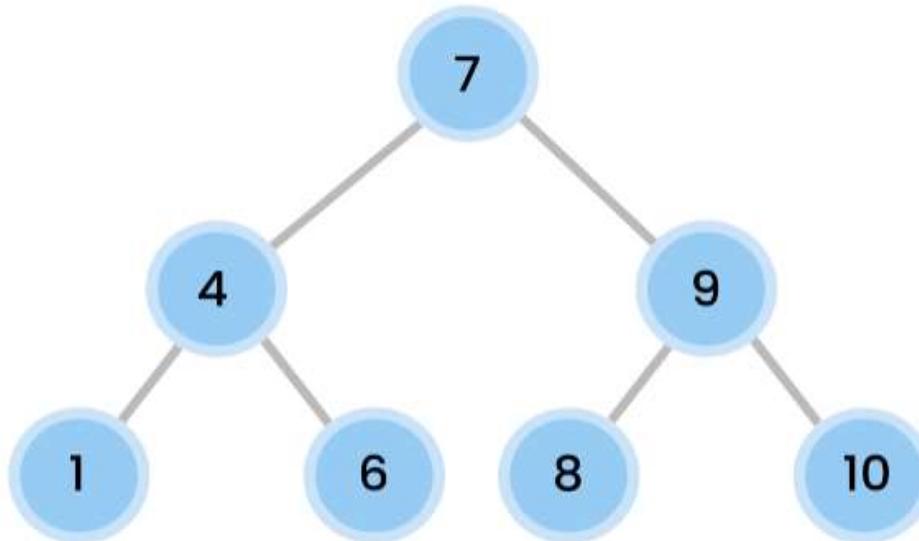


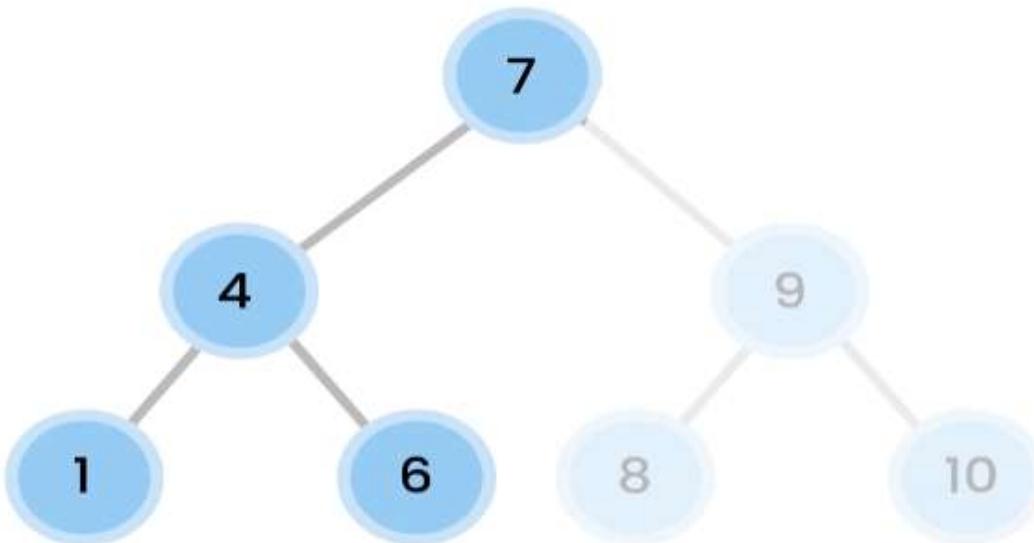
Unbalanced Binary Tree

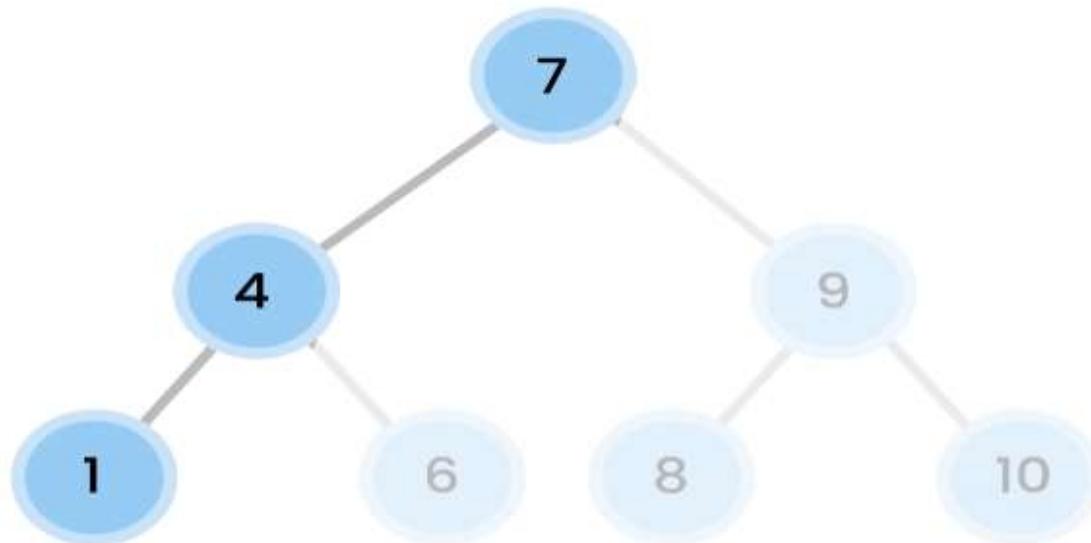




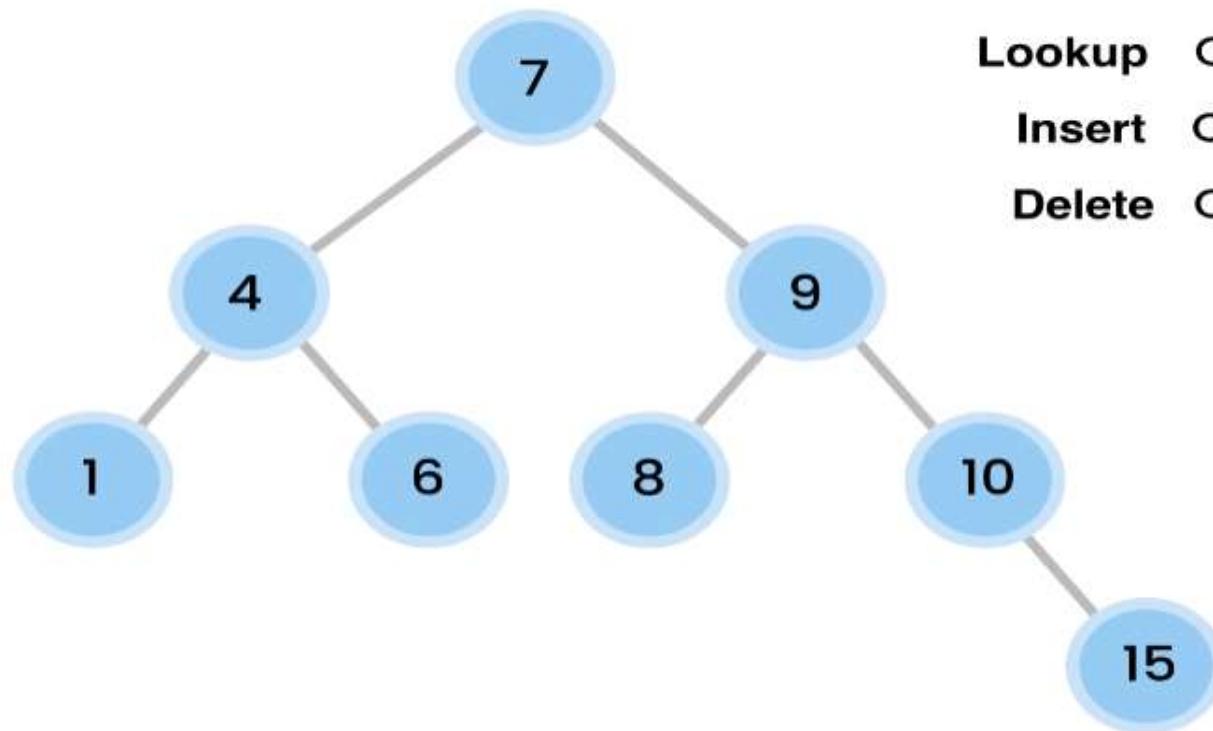
Searching 1







Runtime Complexity BST



Lookup $O(\log n)$

Insert $O(\log n)$

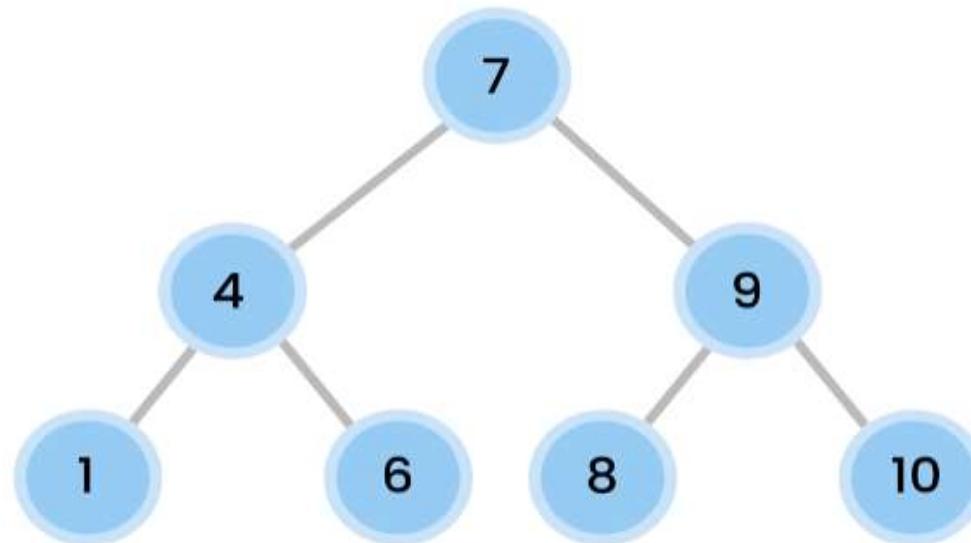
Delete $O(\log n)$

Applications

TREES

- Represent hierarchical data
- Databases
- Autocompletion
- Compilers
- Compression (JPEG, MP3)

Implementation of Following BST.



The screenshot shows an IDE interface with the following details:

- Project:** BST [C:\Users\Hassan\IdeaProjects\BST]
- File:** Tree.java
- Code Content:**

```
1 package com.company;
2
3 public class Tree {
4     private class Node{
5         private int value; // Data for Node
6         private Node leftChild;
7         private Node rightChild;
8
9         public Node(int value){
10             this.value = value;
11         }
12     }
13
14     private Node root;
15
16     public void insert(int value){
17         if(root == null){ // Means an empty BST
18             root = new Node(value);
19             return; // exit out of the function
20         }
21     }
22 }
23
24 }
```

- Toolbars and Status Bar:**
 - File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
 - Event Log
 - Build completed successfully in 4 s 818 ms (moments ago)
 - 21:10 CRLF UTF-8 4 spaces

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help BST [C:\Users\Hassan\IdeaProjects\BST] - Main.java

BST > src > com > company > Main

Main.java > Tree.java

```
package com.company;

public class Main {

    public static void main(String[] args) { args: []
        // write your code here
        Tree tree = new Tree(); tree: Tree@792
        tree.insert(value: 7); tree: Tree@792
        System.out.println("Done");
    }
}
```

Project: BST C:\Users\Hassan\IdeaProjects\BST

src .idea out com.company Main Tree BST.iml External Libraries Scratches and Consoles

Debug: Main

Frames: "main" @ 1 in group "main": RU... main:9, Main (com.company)

Variables:

- args = [String[0]@791]
- tree = (Tree@792)
- root = (TreeNode@793)
 - value = 7
 - leftChild = null
 - rightChild = null
- this\$0 = (Tree@792)

Terminal Messages Debug TODO Event Log

Build completed successfully in 4 s 818 ms (a minute ago) 9:1 CRLF UTF-8 4 spaces

The screenshot shows an IntelliJ IDEA interface with the following details:

- Project Structure:** The left sidebar shows a project named "BST" with a structure:
 - src
 - com
 - company
 - Tree
- Main.java:** A Java file containing a main method.
- Tree.java:** A Java file containing the following code:

```
private class Node{  
    private int value; // Data for Node  
    private Node leftChild;  
    private Node rightChild;  
  
    public Node(int value){  
        this.value = value;  
    }  
  
    @Override  
    public String toString(){  
        return "Node = "+value;  
    }  
  
    private Node root;  
}
```
- Debug Tool Window:** The bottom panel displays the debug session:
 - Frames:** Shows the current frame: "main" @ 1 in group "main"; RU... and the method: "main:9, Main (com.company)".
 - Variables:** Shows local variables:
 - args = {String[0]@791}
 - tree = {Tree@792}
 - root = {Tree\$Node@793} "Node = 7"
- Status Bar:** At the bottom, it says "Build completed successfully in 2 s 963 ms (moments ago)" and shows the current time as 15:36.

Insert Function

The screenshot shows an IDE interface with the following details:

- Project Structure:** The project is named "BST" and contains two files: "Main.java" and "Tree.java".
- Main.java:** Contains a main method.
- Tree.java:** Contains the implementation of the insert function.
- Code Content (Tree.java):**

```
17     }
18 }
19 private Node root;
20
21 public void insert(int value){
22     if(root == null){ // Means an empty BST
23         root = new Node(value);
24         return; // exit out of the function
25     }
26     var current = root;
27     while (true){ // Iterate the loop till we find the parent node in which value will be inserted
28         if(value<current.value){
29             if(current.leftChild == null){ // Means if LeftChild is empty / not available
30                 current.leftChild = new Node(value); // Insert New Node as Left Child
31                 break; // exit out of the loop
32             }
33             current = current.leftChild; // Set current Node to the Left Node (We Go One Level Down)
34         }
35         else{
36             if(current.rightChild == null){
37                 current.rightChild = new Node(value);
38                 break;
39             }
40             current = current.rightChild;
41         }
42     }
43 }
44 }
```

- Toolbars and Status Bar:** The status bar at the bottom shows "Build completed successfully in 1 s 599 ms (moments ago)".
- Bottom Right Corner:** Shows the date and time as "40:46 CRLF UTF-8 4 spaces".

Test

The screenshot shows an IDE interface with the following details:

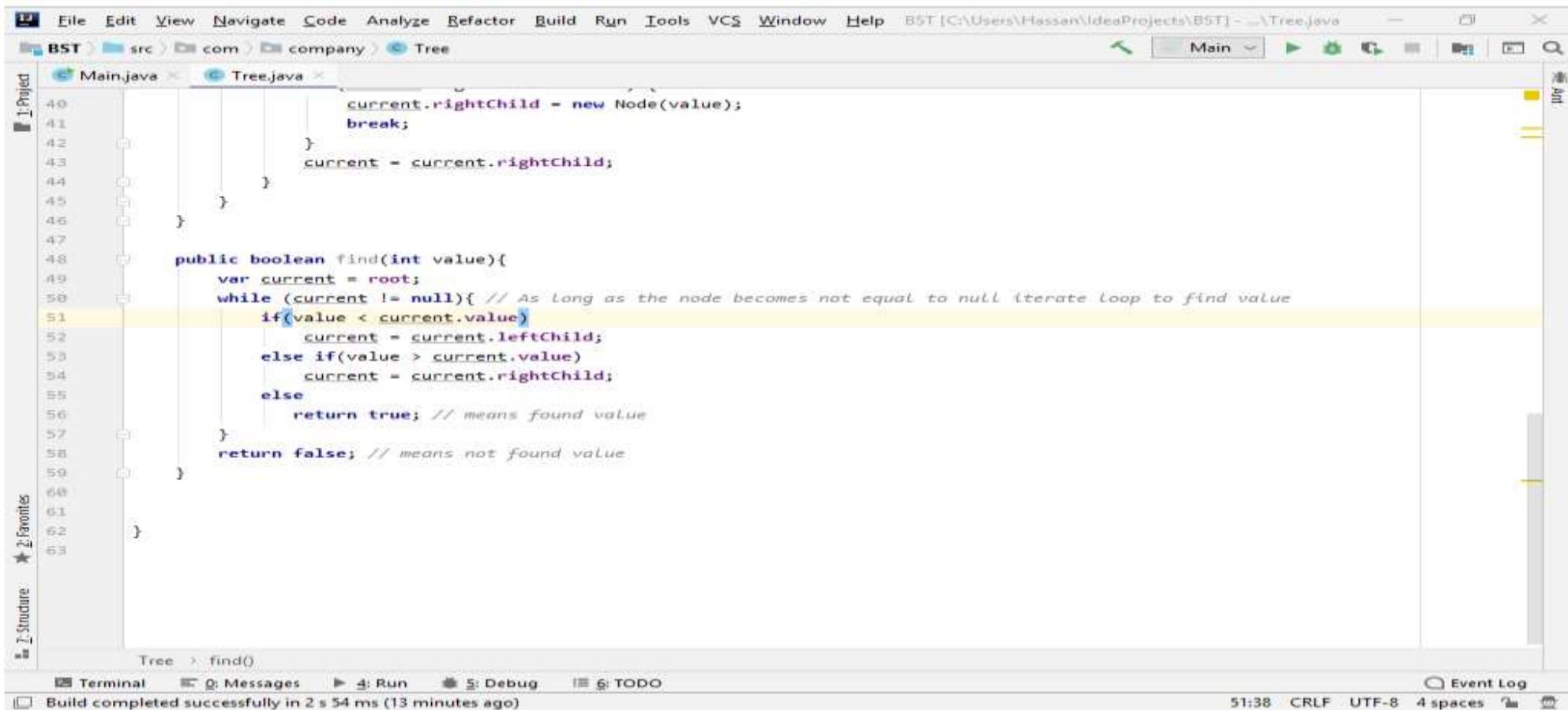
- Project Structure:** The project is named "BST" and contains a package "com.company" which has a file "Main.java".
- Main.java Content:**

```
package com.company;

public class Main {

    public static void main(String[] args) {
        // write your code here
        Tree tree = new Tree();
        tree.insert( value: 7);
        tree.insert( value: 4);
        tree.insert( value: 9);
        tree.insert( value: 1);
        tree.insert( value: 6);
        tree.insert( value: 8);
        tree.insert( value: 10);
        System.out.println("Done");
    }
}
```
- Toolbars and Menus:** Standard IDE menus like File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help are visible at the top.
- Status Bar:** The status bar at the bottom shows "Build completed successfully in 1 s 599 ms (a minute ago)" and "15:1 CRLF UTF-8 4 spaces".
- Bottom Icons:** Icons for Terminal, Messages, Debug, TODO, Event Log, and other tools are present.

Find function



```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help BST [C:\Users\Hassen\IdeaProjects\BST] - Tree.java
BST src com company Tree
Main.java Tree.java
1:Project
2:Favorites
3:Structure
Tree > find()
4:Terminal 5:Messages 6:Run 7:Debug 8:TODO 9:Event Log
Build completed successfully in 2 s 54 ms (13 minutes ago) 51:38 CRLF UTF-8 4 spaces
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
public boolean find(int value){
    var current = root;
    while (current != null){ // As Long as the node becomes not equal to null iterate loop to find value
        if(value < current.value)
            current = current.leftChild;
        else if(value > current.value)
            current = current.rightChild;
        else
            return true; // means found value
    }
    return false; // means not found value
}
```

Test

Screenshot of IntelliJ IDEA showing a Java project named "BST". The Main.java file contains the following code:

```
package com.company;

public class Main {
    public static void main(String[] args) {
        // write your code here
        Tree tree = new Tree();
        tree.insert( value: 7);
        tree.insert( value: 4);
        tree.insert( value: 9);
        tree.insert( value: 1);
        tree.insert( value: 6);
        tree.insert( value: 8);
        tree.insert( value: 10);

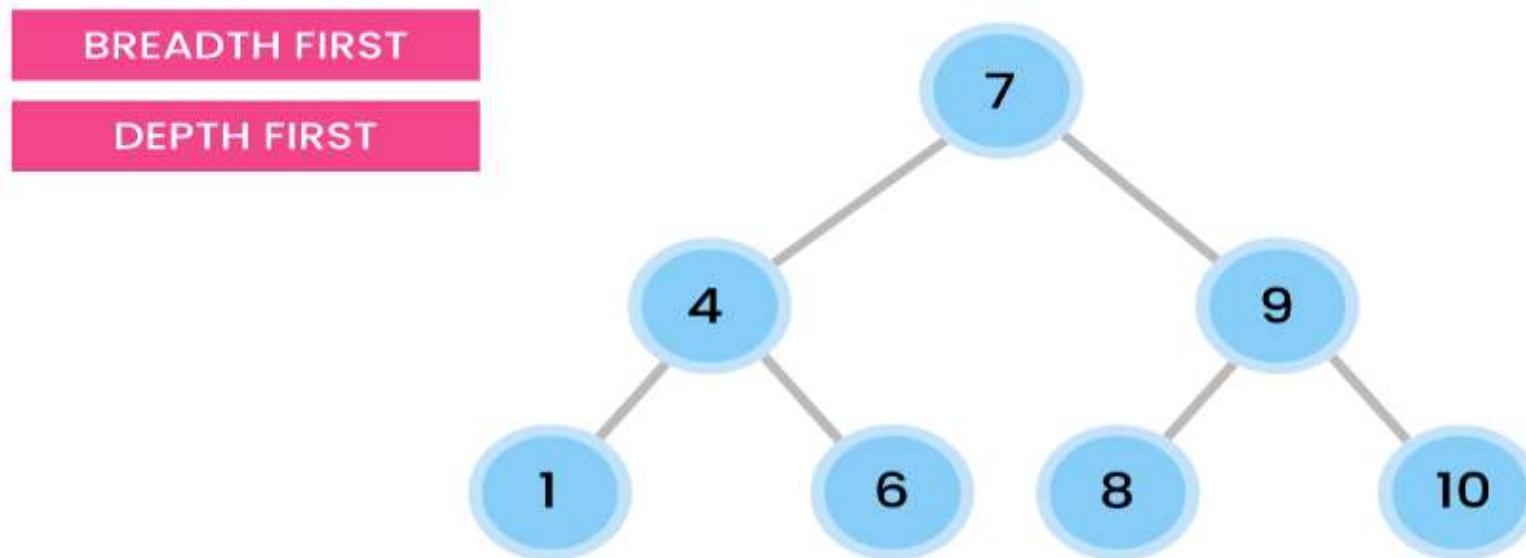
        System.out.println(tree.find( value: 8));
    }
}
```

The Run tab shows the output of the program:

```
"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019.3.3\lib\idea_rt.jar" -Dfile.encoding=UTF-8 Main
true
Process finished with exit code 0
```

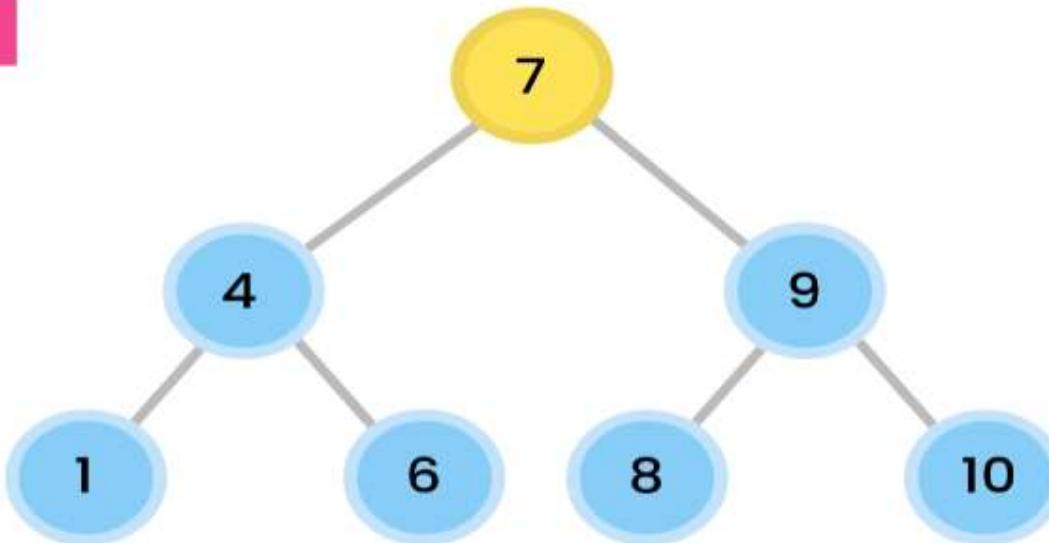
Build completed successfully in 1 s 702 ms (moments ago)

Two Types Of Traversal



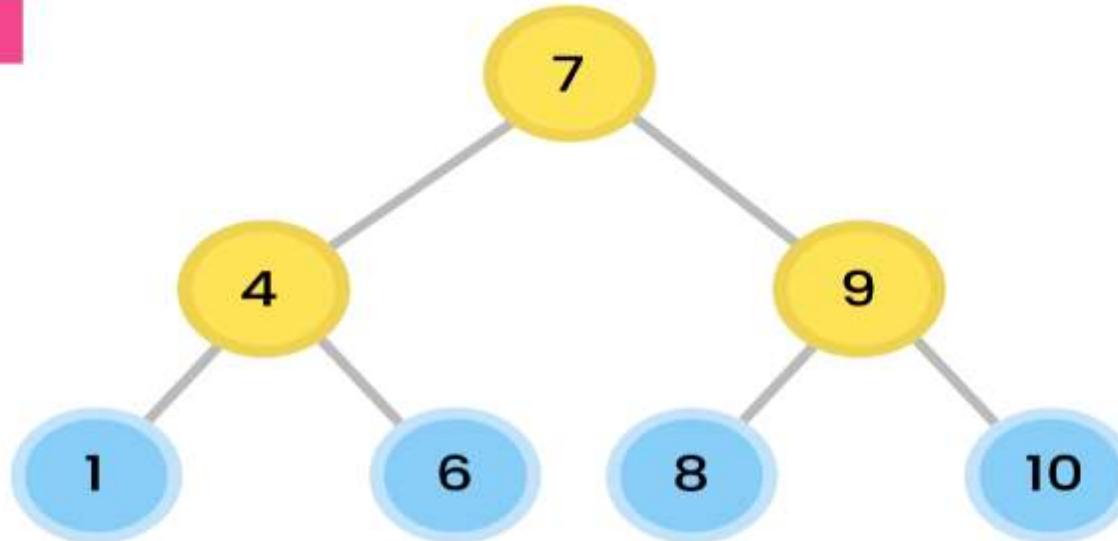
BREADTH FIRST

Level Order



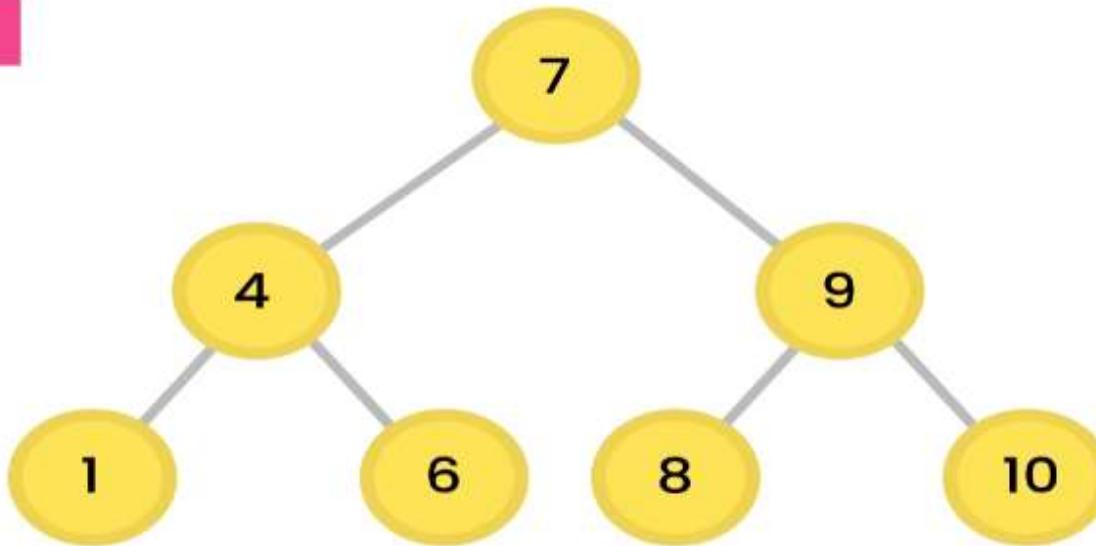
BREADTH FIRST

Level Order



BREADTH FIRST

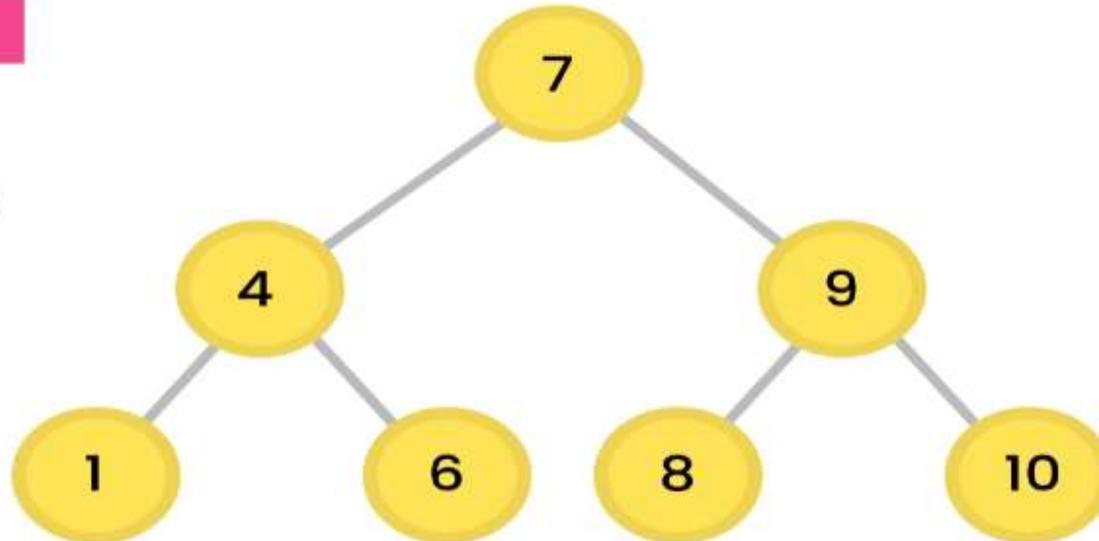
Level Order



BREADTH FIRST

Level Order

7, 4, 9, 1, 6, 8, 10



Three Types Of Depth First Traversal

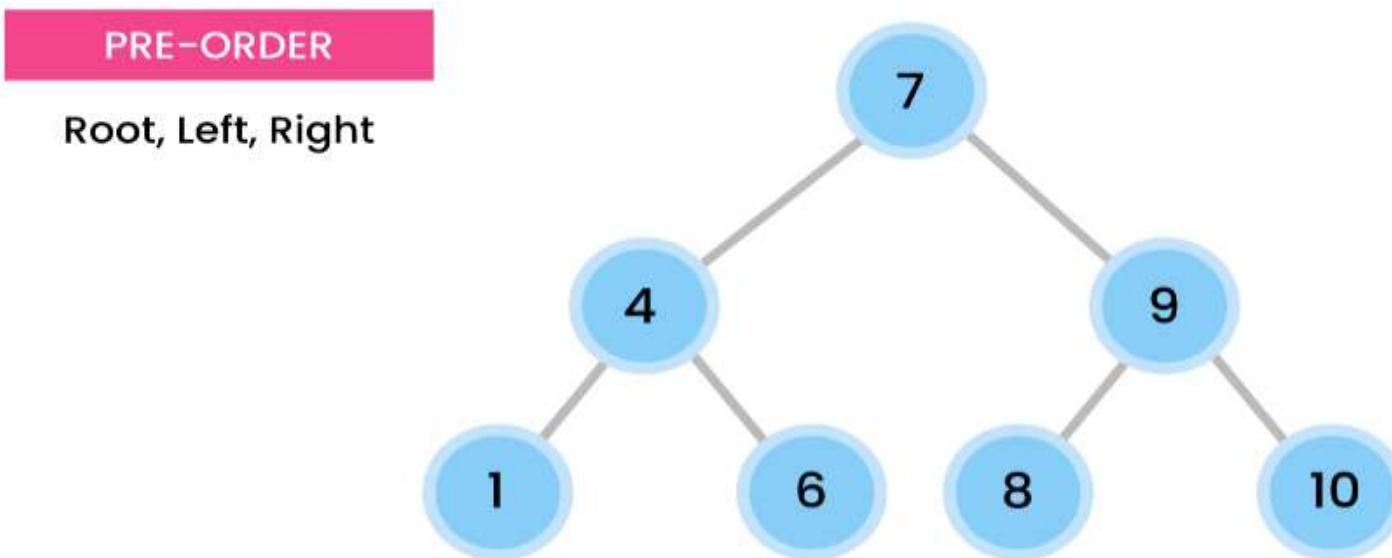
DEPTH FIRST

Pre-order Root, Left, Right

In-order Left, Root, Right

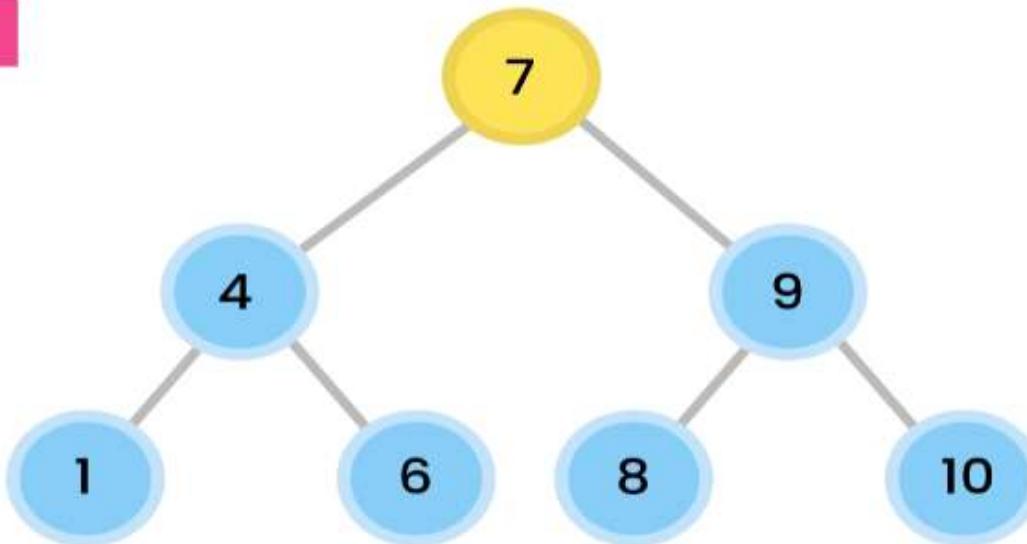
Post-order Left, Right, Root

Pre-Order Traversal



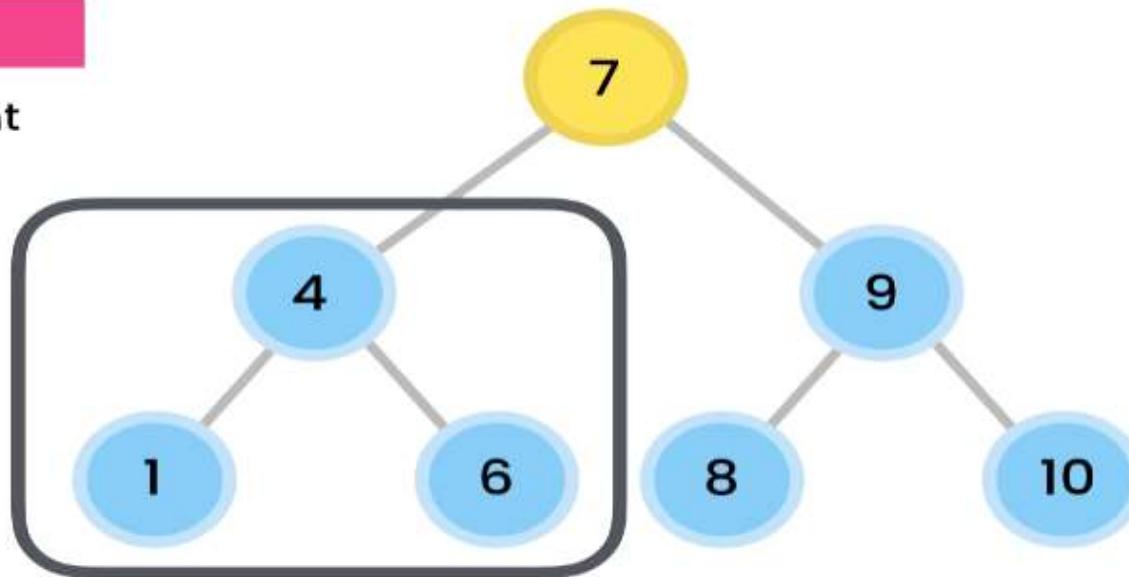
PRE-ORDER

Root, Left, Right



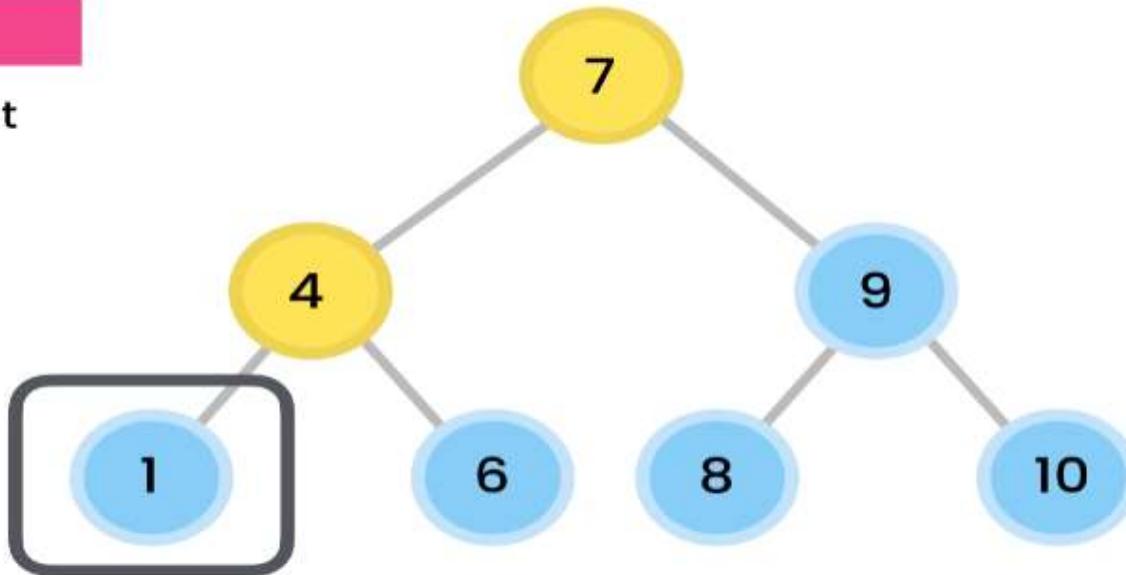
PRE-ORDER

Root, Left, Right



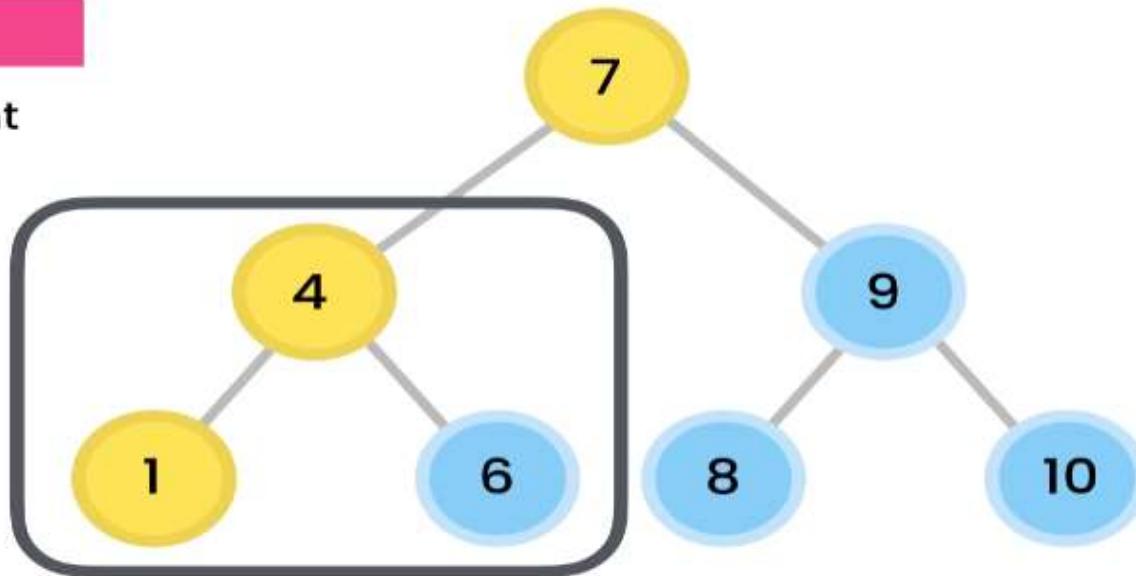
PRE-ORDER

Root, Left, Right



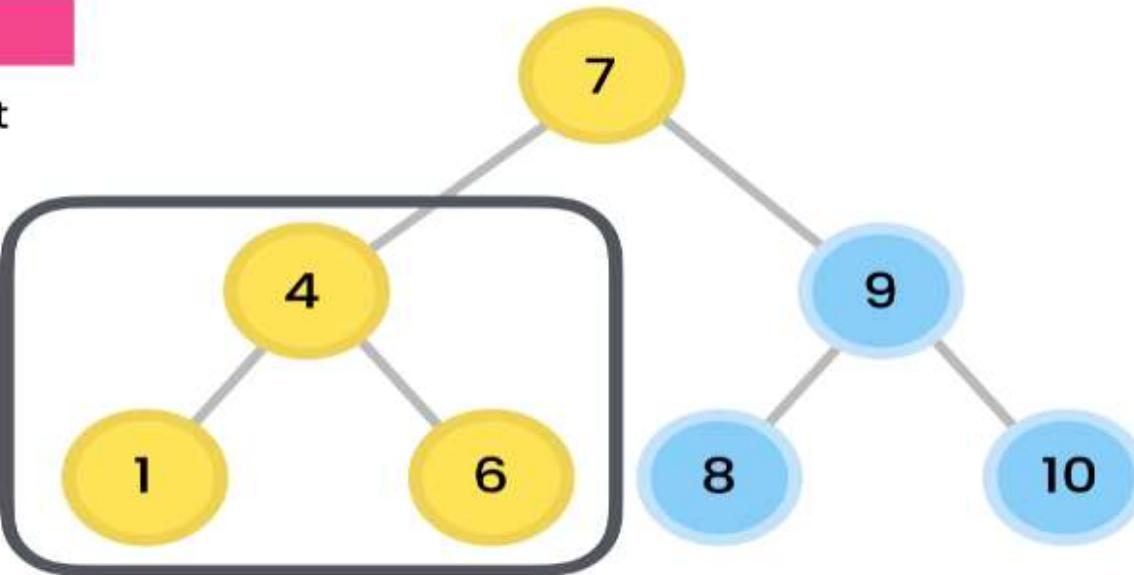
PRE-ORDER

Root, Left, Right



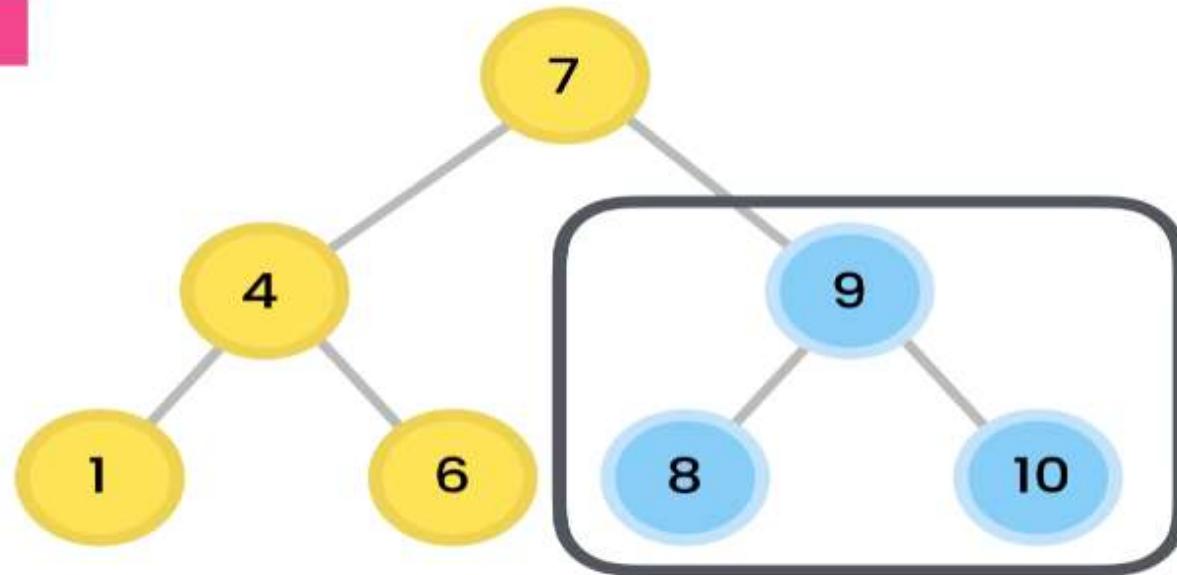
PRE-ORDER

Root, Left, Right



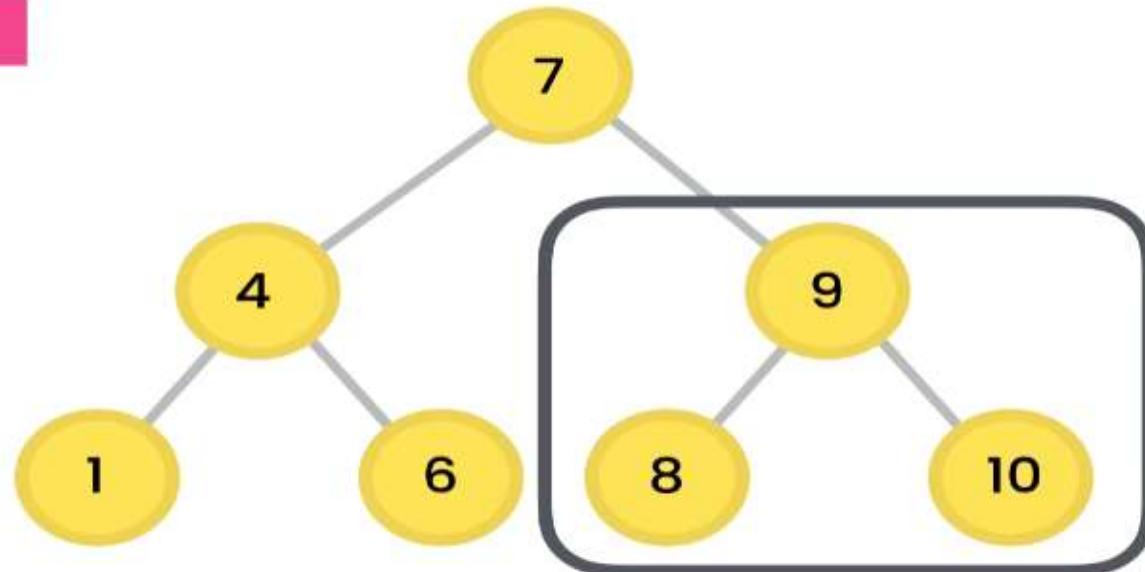
PRE-ORDER

Root, Left, Right



PRE-ORDER

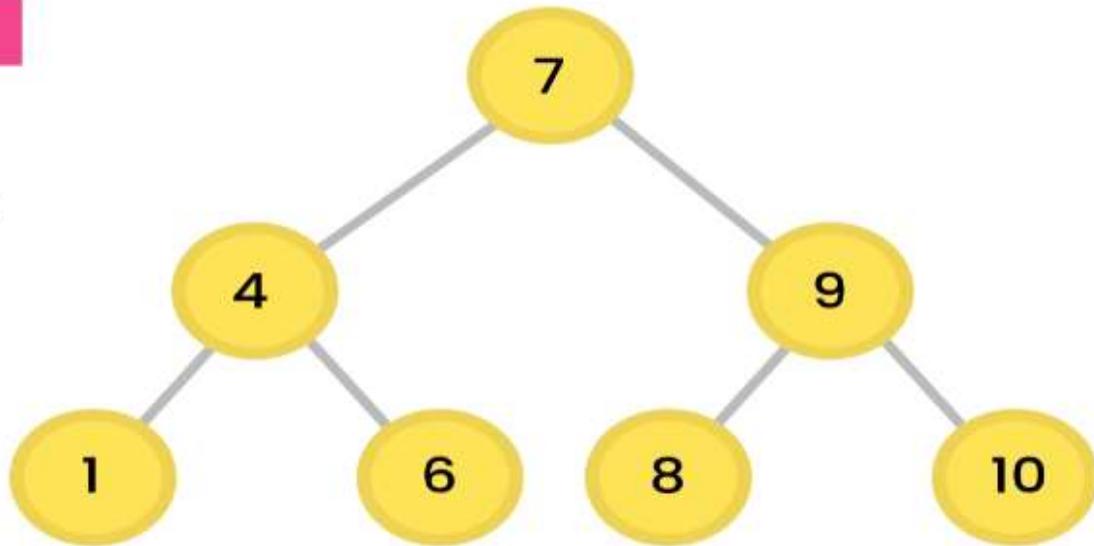
Root, Left, Right



PRE-ORDER

Root, Left, Right

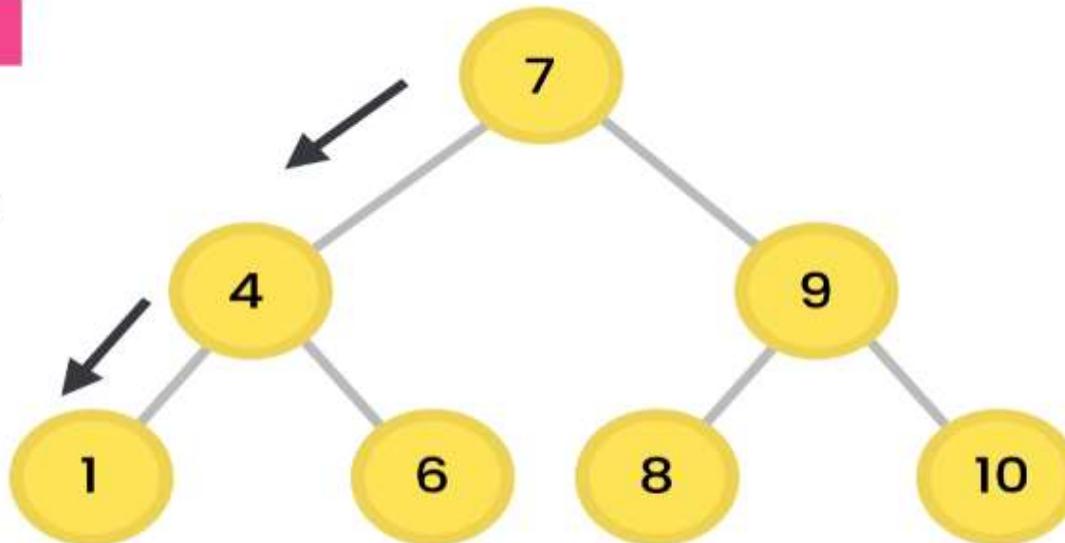
7, 4, 1, 6, 9, 8, 10



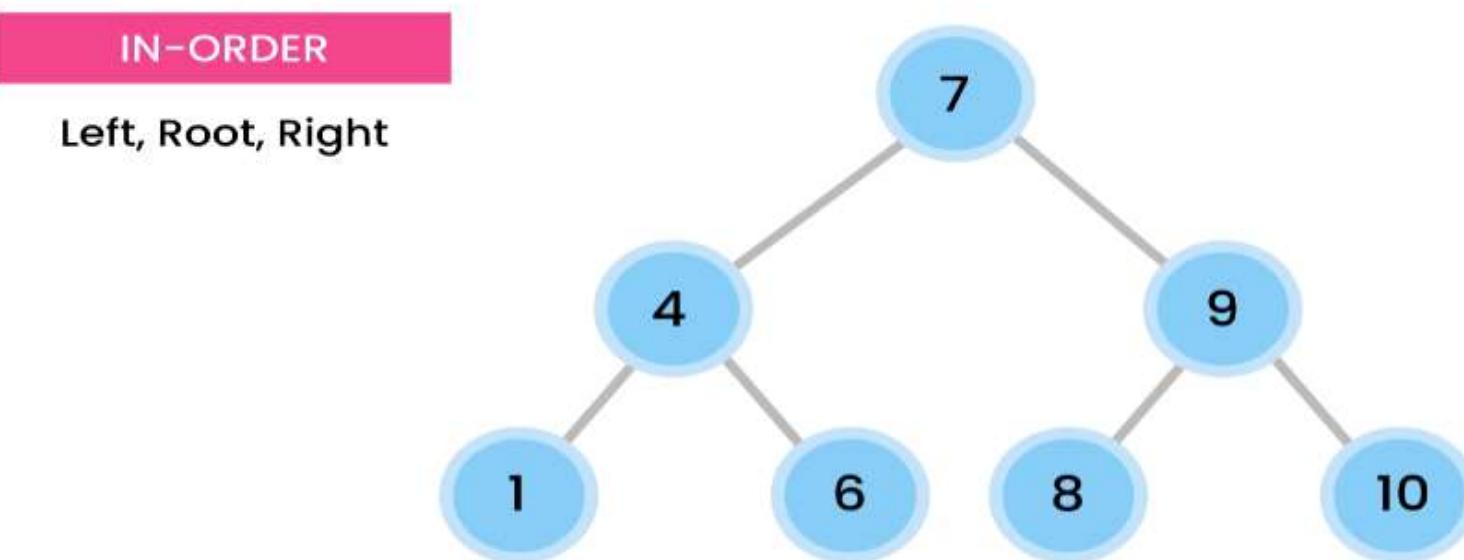
PRE-ORDER

Root, Left, Right

7, 4, 1, 6, 9, 8, 10

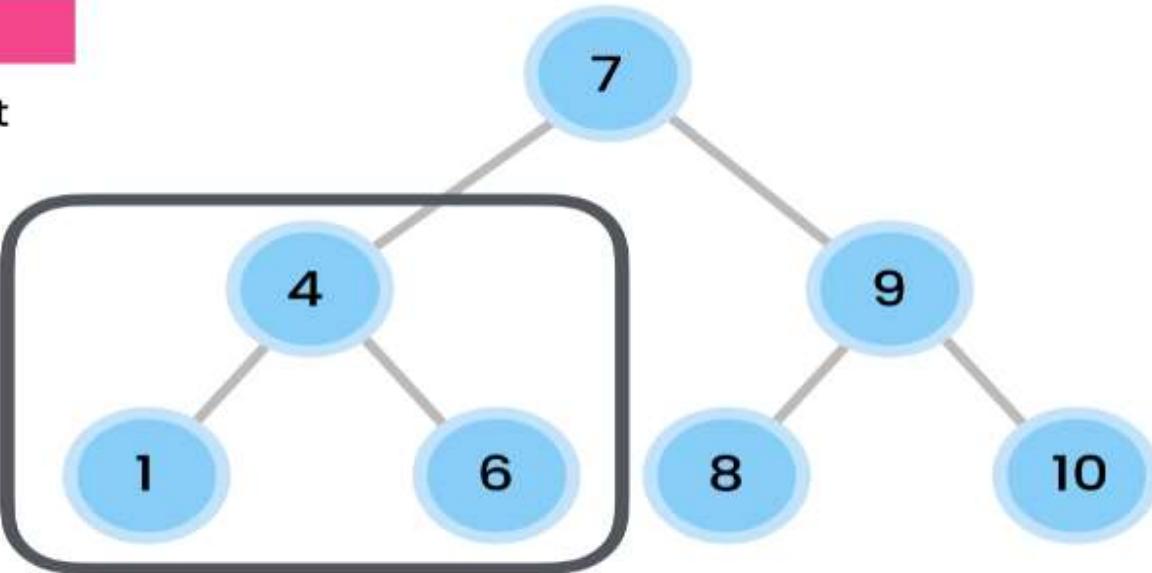


In-Order Traversal



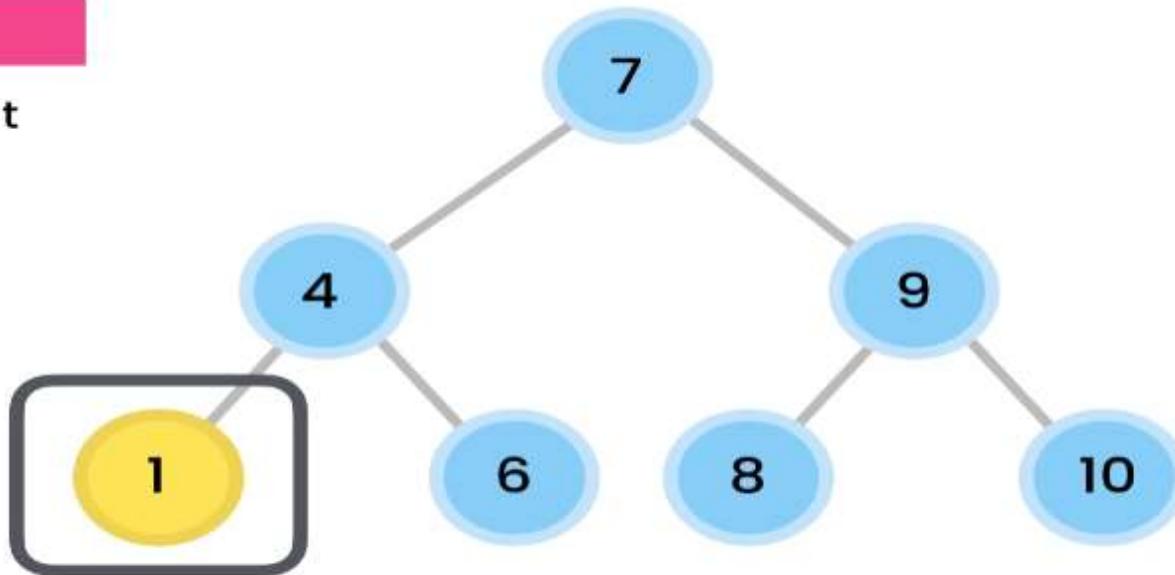
IN-ORDER

Left, Root, Right



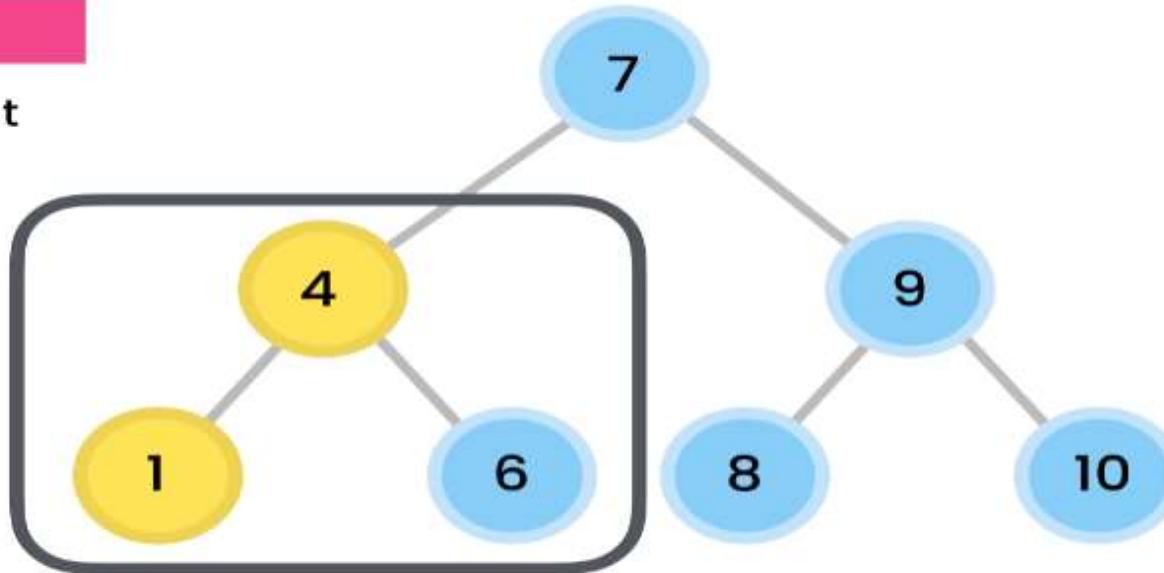
IN-ORDER

Left, Root, Right



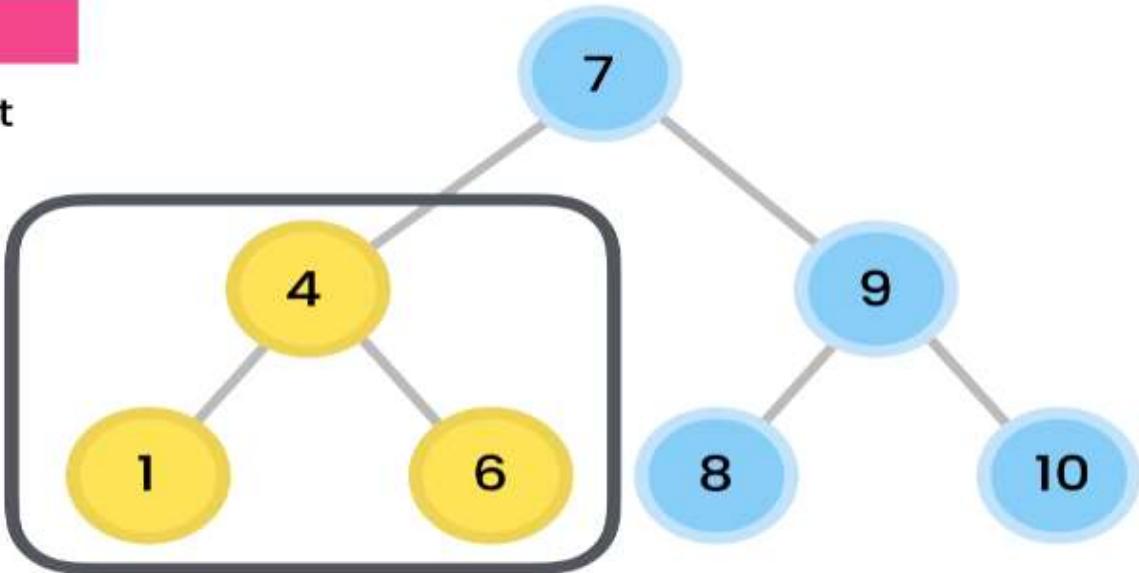
IN-ORDER

Left, Root, Right



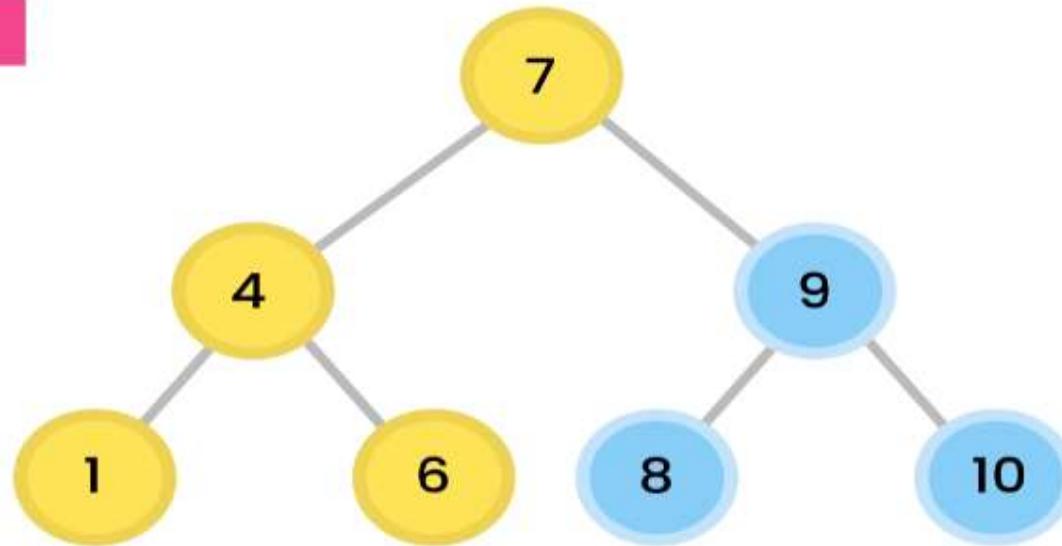
IN-ORDER

Left, Root, Right



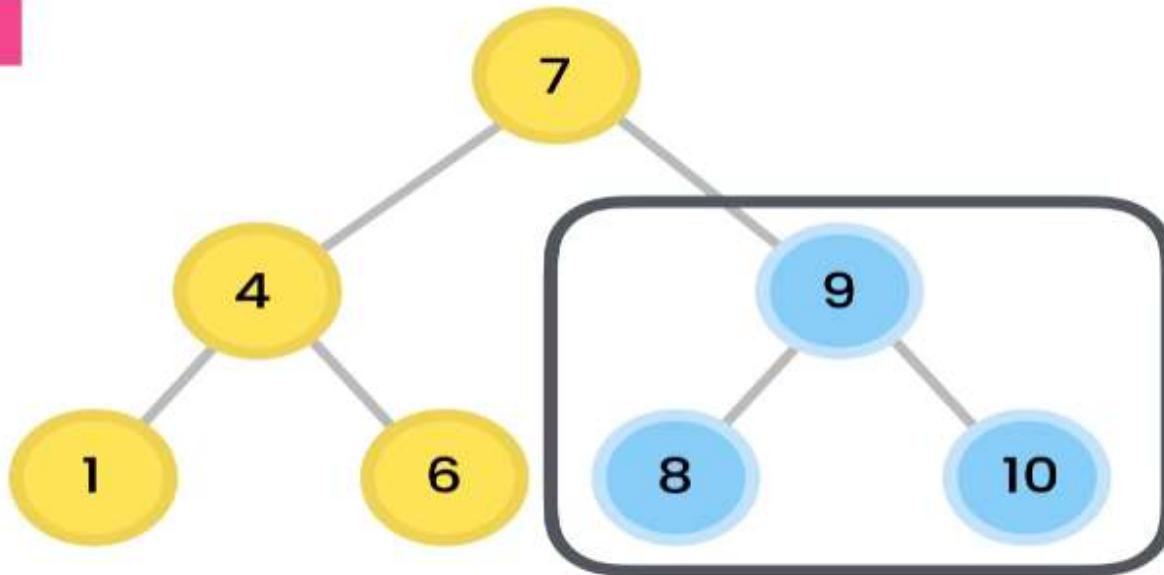
IN-ORDER

Left, Root, Right



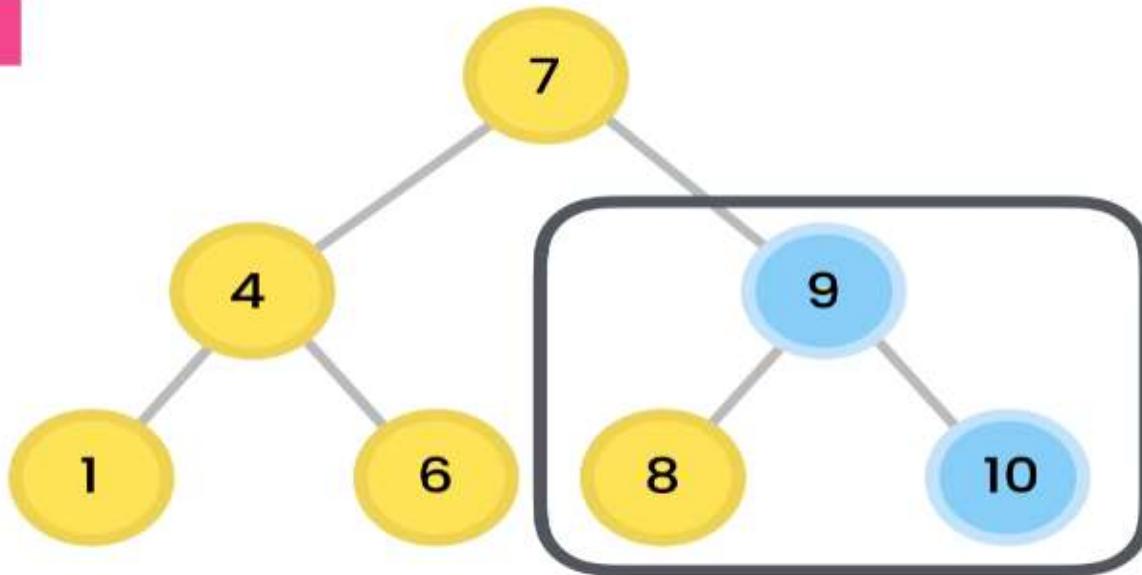
IN-ORDER

Left, Root, Right



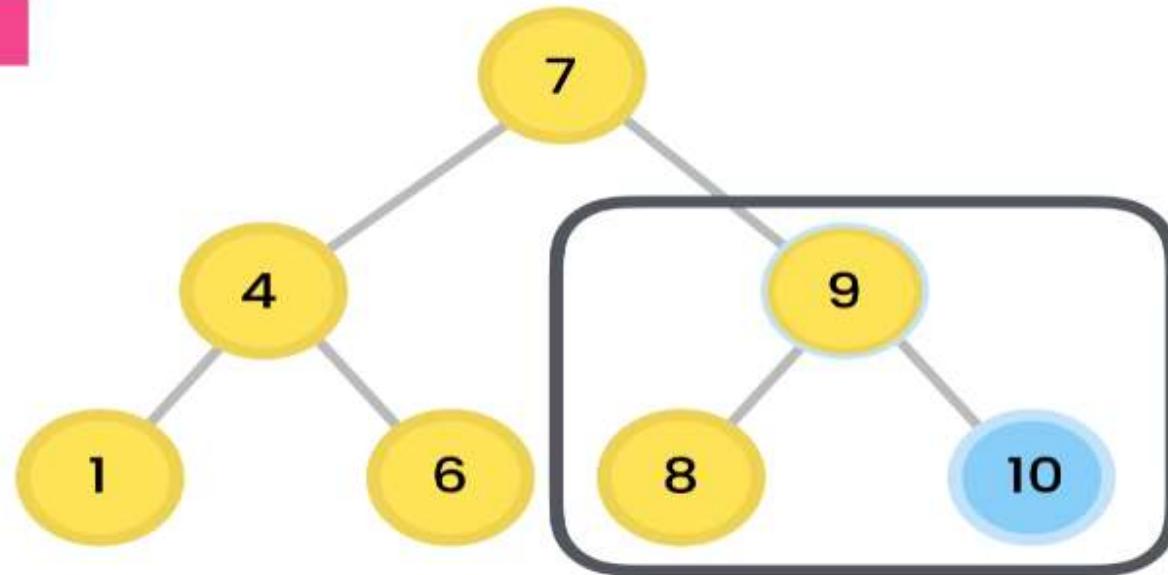
IN-ORDER

Left, Root, Right



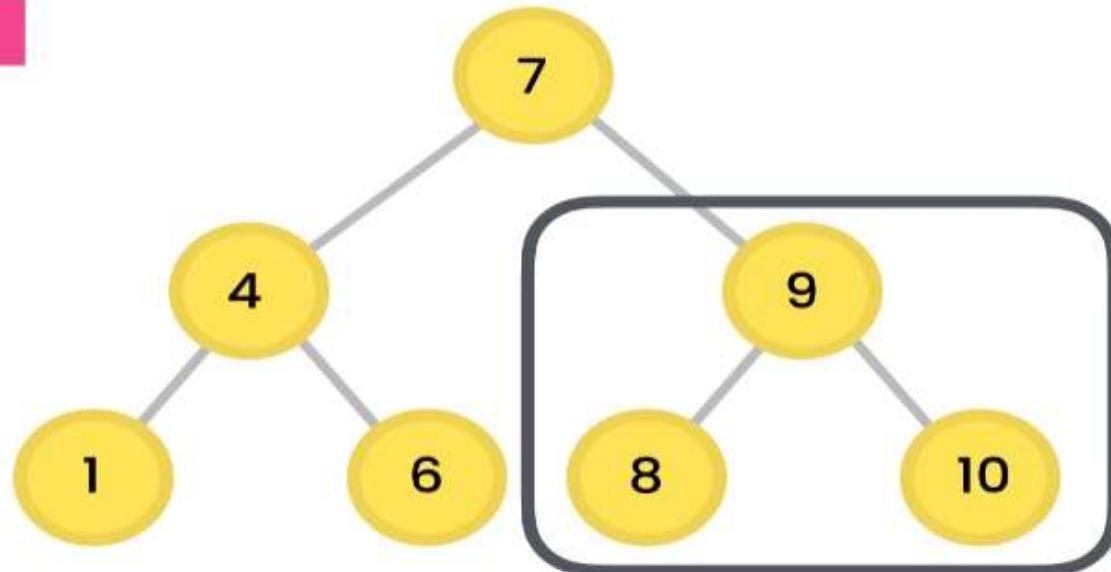
IN-ORDER

Left, Root, Right



IN-ORDER

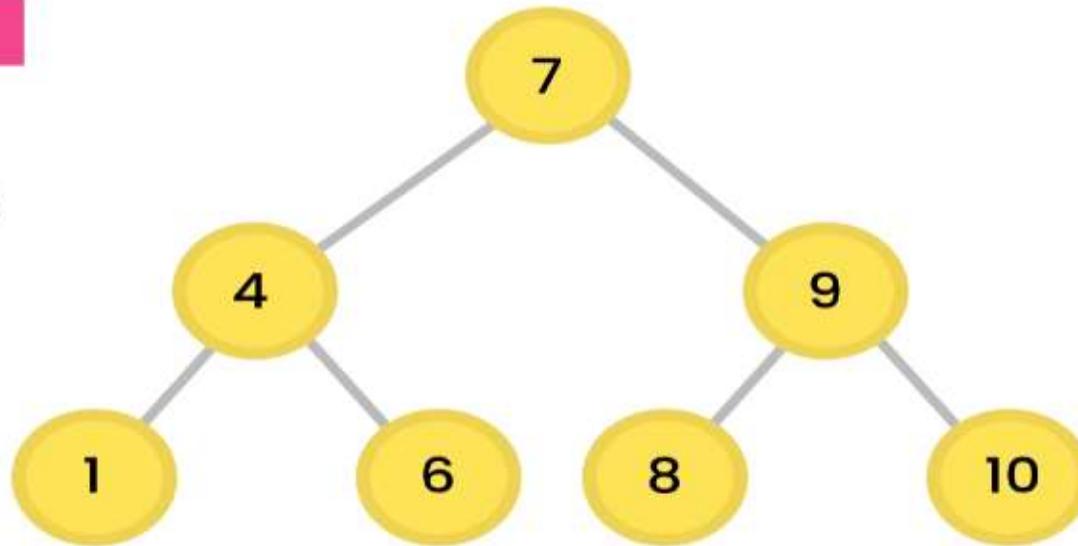
Left, Root, Right



IN-ORDER

Left, Root, Right

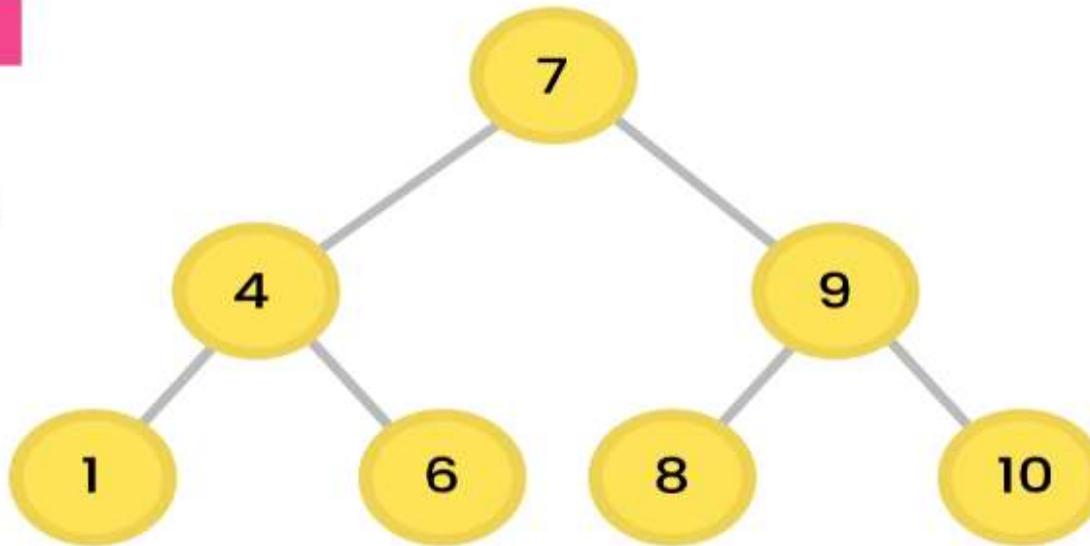
1, 4, 6, 7, 8, 9, 10



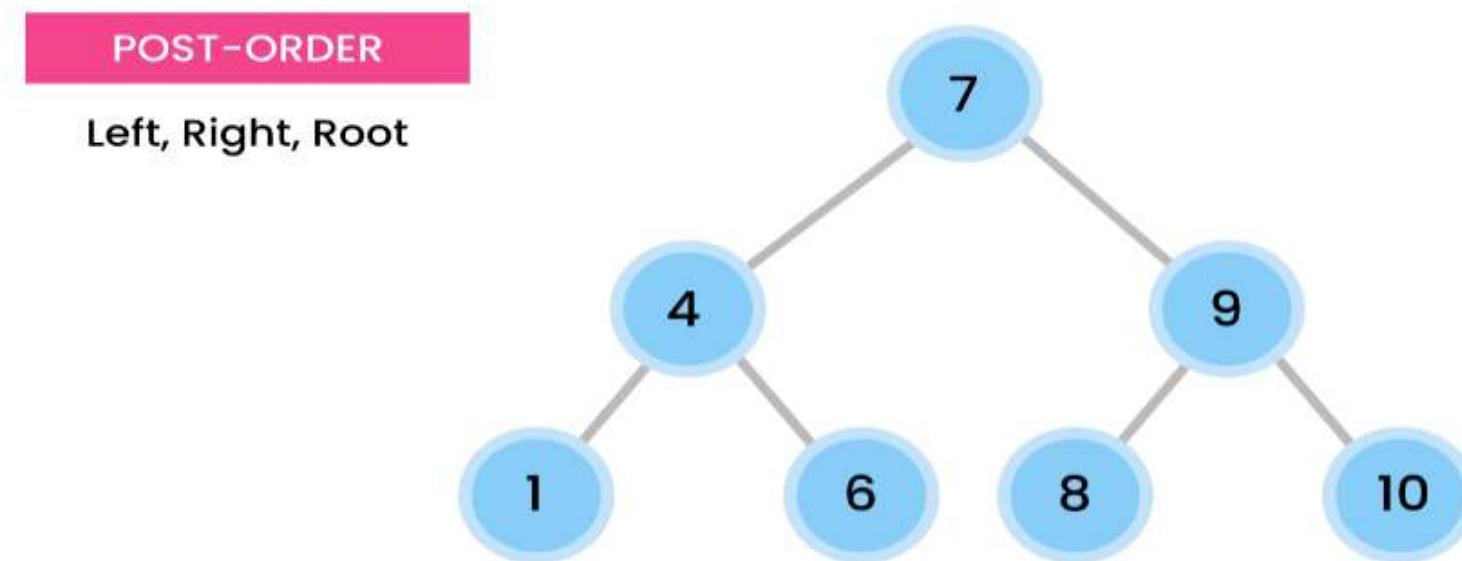
IN-ORDER

Right, Root, Left

10, 9, 8, 7, 6, 4, 1

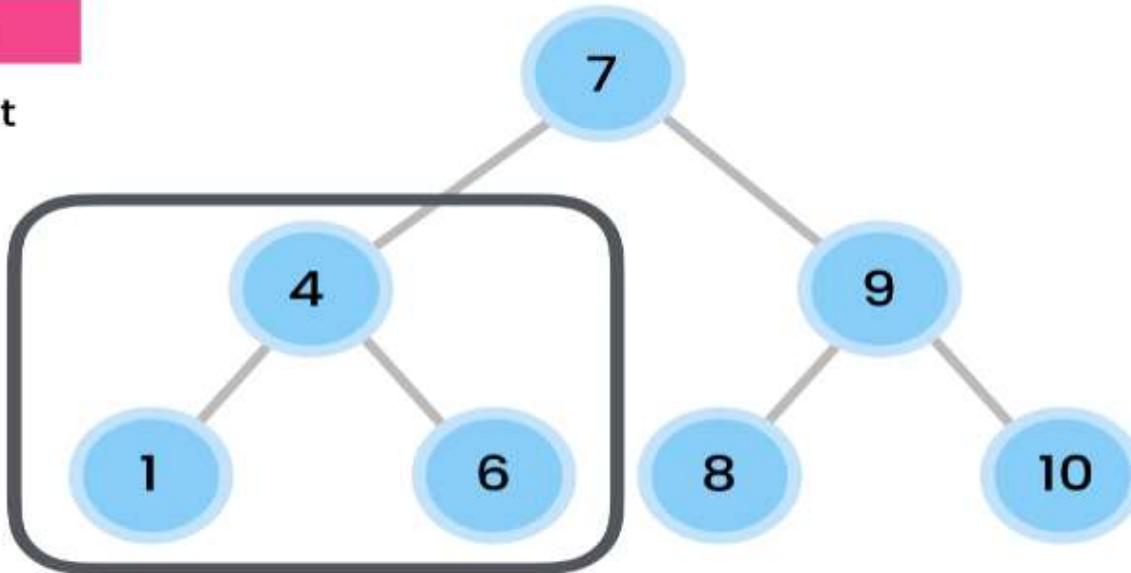


Post-Order Traversal



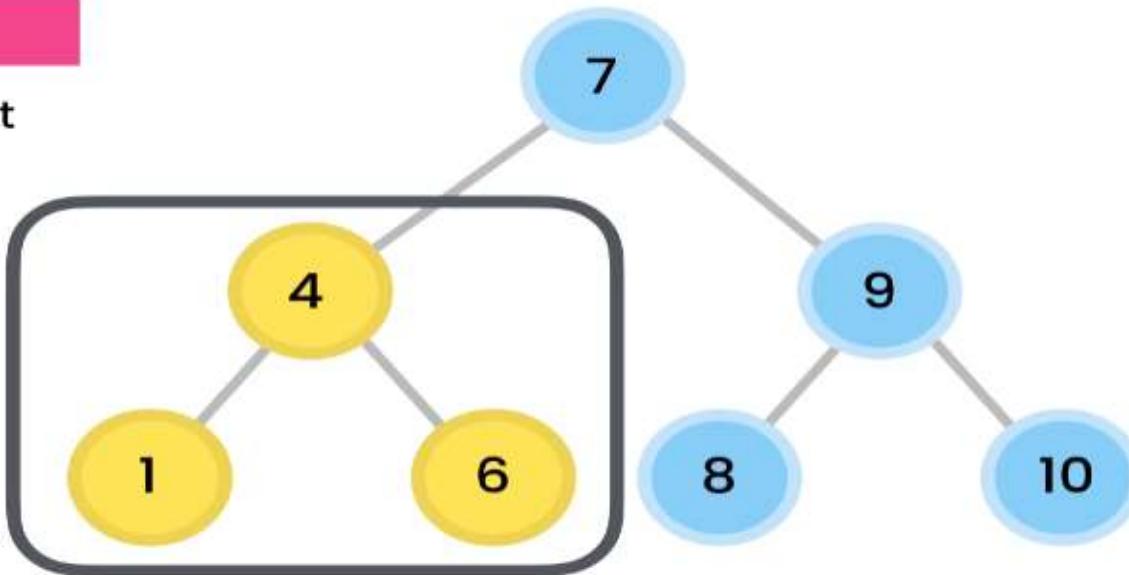
POST-ORDER

Left, Right, Root



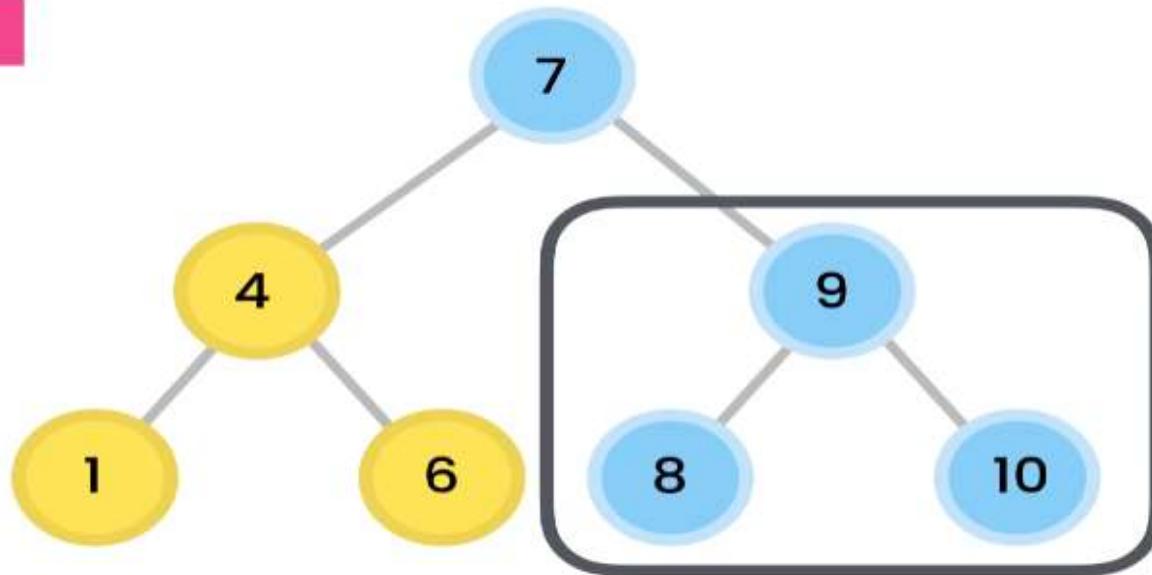
POST-ORDER

Left, Right, Root



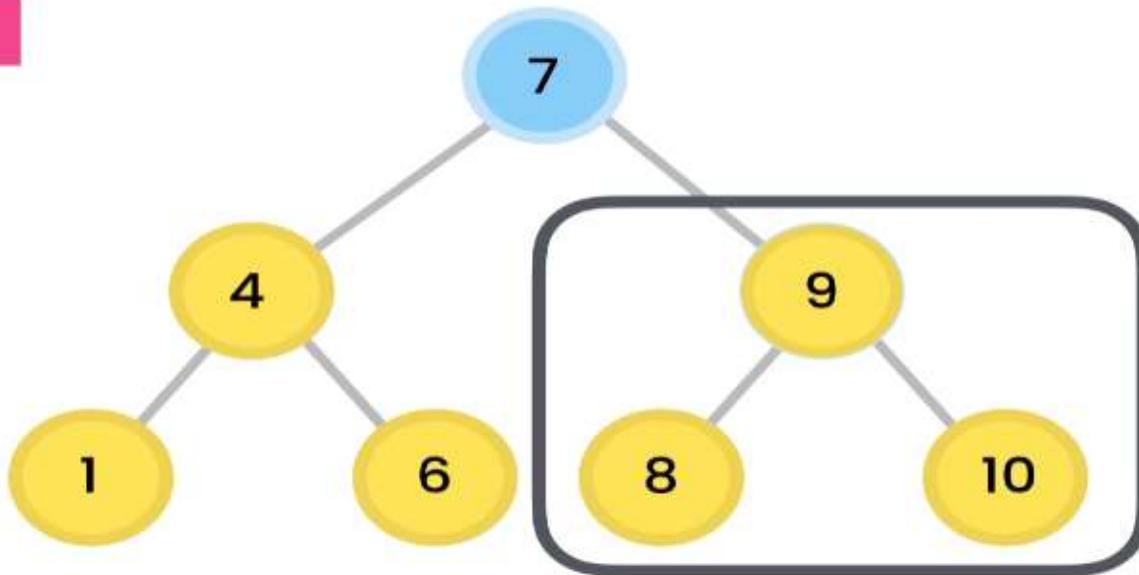
POST-ORDER

Left, Right, Root



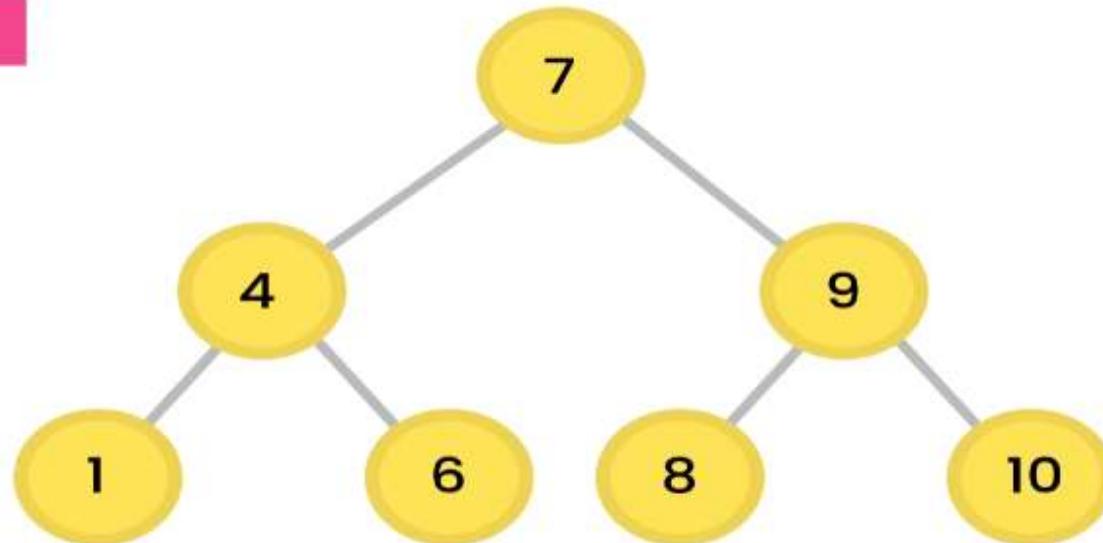
POST-ORDER

Left, Right, Root



POST-ORDER

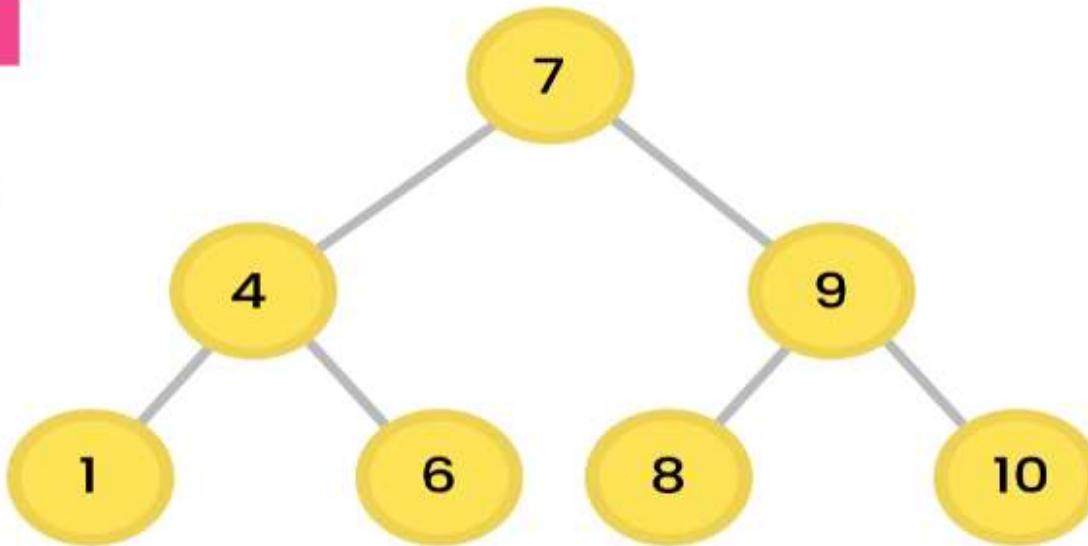
Left, Right, Root



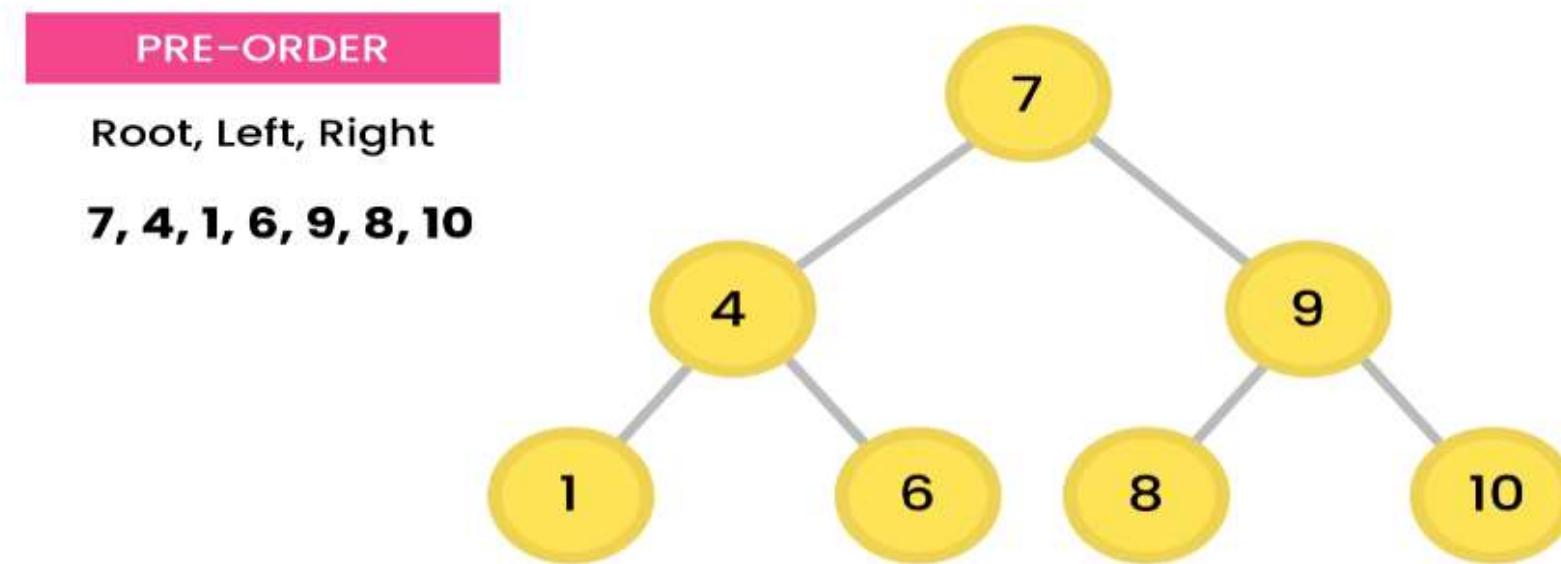
POST-ORDER

Left, Right, Root

1, 6, 4, 8, 10, 9, 7



Implementation Of Following Algorithm

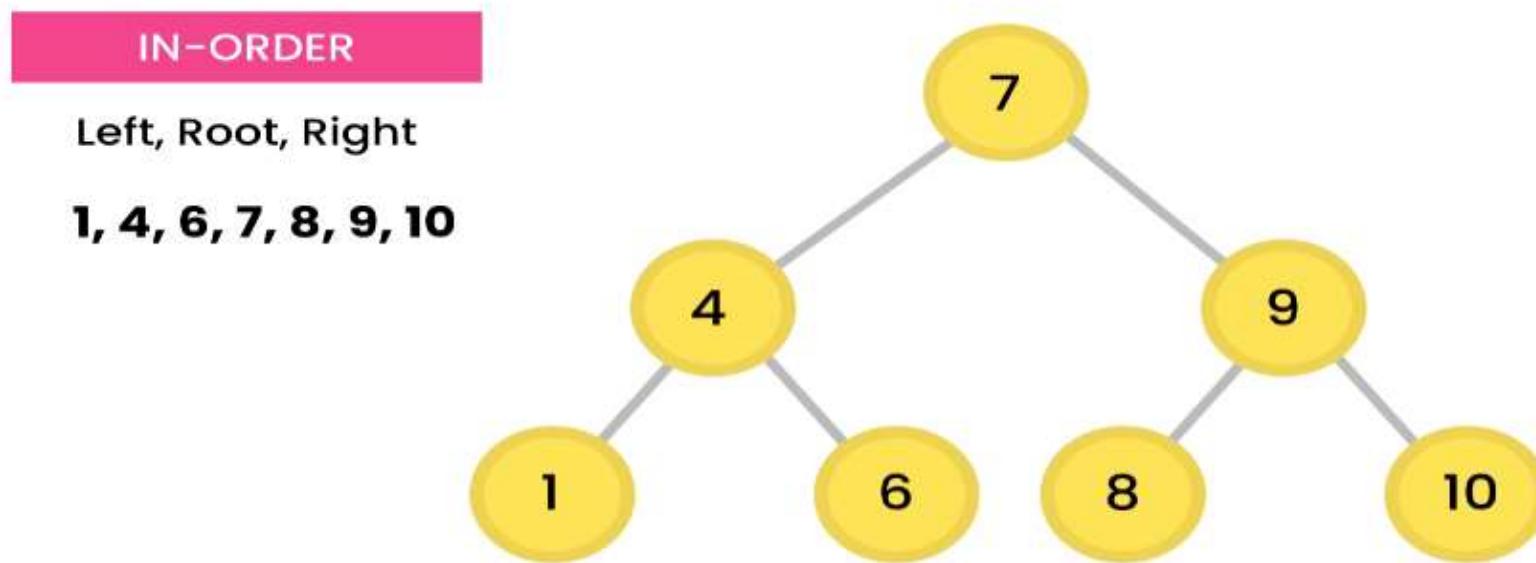


Pre-Order Using Recursion

The screenshot shows the IntelliJ IDEA IDE interface with the following details:

- Project Structure:** The project is named "BST". It contains a "src" directory with a "com" package, which further contains a "company" package. Inside "company", there are "Main.java" and "Tree.java". There is also a "BST.iml" file.
- Main.java:** Contains a static method "main" that creates a tree and performs a pre-order traversal.
- Tree.java:** Contains the definition of a binary tree node and its traversal methods.
 - Node Class:** A class with fields "value", "leftChild", and "rightChild". It has a constructor and a method "traverseInOrder" which prints the value and then recursively calls itself on left and right children.
 - traversePreOrder Method:** A public method that calls the private recursive "traversePreOrder" method on the root node.
 - traverseInOrder Method:** A public method that calls the private recursive "traverseInOrder" method on the root node.
- Code Editor:** The "Tree.java" file is open in the editor. The code implements a search operation and a pre-order traversal. The search operation uses a while loop to traverse the tree until it finds a node with the specified value or reaches a null node. The pre-order traversal is implemented using a recursive method that prints the current node's value and then traverses its left and right children.
- Toolbars and Status Bar:** The status bar at the bottom shows: "Build completed successfully in 1 s 890 ms (11 minutes ago)", "293 chars, 11 line breaks", "75:1 CRLF", "UTF-8", "4 spaces", and "Event Log".

Implementation Of Following Algorithm



The screenshot shows an IDE interface with the following details:

- Project:** BST
- File:** Tree.java
- Code Content:**

```
if(root == null)
    return;
System.out.println(root.value);
traversePreOrder(root.leftChild);
traversePreOrder(root.rightChild);

public void traverseInOrder(){
    traverseInOrder(root);
}

private void traverseInOrder(Node root){
    if(root == null)
        return;
    traverseInOrder(root.leftChild);
    System.out.println(root.value);
    traverseInOrder(root.rightChild);

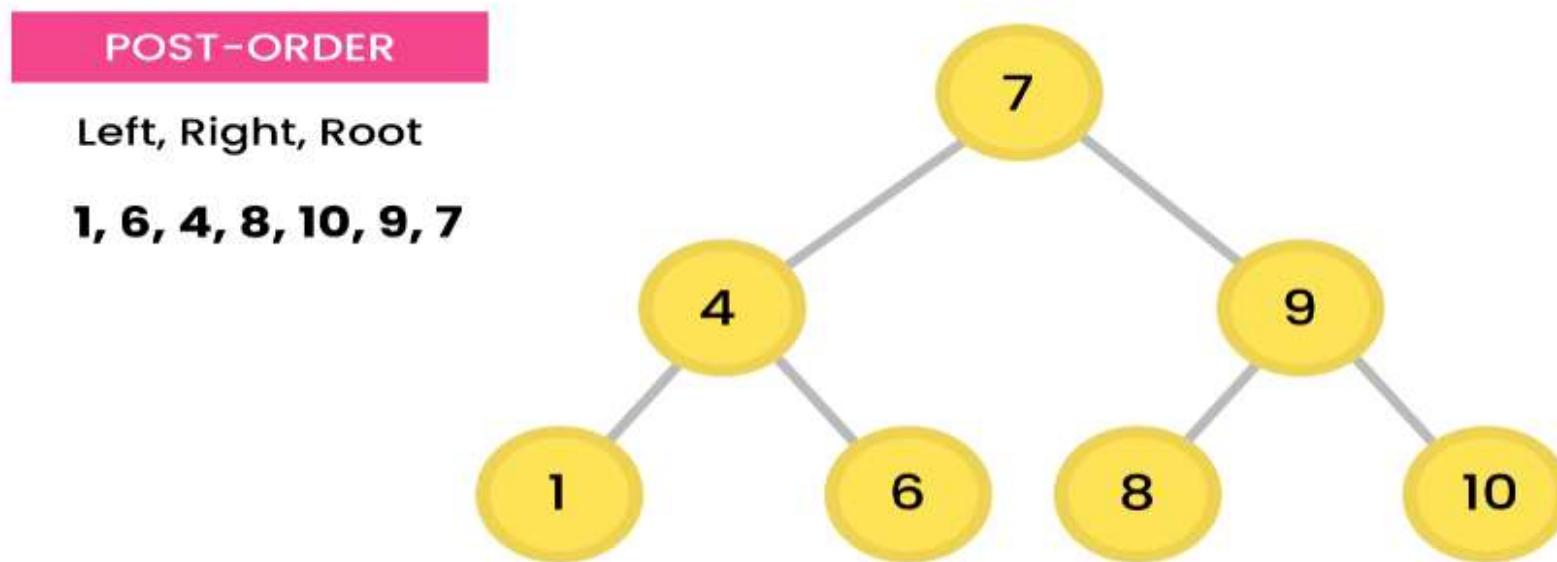
}

public void traversePostOrder(){
    traversePostOrder(root);
}

private void traversePostOrder(Node root){
    if(root == null)
        return;
    traversePostOrder(root.leftChild);
    traversePostOrder(root.rightChild);
}
```

- Toolbars:** Main
- Bottom Status Bar:** Build completed successfully in 1 s 890 ms (13 minutes ago), 288 chars, 11 line breaks, 87:6 CRLF, UTF-8, 4 spaces
- Event Log:** 1 Event Log

Implementation Of Following Algorithm



Post-Order Using Recursion

The screenshot shows a Java project named 'BST' in an IDE. The project structure is as follows:

- Project: BST
- src:
 - com.company:
 - Main
 - Tree
 - BST.iml
- External Libraries
- Scratches and Consoles

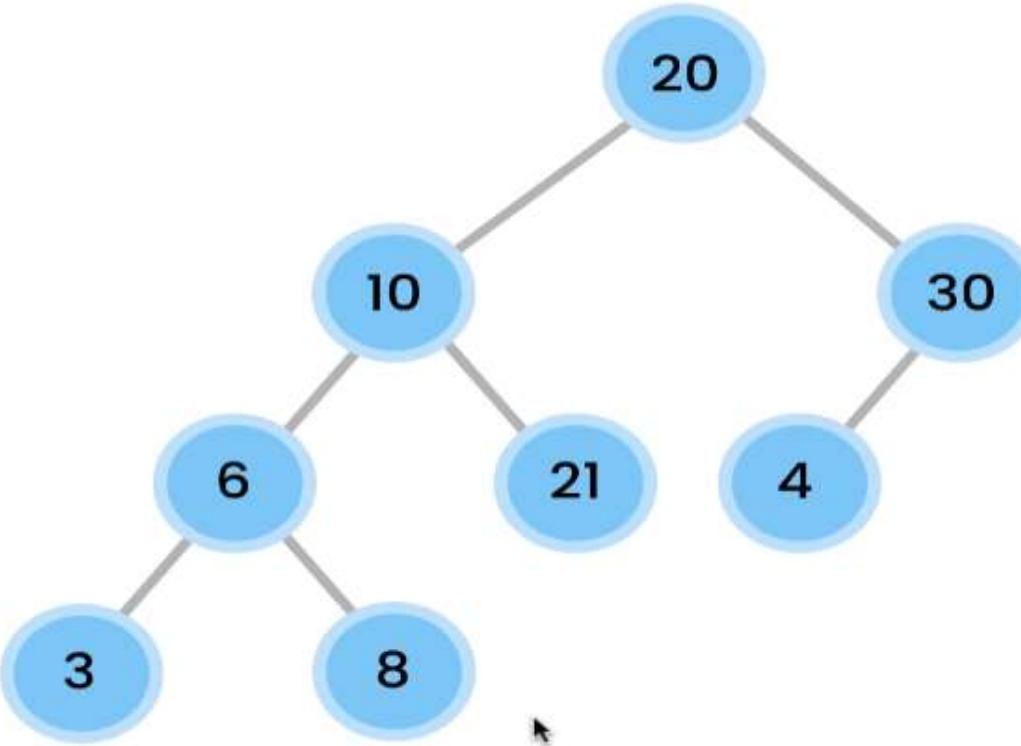
The 'Tree.java' file contains the following code:

```
private void traverseInOrder(Node root){  
    if(root == null)  
        return;  
    traverseInOrder(root.leftChild);  
    System.out.println(root.value);  
    traverseInOrder(root.rightChild);  
}  
  
public void traversePostOrder(){  
    traversePostOrder(root);  
}  
  
private void traversePostOrder(Node root){  
    if(root == null)  
        return;  
    traversePostOrder(root.leftChild);  
    traversePostOrder(root.rightChild);  
    System.out.println(root.value);  
}  
  
static final Node DELIMITER = null;  
  
/* public void printBST(){  
    ...  
}
```

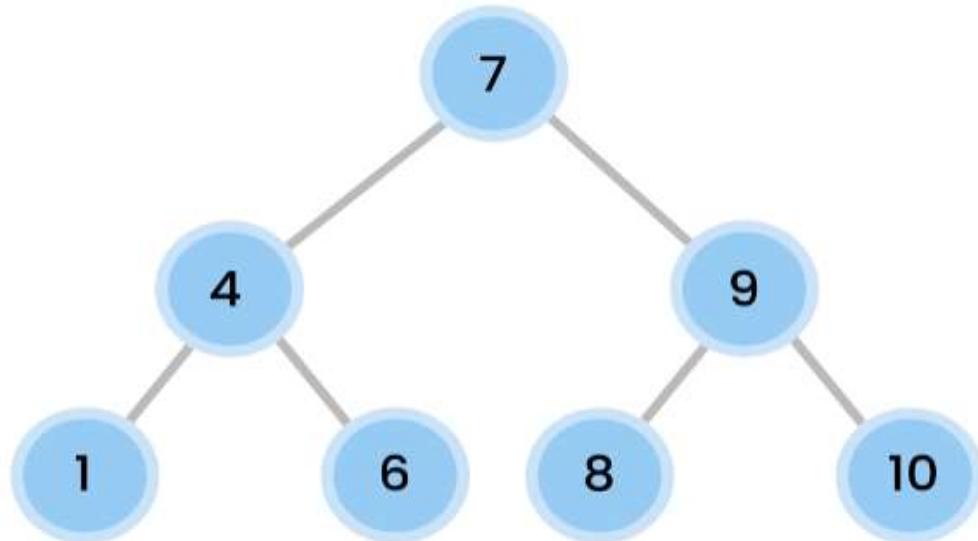
The code implements three methods: `traverseInOrder`, `traversePostOrder`, and `traversePostOrder`. The first method performs an in-order traversal. The second method calls the third method to perform a post-order traversal. The third method performs a post-order traversal by recursively visiting the left and right children before printing the current node's value. A static final node `DELIMITER` is defined at the bottom.

At the bottom of the interface, there are tabs for Run, Debug, TODO, Terminal, and Messages, along with status information: "Build completed successfully in 1 s 890 ms (12 minutes ago)", "298 chars, 11 line breaks", "100:1 CRLF", "UTF-8", "4 spaces", and an Event Log tab.

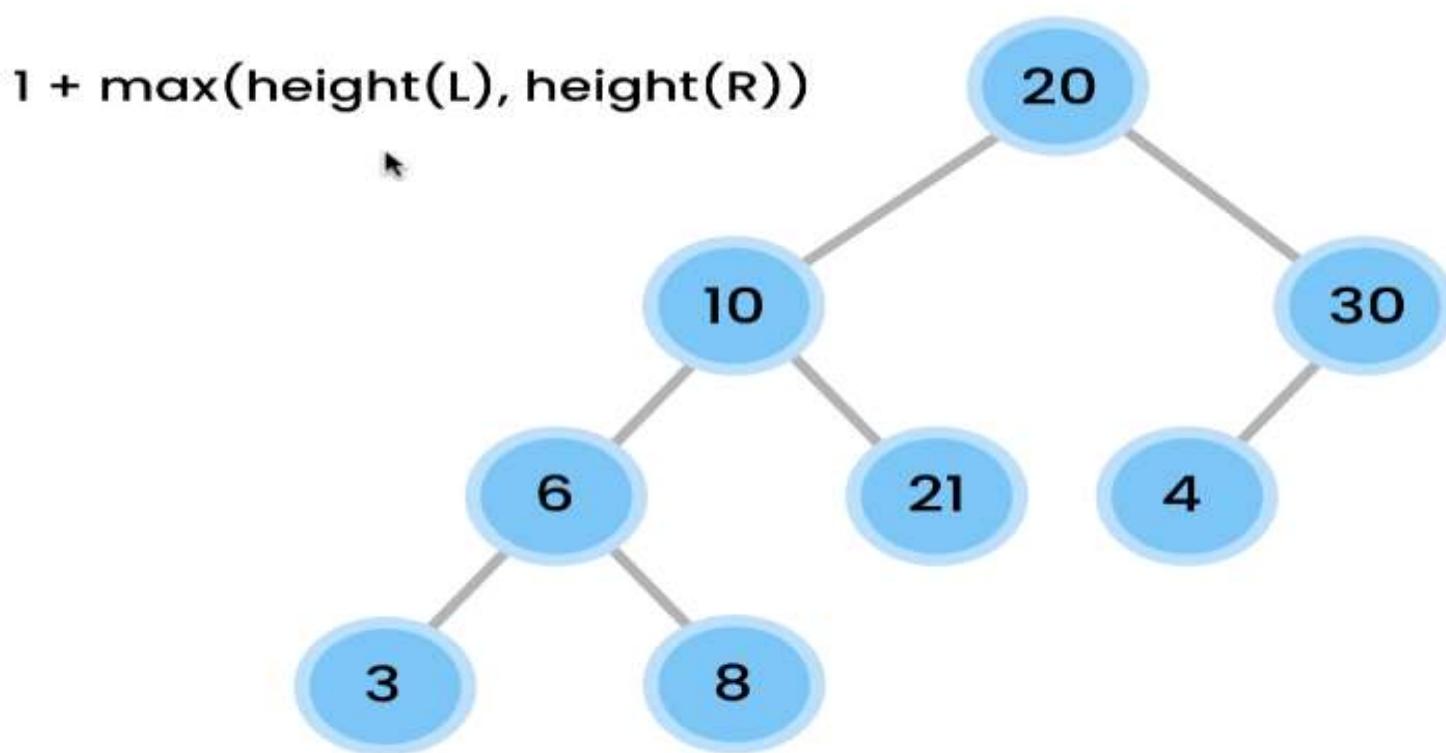
Height & Depth of BT



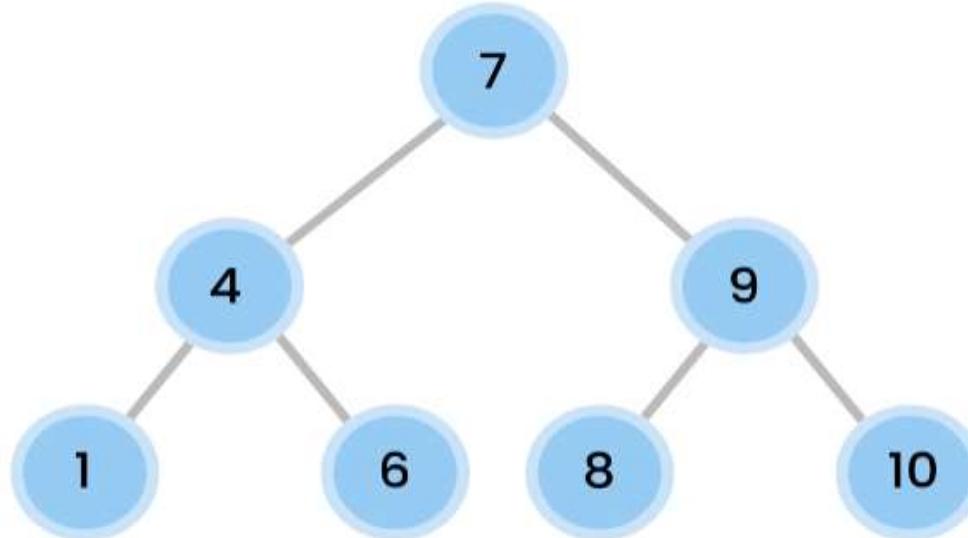
Height & Depth of BT.

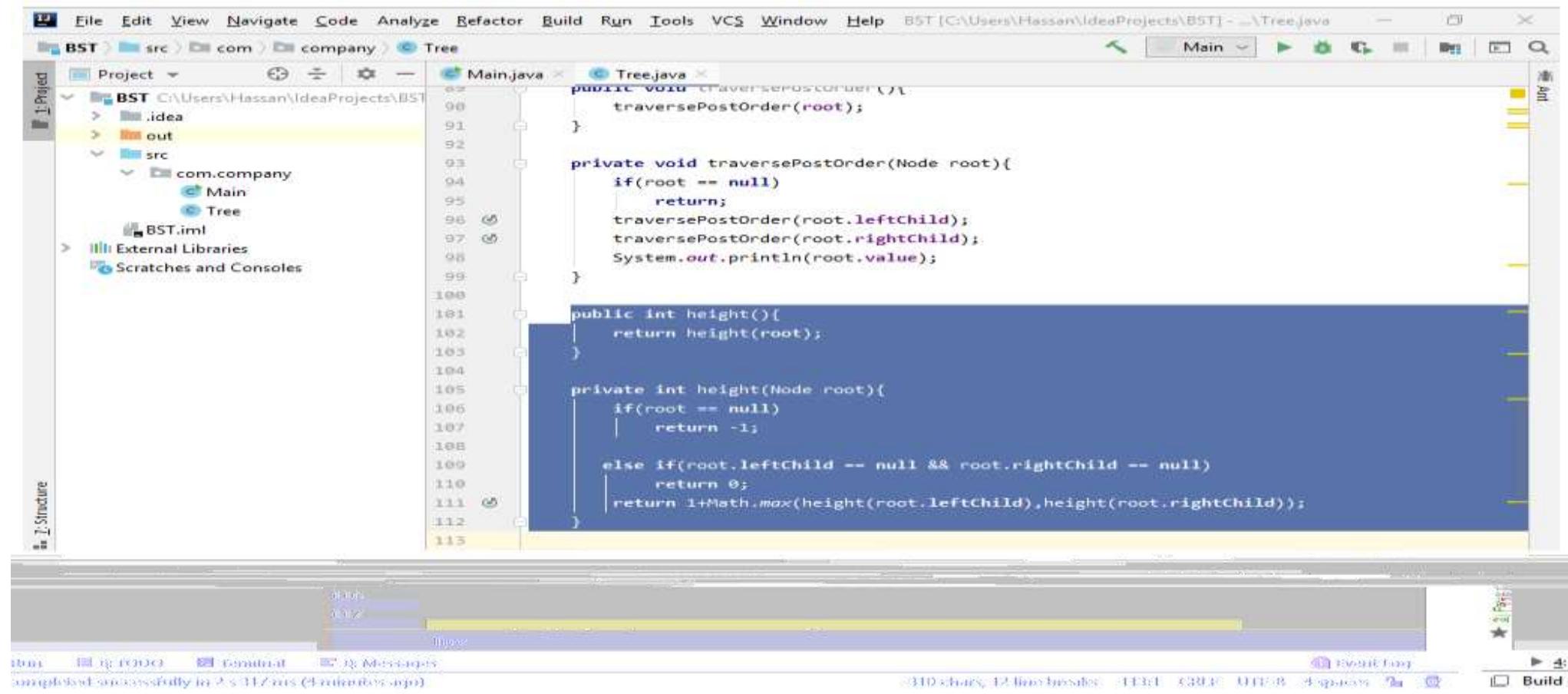


Height Formula (Recursive)



Implementation To Find Height Of BST





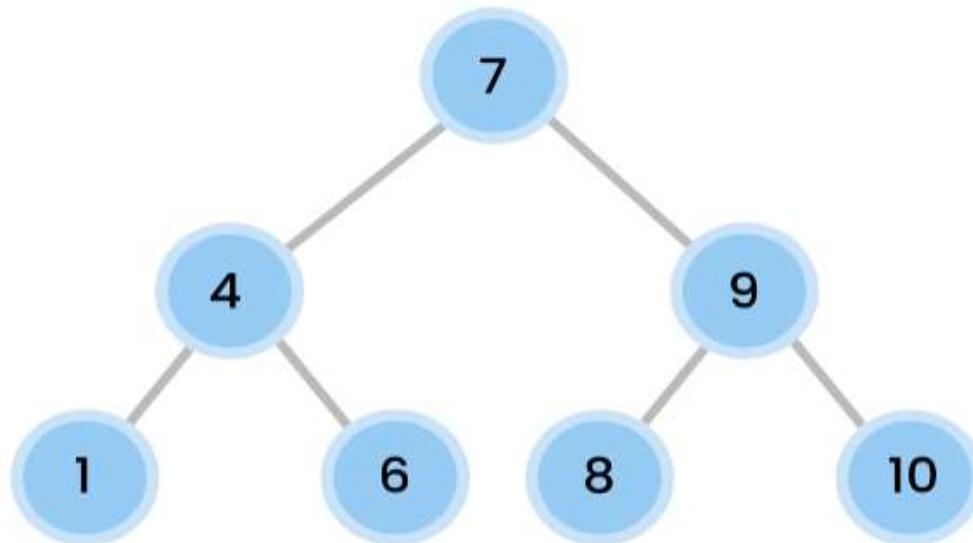
The screenshot shows an IDE interface with the following details:

- Project:** BST
- File:** Main.java
- Code Content:**

```
1 package com.company;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         // write your code here
7         Tree tree = new Tree();
8         tree.insert( value: 7);
9         tree.insert( value: 4);
10        tree.insert( value: 9);
11        tree.insert( value: 1);
12        tree.insert( value: 6);
13        tree.insert( value: 8);
14        tree.insert( value: 10);
15
16        //tree.traversePreOrder();
17        //tree.traverseInOrder();
18        //tree.traversePostOrder();
19        System.out.println("Height of Tree = "+tree.height());
20        System.out.println("Done");
21        //System.out.println(tree.find(8));
22    }
23
24 }
```

- Toolbars and Menus:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help.
- Bottom Status Bar:** Event Log, Build completed successfully in 2 s 317 ms (5 minutes ago), 54 chars, 19:63, CRLF, UTF-8, 4 spaces.

Implementation To Find Min Node Of BST



File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help BST [C:\Users\Hassan\IdeaProjects\BST] - Tree.java

Main.java Tree.java

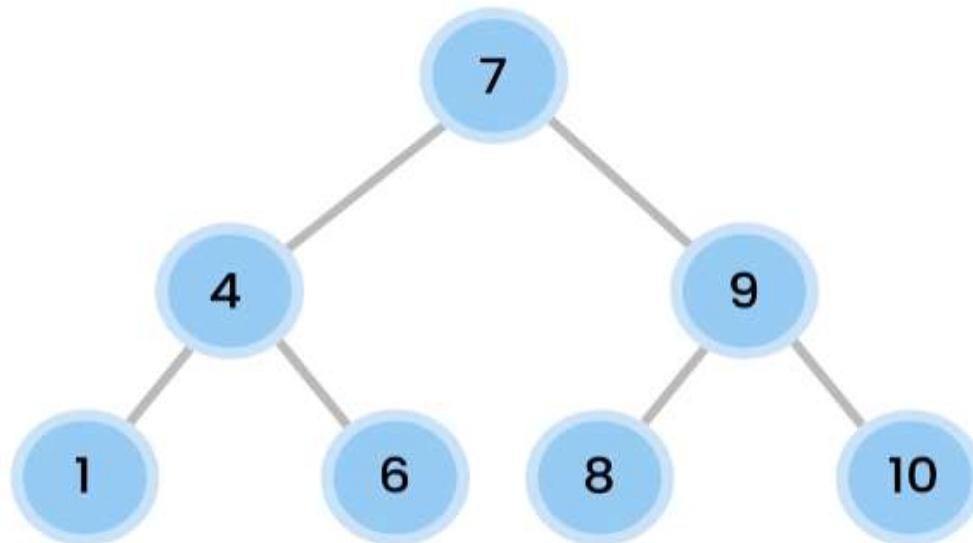
```
120 if(root == null)
121     throw new IllegalStateException();
122 var current = root;
123 var last = current;
124 while (current!=null){
125     last = current;
126     current = current.rightChild;
127 }
128 return last.value;
129
130
131
132 public int minNode(){
133     return minNode(root);
134 }
135
136 private int minNode(Node root){
137     if(root == null)
138         throw new IllegalStateException();
139     var current = root;
140     var last = current;
141     while (current!=null){
142         last = current;
143         current = current.leftChild;
144     }
145     return last.value;
146 }
147
148
```

Tree: > minNode()

Run TODO Terminal Messages Event Log

Build completed successfully in 1 s 942 ms (a minute ago) 365 chars, 14 line breaks 146:6 CRLF UTF-8 4 spaces

Implementation To Find Max Node Of BST



Screenshot of an IDE (IntelliJ IDEA) showing a Java project named "BST". The "src" package contains a "com.company" package with two classes: "Main" and "Tree". The "Tree.java" file is open, displaying the following code:

```
if(root == null)
    return -1;

else if(root.leftChild == null & root.rightChild == null)
    return 0;
return 1+Math.max(height(root.leftChild),height(root.rightChild));

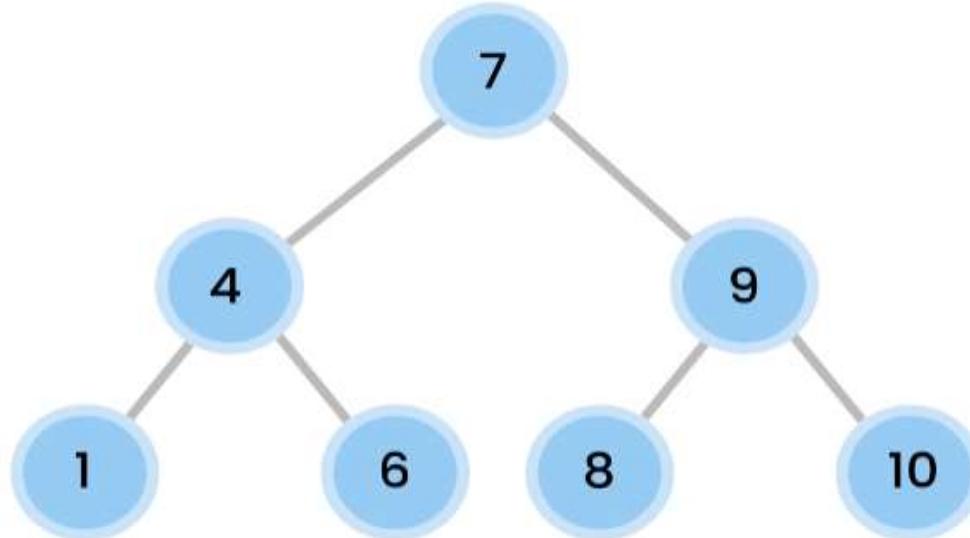
public int maxNode(){
    return maxNode(root);
}

private int maxNode(Node root){
    if(root == null)
        throw new IllegalStateException();
    var current = root;
    var last = current;
    while (current!=null){
        last = current;
        current = current.rightChild;
    }
    return last.value;
}

public int minNode(){
    return minNode(root);
}
```

The code implements a binary search tree with methods for finding the maximum and minimum values. A yellow dot is placed at the end of the closing brace of the `maxNode` method. The IDE interface includes toolbars, a navigation bar, and various status indicators at the bottom.

Implementation To Find Equality of BST



The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Bar:** BST
- File Bar:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help
- Current File:** BST [C:\Users\Hassan\IdeaProjects\BST] - Tree.java
- Code Editor:** Tree.java
- Project Structure:** BST (C:\Users\Hassan\IdeaProjects\BST)
 - .idea
 - .out
 - src
 - com.company
 - Main
 - Tree
 - BST.iml
 - External Libraries
 - Scratches and Consoles
- Toolbars:** Main
- Status Bar:** Build completed successfully in 1 s 972 ms (a minute ago), 514 chars, 15 line breaks, 164:6 CRLF, UTF-8, 4 spaces
- Event Log:** Event Log

```
141     while (current!=null){
142         last = current;
143         current = current.leftChild;
144     }
145     return last.value;
146 }
147
148
149     public boolean equals(Tree other){
150         if(other == null)
151             return false;
152         return equals(root,other.root);
153     }
154
155     private boolean equals(Node first,Node second){
156         if(first == null && second == null)
157             return true;
158         // Using Pre Order Traversal.
159         if(first != null && second != null)
160             return first.value == second.value && equals(first.leftChild,second.leftChild)
161             && equals(first.rightChild,second.rightChild);
162
163         return false;
164     }
165
166
167
168
169     static final Node DEFIMITTER = null;
Tree. > equals()
```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help BST [C:\Users\Hassan\IdeaProjects\BST] - Main.java

Main.java Tree.java

```
public static void main(String[] args) {
    // write your code here
    Tree tree = new Tree();
    tree.insert( value: 7);
    tree.insert( value: 4);
    tree.insert( value: 9);
    tree.insert( value: 1);
    tree.insert( value: 6);
    tree.insert( value: 8);
    tree.insert( value: 10);

    Tree tree2 = new Tree();
    tree2.insert( value: 7);
    tree2.insert( value: 4);
    tree2.insert( value: 9);
    tree2.insert( value: 1);
    tree2.insert( value: 6);
    tree2.insert( value: 8);
    tree2.insert( value: 10);

    //tree.traversePreOrder();
    //tree.traverseInOrder();
    //tree.traversePostOrder();
    System.out.println("Height of Tree = "+tree.height());
    System.out.println("Min Node of Tree = "+tree.minNode());
    System.out.println("Max Node of Tree = "+tree.maxNode());
    System.out.println(tree.equals(tree2));
    System.out.println("Done");
}
```

Main > main()

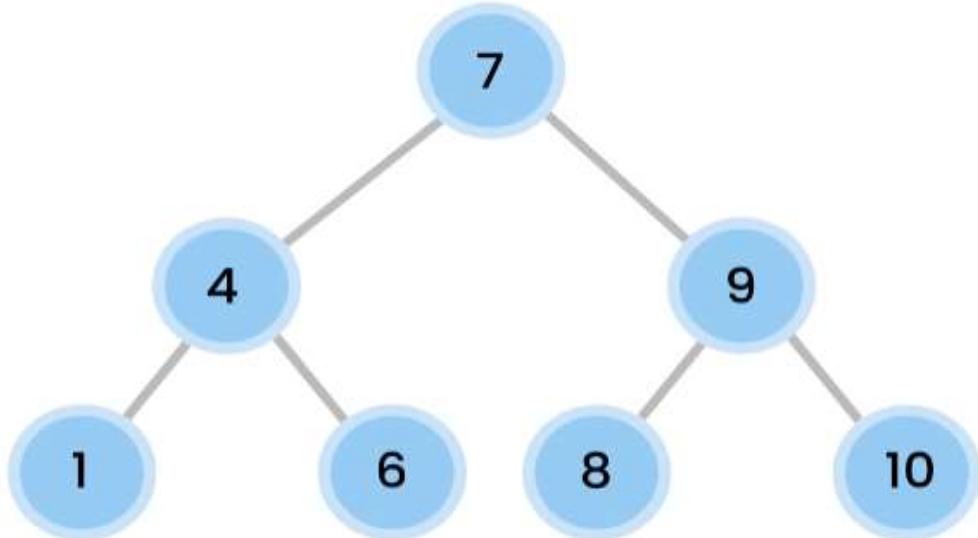
Run TODO Terminal Messages

Build completed successfully in 1 s 972 ms (a minute ago)

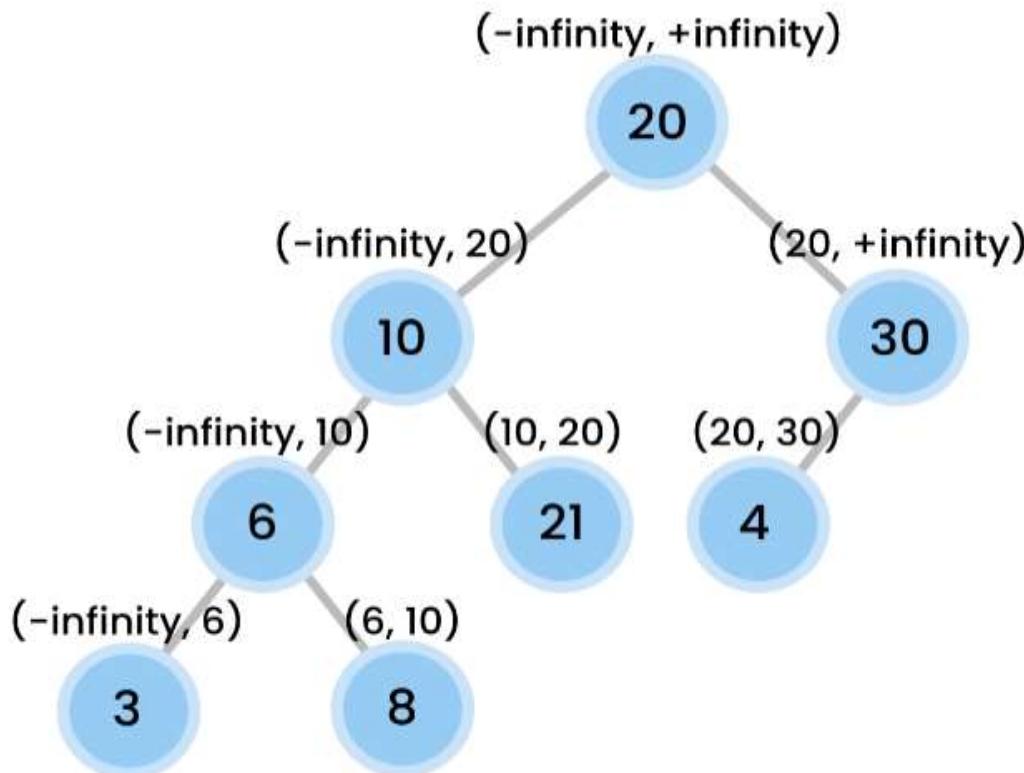
Event Log

31:45 CRLF UTF-8 4 spaces

Validating BST



Validating By Checking Range of Nodes



The screenshot shows an IDE interface with the following details:

- Project:** BST
- File:** Main.java
- Code Content:**

```
// Using Pre Order Traversal
if(first != null && second != null)
    return first.value == second.value && equals(first.leftChild,second.leftChild)
        && equals(first.rightChild,second.rightChild);

return false;

public void swapRoot(){
    var temp = root.leftChild;
    root.leftChild = root.rightChild;
    root.rightChild = temp;
}

public boolean isBinarySearchTree(){
    return isBinarySearchTree(root, Integer.MIN_VALUE, Integer.MAX_VALUE);
}

private boolean isBinarySearchTree(Node root,int min,int max){
    if(root == null)
        return true;
    if(root.value < min || root.value> max)
        return false;
    return
        isBinarySearchTree(root.leftChild,min, max: root.value-1)
        && isBinarySearchTree(root.rightChild, min: root.value+1,max);
}
```

- Toolbars and Menus:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help.
- Bottom Status Bar:** Event Log, Build completed successfully in 2 s 3 ms (moments ago), 169:32 CRLF UTF-8 4 spaces.

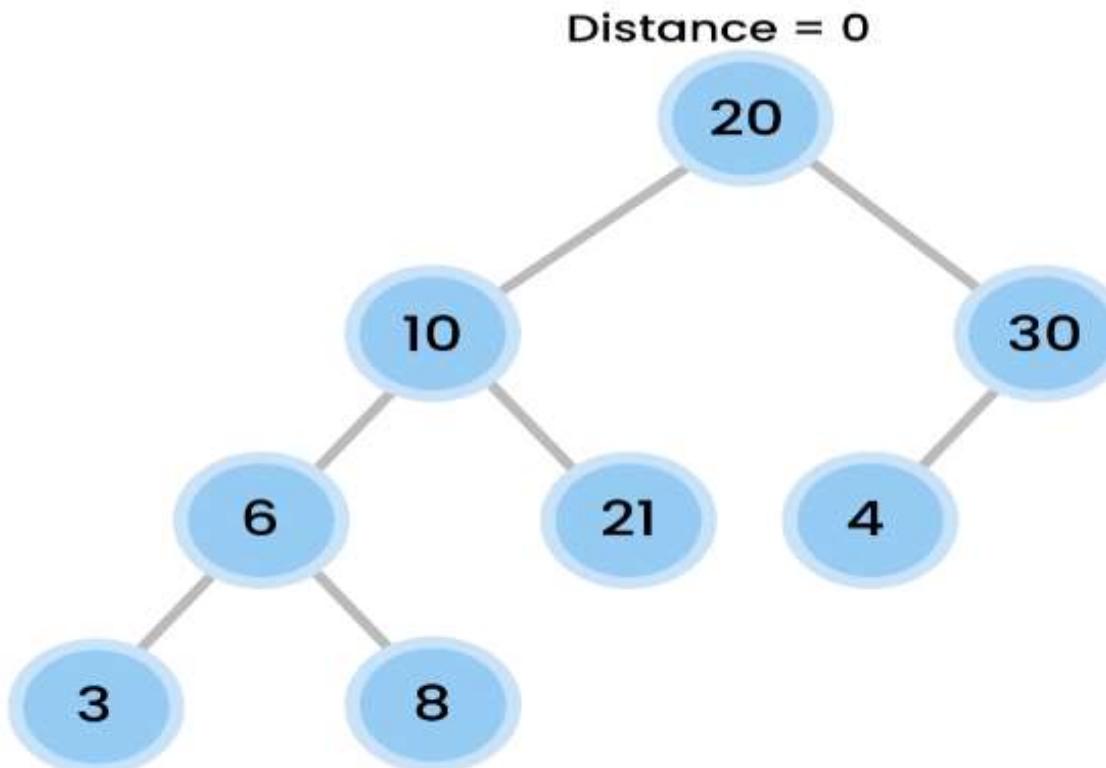
The screenshot shows an IDE interface with the following details:

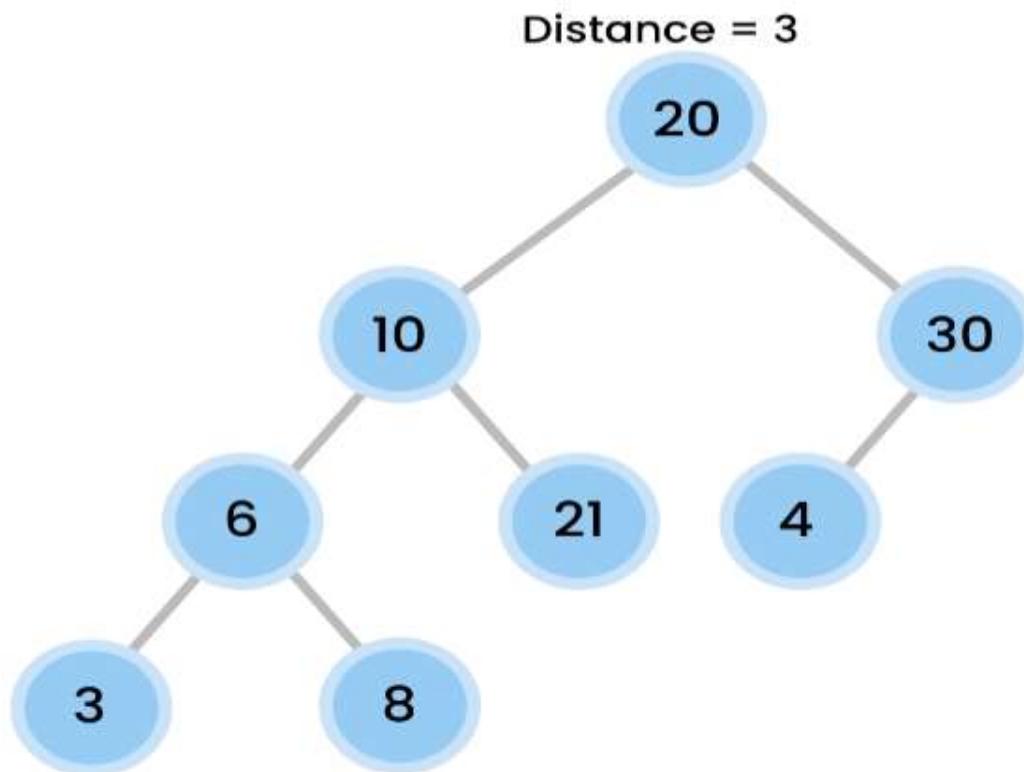
- Project:** BST
- File:** Main.java
- Code Content:**

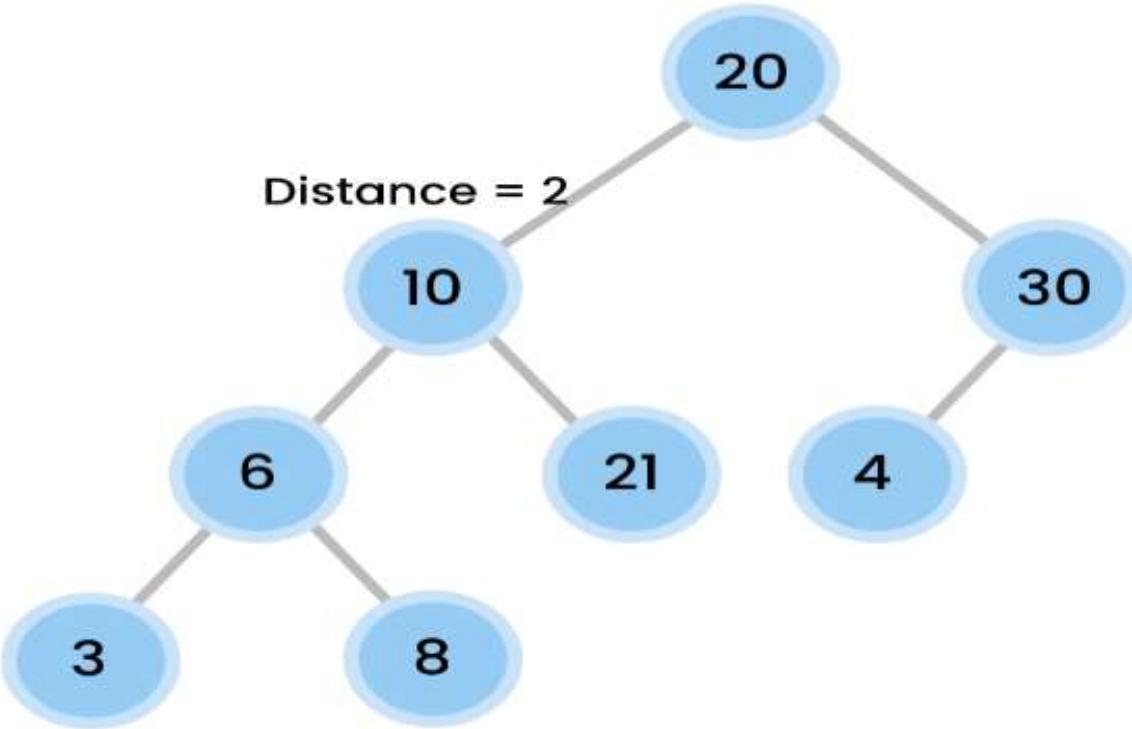
```
15     Tree tree2 = new Tree();
16     tree2.insert( value: 7);
17     tree2.insert( value: 4);
18     tree2.insert( value: 9);
19     tree2.insert( value: 1);
20     tree2.insert( value: 6);
21     tree2.insert( value: 8);
22     tree2.insert( value: 10);
23
24     //tree.traversePreOrder();
25     //tree.traverseInOrder();
26     //tree.traversePostOrder();
27     System.out.println("Height of Tree = "+tree.height());
28     System.out.println("Min Node of Tree = "+tree.minNode());
29     System.out.println("Max Node of Tree = "+tree.maxNode());
30     System.out.println(tree.equals(tree2));
31     tree2.swapRoot();
32     System.out.println(tree2.isBinarySearchTree());
33     System.out.println("Done");
34     //System.out.println(tree.find(8));
35
36 }
37
38 }
```

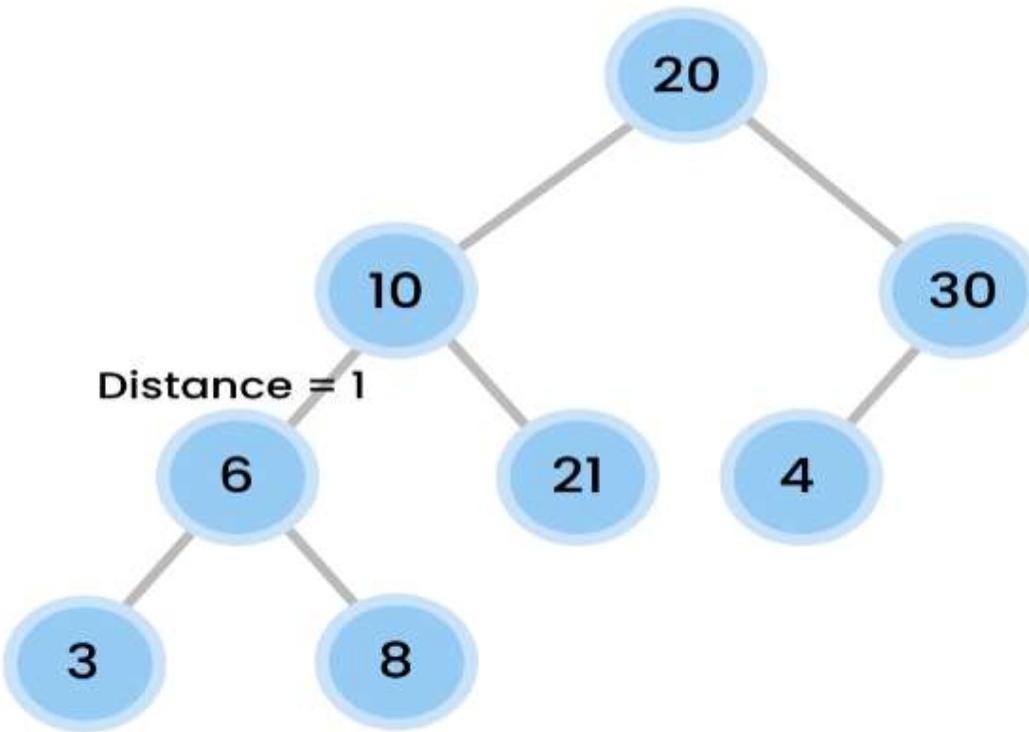
- Toolbars and Menus:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help.
- Bottom Status Bar:** Main > main(), Event Log, Build completed successfully in 2 s 3 ms (a minute ago), 32:26 CRLF UTF-8 4 spaces.

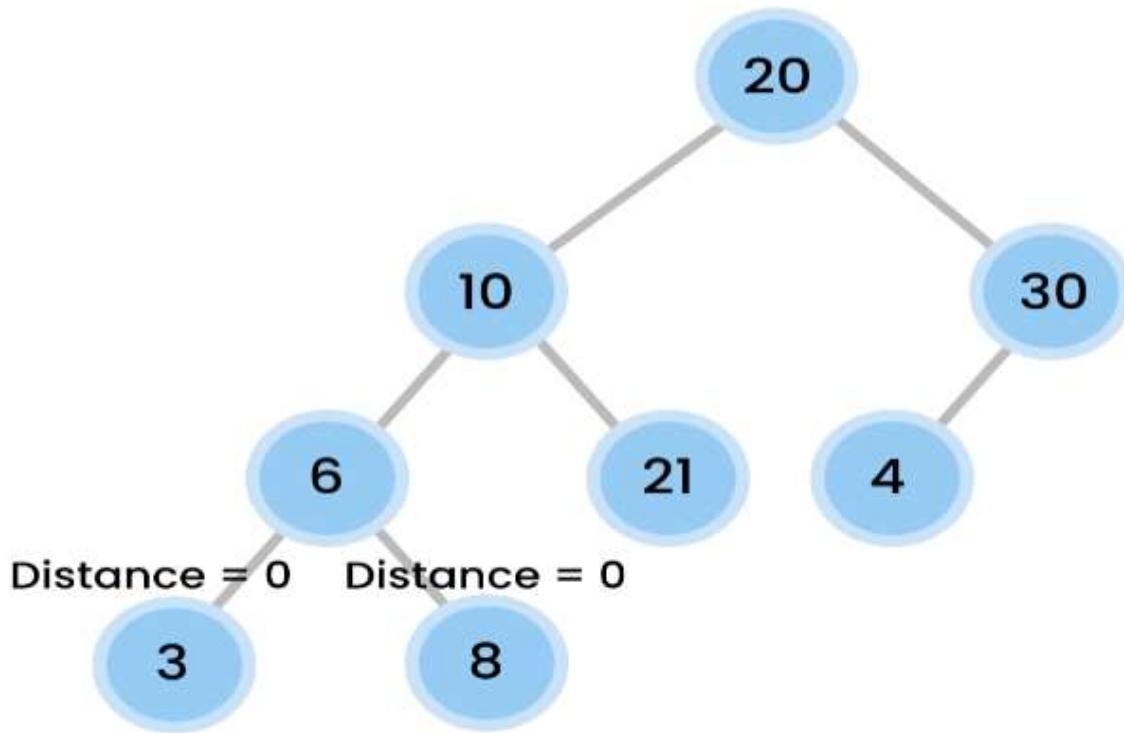
Nodes at K Distance from the Root Node



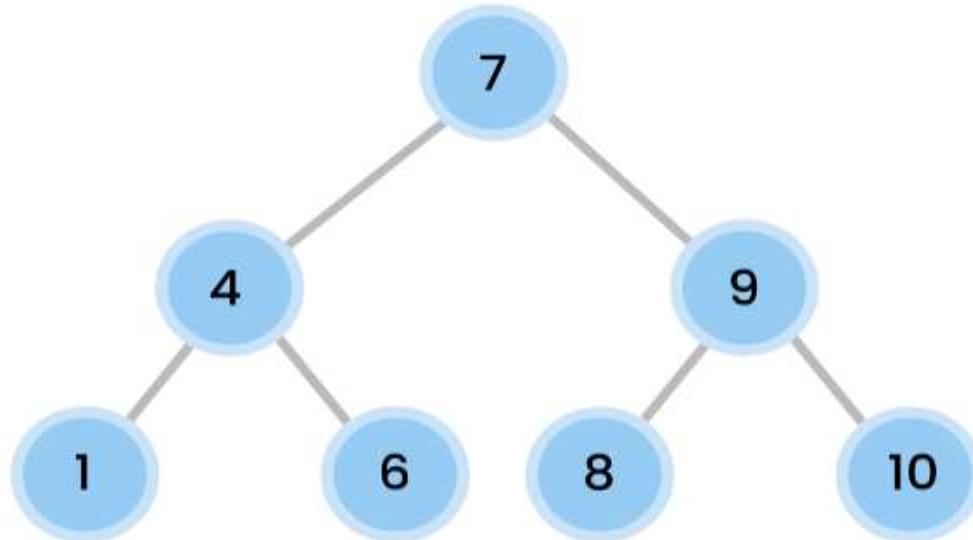








Implementation To Find K Distance of Nodes In Following BST



The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The left sidebar shows a project named "BST" containing a ".idea" file, an "out" folder, and a "src" directory. Inside "src", there is a package "com.company" containing classes "Main" and "Tree".
- File Tabs:** The top bar has tabs for "Main.java" and "Tree.java".
- Code Editor:** The main area displays the "Tree.java" code:

```
        return true;
    if(root.value < min || root.value > max)
        return false;
    return
        isBinarySearchTree(root.leftChild,min, max:root.value-1)
        && isBinarySearchTree(root.rightChild, min: root.value+1,max);

    public void printNodesAtDistance(int distance){
        printNodesAtDistance(root,distance);
    }

    private void printNodesAtDistance(Node root,int distance){

        if(root == null)
            return;
        else if(distance == 0){
            System.out.println(root.value);
            return;
        }
        printNodesAtDistance(root.leftChild, [distance]-1);
        printNodesAtDistance(root.rightChild, [distance]-1);
    }

    static final Node DELIMITER = null;
}
```
- Toolbars and Status Bar:** The bottom of the screen includes toolbars for Run, TODO, Terminal, and Messages, along with a status bar showing "Build completed successfully in 5 s 931 ms (a minute ago)", "438 chars, 17 line breaks 203:1 CRLF UTF-8 4 spaces", and an Event Log tab.

The screenshot shows an IDE interface with the following details:

- Project:** BST
- File:** Main.java
- Code Content:**

```
tree2.insert( value: 7);
tree2.insert( value: 4);
tree2.insert( value: 9);
tree2.insert( value: 1);
tree2.insert( value: 6);
tree2.insert( value: 8);
tree2.insert( value: 10);

//tree.traversePreOrder();
//tree.traverseInOrder();
//tree.traversePostOrder();
System.out.println("Height of Tree = "+tree.height());
System.out.println("Min Node of Tree = "+tree.minNode());
System.out.println("Max Node of Tree = "+tree.maxNode());
System.out.println(tree.equals(tree2));
tree2.swapRoot();
System.out.println(tree2.isBinarySearchTree());

tree.printNodesAtDistance(0);
System.out.println("Done");
//System.out.println(tree.find(8));
```

- Toolbars and Menus:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help.
- Status Bar:** Build completed successfully in 5 s 931 ms (2 minutes ago), 29 chars, 35:38, CRLF, UTF-8, 4 spaces.
- Event Log:** An event log icon is visible in the bottom right corner.