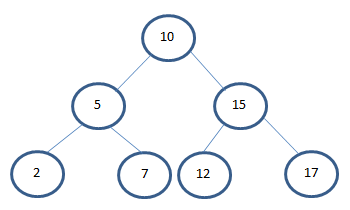
Binary Search Tree PreOrder traversal using Non-Recursion.

In previous post we studied how to traverse a tree by PreOrder Recursion. To see how to insert a node in tree click [here](http://data-structure-learning.blogspot.com/2015/05/part-1-binary-search-tree-introduction.html).

In this post we will study how to traverse a tree by Non-Recursion PreOrder traversal.



To reiterate how to traverse a tree in PreOrder traversal by recursion we follow 3 steps:

3 step description for PreOrder traversal

Display the current element.

Traverse left subtree by calling function preOrder recursively.

Traverse the right subtree by calling preOrder function recursively.

We will modify the above algorithm and traverse without recursion.

So a modified algorithm will look like following:

Use Stack<E> class of Java Collection Framework to store the nodes.

Stack<Node> s = **new** Stack<Node>();

Store root in temp as we don’t want root to move anywhere.

Node temp = root;

Now add the temp to stack.

stack.add(temp);

Now check if stack is empty

**while** (!s.isEmpty())

Now pop the node from stack and display it. This step is similar to step 1 in recursion traversal.

temp = stack.pop();

temp.displayNode();

Next step is to push right child in stack. BUT, in recursion it says that left child is pushed then right child is pushed in stack. Well the thing is when we pop element from stack only top element is popped. So if we just push left children in stack then after sometime we will consume all left children and stack will be empty. So we put the right child in stack first then insert left child.

/\*\*

\* Push the right child to keep track of them

\* for future traversal

\* \*/

**if** (temp.rightChild != **null**) {

stack.push(temp.rightChild);

}

/\*\*

\* Push left child on stack.

\* As left child is on top it will be popped

\* in next step.

\* \*/

**if** (temp.leftChild != **null**) {

stack.push(temp.leftChild);

}

We are now done with the PreOrder traversal without recursion. Now let us write entire code together for better visibility. I will comment the code so it is easy to understand it.

/\*\*

\* PreOrder traversal Non-Recursion

\* \*/

**public** **void** preOrderNon(Node root) {

//if root is null then tree is empty

**if** (root == **null**) {

**return**;

}

//We don't want to move root so copy it to temp.

Node temp = root;

//Define a new stack to store elements into it.

Stack<Node> stack = **new** Stack<Node>();

//Add root to stack. temp has root right now.

stack.add(temp);

//Run the loop till stack is not empty.

**while** (!stack.isEmpty()) {

/\*\*

\* Pop the element and display it.

\*/

temp = stack.pop();

temp.displayNode();

/\*\*

\* Push the right child to keep track of them

\* for future traversal

\* \*/

**if** (temp.rightChild != **null**) {

stack.push(temp.rightChild);

}

/\*\*

\* Push left child on stack.

\* As left child is on top it will be popped

\* in next step.

\* \*/

**if** (temp.leftChild != **null**) {

stack.push(temp.leftChild);

}

}

}

Output:

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

In next post we will see how to traverse PostOrder by Recursion and also PostOrder by Non-Recursion.