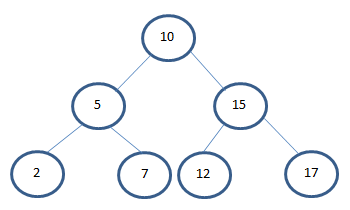
Binary Search Tree InOrder traversal using recursion.

In previous blog we studied what is tree and how to insert data into the tree. To read previous blog click [here](http://data-structure-learning.blogspot.com/2015/05/part-1-binary-search-tree-introduction.html).

The most important thing to solve problems on tree is to learn tree traversals. To name a few InOrder, PreOrder, PostOrder, Bread-First-Search.

In this post we will study recursive InOrder traversal. Consider the following tree.



3 step description for InOrder traversal

Traverse left subtree by calling function inOrder recursively.

Display the current element.

Traverse the right subtree by calling inOrder function recursively.

/\*\*

\* In order.

\*

\* **@param** localRoot the local root

\*/

**public** **void** inOrder(Node localRoot) {

/\*\*

\* Base condition to end recursion.

\* \*/

**if** (localRoot != **null**) {

//Step 1: Recursively traverse left subtree.

inOrder(localRoot.leftChild);

//Step 2: Print the current node.

localRoot.displayNode();

//Step 3: Recursively traverse right subtree.

inOrder(localRoot.rightChild);

}

}

Output:

{ 2 } { 5 } { 7 } { 10 } { 12 } { 15 } { 17 }

Now if you see the output closely then the output values of nodes are sorted. So InOrder gives us output of binary search tree in sorted order.

InOrder is used check whether the Binary Search Tree is properly populated or not. In other terms to check whether insert(int val) method is working properly or not.

In next post we will see how to traverse the BST (Binary Search Tree) using InOrder without recursion.

Click here to see next post.