https://course.acciojob.com/idle?question=fa5f5548-8224-452e-8531-49dbbf53f5f5

MEDIUM

Max Score: 40 Points

Zig Zag Traversal of Tree

Given the `root` node of a tree, print its nodes in zig zag order, i.e. print the first level left to right, next level right to left, third level left to right and so on.

Note

You need to complete the given function. The input and printing of output will be handled by the driver code.

Input Format

The first line of input contains a `string` represeting the nodes, `N` is to show null node.

Output Format

For each test case print the nodes of the tree in zig zag traversal.

Example 1

Input

```
1 2 3 4 5 6 7
```

Output

```
1 3 2 4 5 6 7
```

Explanation

```
Original tree was:
              1
           /    \
          2      3
         / \    / \
        4   5  6   7

After Zig Zag traversal, tree formed would be:

              1
           /    \
          3      2
         / \    / \
        4   5  6   7
```

Example 2

Input

```
5 8 7
```

Output

```
5 7 8
```

Explanation

```
Original Tree was:

        5
       / \
      8   7

New tree formed is:
        5
       / \
      7   8
```

Constraints

`1 <= n <= 10^5`

Value of any node is less than `2^32`

Topic Tags

- Queues
- Trees
- Stacks

My code

```
// in java
import java.util.LinkedList;
import java.util.Queue;
import java.io.*;
import java.util.*;



class Main {
    static Node buildTree(String str) {
        if (str.length() == 0 || str.charAt(0) == 'N') {
            return null;
        }
        String ip[] = str.split(" ");
        Node root = new Node(Integer.parseInt(ip[0]));
        Queue<Node> queue = new LinkedList<>();
        queue.add(root);
        int i = 1;
        while (queue.size() > 0 && i < ip.length) {
            Node currNode = queue.peek();
            queue.remove();
            String currVal = ip[i];
            if (!currVal.equals("N")) {
                currNode.left = new Node(Integer.parseInt(currVal));
                queue.add(currNode.left);
            }
            i++;
            if (i >= ip.length) break;
            currVal = ip[i];
            if (!currVal.equals("N")) {
                currNode.right = new Node(Integer.parseInt(currVal));
                queue.add(currNode.right);
            }
            i++;
        }

        return root;
    }

    public static void main(String[] args) throws IOException {
        BufferedReader br =new BufferedReader(new
InputStreamReader(System.in));
        String s1 = br.readLine();
        Node root1 = buildTree(s1);
        Solution g = new Solution();
        g.binaryTreeZigZagTraversal(root1);
    }
}


class Node {
    int data;
    Node left;
    Node right;
    Node(int data) {
        this.data = data;
        left = null;
        right = null;
    }
}

class Solution {
    public static void binaryTreeZigZagTraversal(Node root) {
        //Your code here
                  Stack<Node> st1 = new Stack<>();
          Stack<Node> st2= new Stack<>();
                st1.push(root);
                while(true)
                        {
                                int f1=0;
                                        int f2=0;
                                while(!st1.empty())
                                        {
```

```
                f1=1;
                    Node t=st1.pop();
                    if(t.left!=null)
                st2.push(t.left);
                    if(t.right!=null)
                    st2.push(t.right);
                    System.out.print(t.data+" ");
                }

            while(!st2.empty())
                {
                f2=1;
                    Node t=st2.pop();
                    if(t.right!=null)
                    st1.push(t.right);
                    if(t.left!=null)
                   st1.push(t.left);

                    System.out.print(t.data+" ");
                }
            if(f1==0 && f2==0)
                break;
        }


  }
}
```