Find a node in BST.

This is 7th post for the Tree section. Previously, I have written post on Inserting node in BST, Inorder Traversal ([Recursive](http://data-structure-learning.blogspot.com/2015/05/part-2-bst-inorder-recursion.html) and [Non-Recursive](http://data-structure-learning.blogspot.com/2015/05/binary-search-tree-inorder-traversal.html)), Preorder ([Recursive](http://data-structure-learning.blogspot.com/2015/05/part-4-bst-preorder-traversal-by.html) and [Non-Recursive](http://data-structure-learning.blogspot.com/2015/05/part-5-bst-preorder-traversal-by-non.html)) and [Size](http://data-structure-learning.blogspot.com/2015/05/size-of-bst-or-total-nodes-in-bst.html) of the tree.

In this post I will write how to search for a particular Value in tree.

We will search for Value in tree.

Let us consider following tree:

Let us try to search all Values of nodes. So as we know that Binary search tree has property that left subtree will have values less than root. And right subtree has values greater than root.

|  |  |
| --- | --- |
| **Search for Value** | **How to search** |
| Search for 10 | 10 is root we found it return true. |
| Search for 5 | 5<10. Go to left child of current node. 5 is current node. Found the value. Return true. |
| Search for 2 | 2<10 go to left child. 2<5 go to left child. 2 is current node. Found the value. Return true. |
| Search for 7 | 7<10 go to left child. 7>5 go to right child. 7 is current node. Found the value. Return true. |
| Search for 15 | 15>10 go to right child. 15 is current node. Found the value. Return true. |
| Search for 12 | 12>10 go to right child. 12<15 go to left child. 12 is current node. Found the value. Return true. |
| Search for 17 | 17>10 go to right child. 17>15 go to right child. 17 is current node. Found the value.  Return true. |

Following is the code that returns true if value exists or returns false if value does not exists.

/\*\*

\* Find element in tree.

\*/

**public** **boolean** find(**int** key) {

//Take a copy for root.

Node current = root;

//We will check whether current node's data is the key to be found.

**while** (current.iData != key) {

//if current node's data is greater then search for left-subtree

**if** (current.iData > key){

current = current.leftChild;

}

//if current node's data is lesser then search for right-subtree

**else**{

current = current.rightChild;

}

//If it is null then return false;

**if** (current == **null**)

**return** **false**;

}

**return** **true**;

}