September 29, 2021

TREE TRAVERSALS

Kulwinder Kaur
kulwinder3213

DURATION
11min

CATEGORIES
Competitive Programming
Graph Theory

TAGS
Problem Solving Trees Java

DEFINITION

Tree traversal or traversing a tree is simply visiting each node and subtree of a tree. We can call any tree to be traversed once we are done with visiting each node and leaf of a tree.

WHY TREE TRAVERSAL ALGORITHMS ARE REQUIRED

Trees have a hierarchical structure and each node contains some of the information. So, to get that information we must visit each node to get the required data from the tree, and to traverse the whole tree we need traversing algorithms.

Note: Please read the previous article, “Tree Data Structure” for tree terminology.

TREE TRAVERSAL ALGORITHMS

There are two types of algorithms: depth first algorithms and breadth first algorithms.

DEPTH FIRST ALGORITHM:

There are three traversals given below.

INORDER TRAVERSAL:

(Left-root-right) This is when we traverse the left subtree (or child) of a tree first, then visit the root, and then the right subtree (or child). The same rule applies to its subtrees.

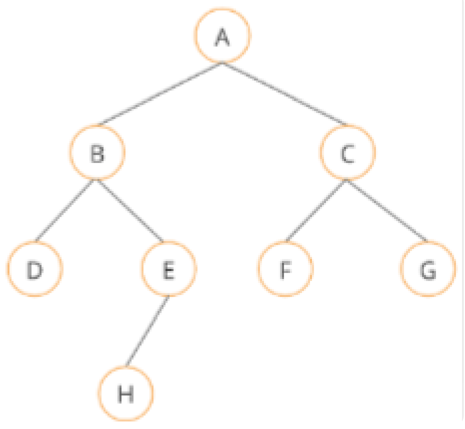


Figure 1

For the above Figure 1: Inorder traversal will be
in this order: **D B H E A F C G** (left - root - right)

Java implementation: There are two methods to traverse: recursive and iterative. The recursive approach is given below.

```java
public static void inOrder(Node root) {
  if (root == null) return;
  inOrder(root.left);
  System.out.print(root.data + "  ");
  inOrder(root.right);
}
```

PREORDER TRAVERSAL:

(Root-left-right) In this instance we will visit the root first, then the left subtree (and traversing), and then traverse the right subtree.
For Figure 1 the preorder traversal will be: **A B D E H C F G** (root-left-right)

Java implementation: **recursive approach**

```java
public static void preOrder(Node root) {
  if (root == null) return;
  System.out.print(root.data + "  ");
  preOrder(root.left);
  preOrder(root.right);
}
```

POSTORDER TRAVERSAL:

支持

(Left-right-root) In postorder traversal we visit the left subtree first, then traverse the right subtree, and then visit the root. This applies to all the subtrees present for a particular node.

For Figure 1 the postorder traversal will be:

Postorder : **D H E B F G C A** (Left-Root-Right)

Java implementation: **recursive approach**

```
public static void postOrder(Node root) {
  if (root == null) return;
  preOrder(root.left);
  preOrder(root.right);
  System.out.print(root.data + "  ");
}
```

BREADTH FIRST ALGORITHM:

It states that a tree should be traversed level by level in increasing numbers from root node to the last level. According to BFA we need to visit the root of a tree, then the root's left child, then the right child has to be visited. In the same way we can traverse the whole tree by level.

For Figure 1 tree zig zag or breadth first traversal will be: **A B C D E F G H**

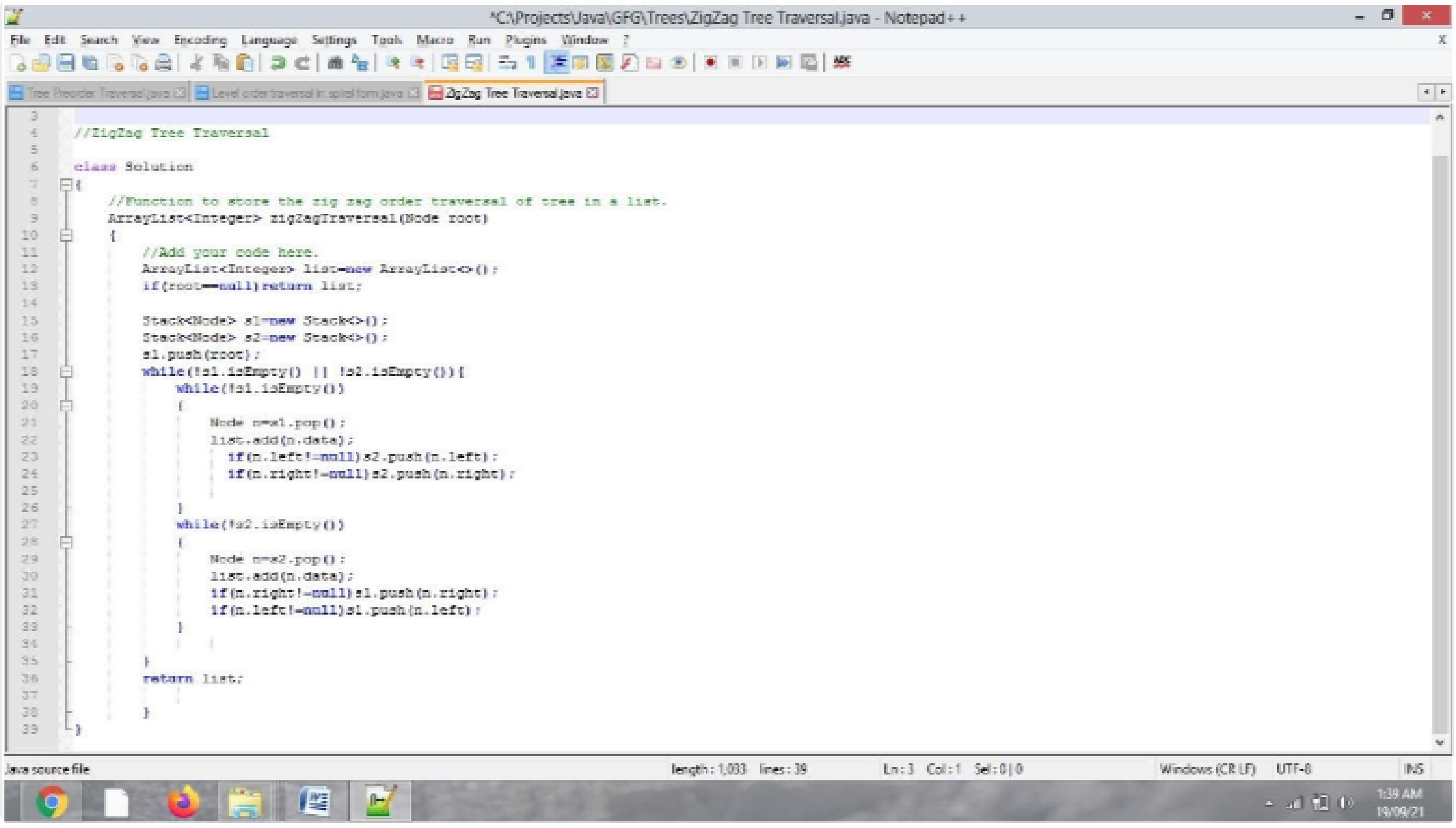
Java implementation:

```
class Solution {
  //Function to store the zig zag order traversal of tree in a list.
  ArrayList < Integer > zigZagTraversal(Node root) {
    //Add your code here.
    ArrayList < Integer > list = new ArrayList < > ();
    if (root == null) return list;

    Stack < Node > s1 = new Stack < > ();
    Stack < Node > s2 = new Stack < > ();

    s1.push(root);

    while (!s1.isEmpty() || !s2.isEmpty()) {

      while (!s1.isEmpty()) {
        Node n = s1.pop();
```

```java
        list.add(n.data);
        if (n.left != null) s2.push(n.left);
        if (n.right != null) s2.push(n.right);

      }
      while (!s2.isEmpty()) {
        Node n = s2.pop();
        list.add(n.data);
        if (n.right != null) s1.push(n.right);
        if (n.left != null) s1.push(n.left);
      }
    }
    return list;
  }
}
```

In the above code we are using two stacks to store the nodes for each level and the inner loop. For each stack one will print (or add in to list) the nodes and also add the next level node to stack two so that for the next run of the outer loop it can be used to process other nodes.

Code Snippet:



*C:\Projects\Java\GFG\Trees\ZigZag Tree Traversal.java - Notepad++

```java

//ZigZag Tree Traversal

class Solution
{
    //Function to store the zig zag order traversal of tree in a list.
    ArrayList<Integer> zigZagTraversal(Node root)
    {
        //Add your code here.
        ArrayList<Integer> list=new ArrayList<>();
        if(root==null)return list;

        Stack<Node> s1=new Stack<>();
        Stack<Node> s2=new Stack<>();
        s1.push(root);
        while(!s1.isEmpty() || !s2.isEmpty()){
            while(!s1.isEmpty())
            {
                Node n=s1.pop();
                list.add(n.data);
                  if(n.left!=null)s2.push(n.left);
                  if(n.right!=null)s2.push(n.right);

            }
            while(!s2.isEmpty())
            {
                Node n=s2.pop();
                list.add(n.data);
                if(n.right!=null)s1.push(n.right);
                if(n.left!=null)s1.push(n.left);
            }

        }
        return list;

        }
}
```

Java source file   length : 1,033   lines : 39   Ln : 3   Col : 1   Sel : 0 | 0   Windows (CR LF)   UTF-8   INS

File
Edit
Search
View
Encoding
Language
Settings
Tools
Macro
Run
Plugins
Window
?
Tree Preorder Traversal.java
Level order traversal in spiral form.java
ZigZag Tree Traversal.java
1:39 AM
19/09/21

2 2

支持